DVB-T2 Receiver: Physical Layer Simulator

Heřmánek Antonín, Mazanec Tomáš, Tichý Milan
{hermanek,mazanec,tichy}@utia.cas.cz
+420-2-6605 2470, +420-2-6605 2472

Contents

Revisions

Revision	Date	Author	Description of document changes
0	18.8.2009	T.M.	New document
1	19.8.2009	A.H.	First revision; content specification
2	27.8.2009	M.T.	Text corrections
3			
4			

1 Introduction

The following document describes the DVB-T2 software package for Matlab. The package contains the utilities and test benches for signal reception of the second generation digital terrestrial television broadcasting system (DVB-T2) that is standardized by the European telecommunication standard ETSI EN 302 755 [1].

The software package is a product of cooperation between UTIA AS CR (Czech republic) and Screen Service (Italy). The aim of the project is to built a professional DVB-T2 receiver for signal measurements and testing of the DVB-T2 broadcasting system.

The fundamental DVB-T2 functions are distributed in the binary form of Matlab *p-code*, while the test benches are provided in the standard Matlab *m-code*.

2 Description of Implemented Parts of the DVB-T2 Receiver

The software package contains routines for testing of the following receiver parts: P1 symbol processing, time and frequency synchronization, OFDM symbol demodulation, channel equalization and frequency de-interleaving. The structure of the UTIA part of DVB-T2 receiver is depicted in Figure 1. The functionality of individual blocks is roughly described here:

P1 processing consists of functions for DVB-T2 signal detection, i.e. P1 detection and decoding and coarse time and frequency offset estimation.

Time and frequency synchronization consists of blocks for fine time and frequency estimation, frequency offset correction end guard interval estimation.

OFDM demodulation and channel equalization consists of functions for OFDM demodulation using FFT (including the removal of unused frequency band for normal and extended mode), pilot pattern generation (continuous and scattered), channel estimation in frequency domain for P2 and data symbol and channel interpolation in frequency.

Frequency de-interleaving block provides de-interleaving of frequency bins with respect to the original data stream.

Most of the frame parameters are obtained from the input signal except for the info coded in P2 symbols. Namely these parameters are not decoded/detected: number of data symbols in the DVB-T2 frame, M-QAM order, FEC coding rate.

3 Package Structure and Naming Conventions

All Matlab files are located in the `src` directory. Test vectors are located in the `data` directory. Only one data set of test vectors is distributed with the package. The package follows these naming conventions:

- `rx_` – function implementing the receiver components
- `tx_` – function implementing the transmitter components

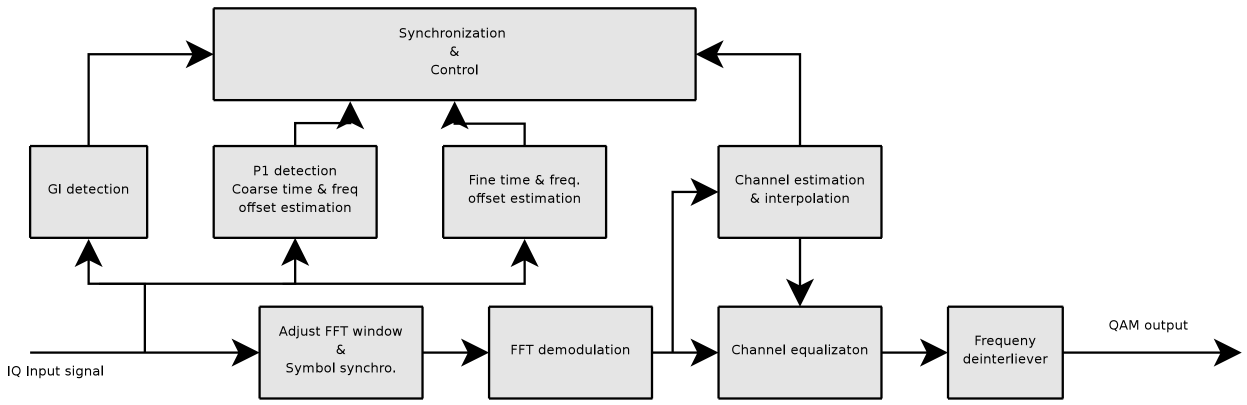

Figure 1: The block diagram of the UTIA part of DVB-T2 receiver

- `rxtx_` – function used "between" transmitter and receiver
- `test_` – test bench (see Section 4)
- no prefix – auxiliary or common function

All transmission parameters (FFT size, guard interval size, pilot pattern etc.) are specified by the structure `Sconf`.

The package shares the following common function:

- `rx_param()` – initializes or updates configuration structure `Sconf`; see the file `test_eq4`, lines 13 and 24 as an example
- `channel_siso()` – Single Input Single Output channel model; it implements three different channels:
  - `'flat'` – flat fading channel
  - `'freqsel1'` – frequency selective channel (Rayleigh fading) with high delay spread; the model parameters are:

    relative path delays $= [0\ 2\ 17\ 36\ 75\ 137]$

    path gains $= 10\log_{10}([1\ 0.3162\ 0.1995\ 0.1296\ 0.1\ 0.1])$
  - `'freqsel2'` – frequency selective channel (Rayleigh fading) with short delay spread; the model parameters are:

    relative path delays $= [0\ 1\ 3\ 4\ 5\ 7\ 8\ 12\ 17\ 24\ 29\ 49]$

    path gains $= 10\log_{10}([0.2478\ 0.1287\ 0.3088\ 0.4252\ 0.49\ 0.0365\ 0.1197\ 0.1948\ 0.4187\ 0.317\ 0.2055\ 0.1846])$

In all cases the Doppler shift is zero.

3.1 P1 Processing

The P1 processing consists of several functional blocks such as the DVB-T2 frame detector, position detector, P1 decoder performing OFDM (FFT), estimator of S1 and S2 parameters, which specify modulation mode, and functional module for coarse time and frequency synchronization.

The P1 detector provides information that a DVB-T2 signal is present in the received signal. It is also used for coarse time estimation of the the beginning of T2 frame. The UTIA implementation has followed several approaches, one based on the method proposed in the guidelines, the others based on the cross-correlation functions. The P1 processing functions are described in the following sections.

3.1.1 Detection Based on the Phase Shift Property of P1 Structure

This method follows directly the method proposed in the guidelines. The P1 detector consists of two parallel convolutions. Its structure is described in [2], Section 1.2.2.5 “P1 detection”. The output of the correlator peaks if the P1 symbol is present in the input signal. Peak maximum differs significantly according to the SNR applied and also according to the type of the channel (flat fading vs. frequency selective).

The rising edge of the peak is used for coarse time synchronization. The angle value of correlator output at the rising edge position is used for the coarse estimation of fractional frequency offset.

Matlab function:

```
function [y, indx_detected, y_cfo_est, eps_cfo_est] = rx_P1_detect(x, threshold)

  Inputs:
    x          -- dvb_t2 signal in a baseband (a row vector)
    threshold -- the initial threshold value

  Outputs:
    y              -- correlation function as defined in the guidelines
    indx_detected -- indices of P1 symbol start
    y_cfo_est      -- fine freq. offset esimation signal
    eps_cfo_est    -- fine freq. offset values for 'indx_detected'
```

3.1.2 Detection Based on the Cross-Correlation with a Pattern

This method is based on the cross-correlation between the input signal and P1 patterns. Depending on the S1 and S2 parameters, there are 128 possible signal patterns where only 18 are defined in the standard, others are reserved for future use. However, thanks to the way how the S1 and S2 fields are inserted into the P1 symbol, and thanks to the P1 symbol orthogonality in time and frequency domain, it is sufficient to test patterns only for different S1s, where S2 equal to all zeros.

The corresponding Matlab function works in two modes: 'full' and 'partial'. In the 'full' mode, it calculates cross-correlation with 8 different patterns (corresponding to 8 different S1 values) and returns the results of all correlations and the position of maximum peak in each correlation. In the 'partial' mode, there are two simplifications:

1. The pattern has been shortened to 256 samples.
2. The 'binary correlation' is calculated—only pattern signs and input signal are correlated.

The method is similar to those published in [3] and [5].

Matlab function:

```
function [yy, est_indices] = rx_P1_detect2(x, threshold,mode)

  Inputs:
    x         -- dvb_t2 signal in a baseband (a row vector)
    threshold -- the initial threshold value

  Outputs:
    y           -- cross-correlation outputs with patterns for all S1
    eps_indices -- indices of P1 symbol start for all patterns
```

3.1.3 P1 Demodulator and Decoder

This function demodulates the input P1 symbol using FFT and decodes and estimates S1 and S2 fields. Correspondingly, it sets the configuration vector parameters: FFT size, mode, S1, S2. The function also estimates the integer frequency offset.

Matlab function:

```
function [Param, OutputValues] = rx_P1_decode(Param, x, ifplot)

  Inputs:
    Param  -- parameter vector generated using rx_param() function
    x      -- input signal of size 1024, part A of the DVB-T2 P1 symbol
    ifplot -- if set to '1', it plots debug plots

  Outputs:
    OutputValues -- structure containing:
      frequency-shift -- integer frequency offset
      S_min           -- number of errors in S1 and S2 field
```

3.2 Time and Frequency Synchronization

The methods used for time and frequency synchronization include methods for fine time and frequency offset estimation and guard interval estimation. Functions implementing these methods are presented in the following sections.

3.2.1 Guard Interval Estimation

The function is based on the well known method based on cross-correlation of the OFDM symbol with the cyclic prefix. As an input it requires a vector containing both P2 and data part. Its size must be a multiple of FFT size.

Matlab function:

```
function [x, Sconf] = rx_GI_detect(Sconf, frame)

  Inputs:
    Sconf -- configuration vector
    frame -- one T2 frame excluding P1 symbol must be a multiple of FFT_size

  Outputs:
    x -- estimated guard interval size
```

3.2.2 Time Domain Time and Frequency Offset Estimation

The function uses the method based on cross-correlation of the OFDM symbol with the cyclic prefix. Time synchronization within a given T2-frame segment is determined by index where the correlation metrics Lambda has its maximum. Carrier frequency offset (CFO) is then determined as a value of `cfo_est` signal at the same index where Lambda has its maximum.

Matlab function:

```
function [Lambda, cfo_est] = rx_coarseEst_GI(Sconf, x)

  Inputs:
    Sconf -- DVB-T2 parameter vector
    x     -- at least two input symbols, all with GI included,
             length of x must be at least 2*symbol_size

  Outputs:
    Lambda  -- correlation metrics
    cfo_est -- CFO estimated output signal
```

Usage of the function `rx_coarseEst_GI`:

```
  [Lamda, cfo_est] = rx_coarseEst_GI(frame_segment);
  [value index] = max(Lambda);
  CFO_estimated = cfo_est(index);
```

Note that examples of usage of this function can be found in a variety of test-benches, for example in `test_ESTandEQ.m`, `testbench_TOinFreq.m`, `testbench_intCFOinFreq.m` etc.

3.2.3 Frequency Domain Time and Frequency Offset Estimation

Functions for frequency domain time and frequency offset estimation use several pilot-based methods operating in frequency domain. The methods were implemented for time offset, integer carrier frequency offset and fine carrier frequency offset estimations. Detailed description and analysis of these methods can be found in [4] and [6].

All methods use symbol pilots extracted from OFDM symbols. According to the DVB-T2 standard, P2 pilots are extracted from P2 symbols and continual or scattered pilots from data symbols.

Time offset on P2 pilots. This method enumerates average of a phase rotation of each P2-pilot. Scaling of this average value leads to exact time-offset value. Note that P2 pilots are extracted in normal bandwidth DVB-T2 mode, i.e. extended bandwidth sub-carriers are ignored.

Matlab function, where P2 pilots method is selected:

```
function time_offset = rx_fineTOinFreq(s_conf, rxp, txp, method)
  Inputs:
    s_conf -- configuration structure; see rx_param() function
    rxp    -- RX pilots matrix (vector), pilots extracted from received symbols
    txp    -- TX pilots matrix (vector), transmitted pilots
              (size of these matrices depends on selected method)
    method -- specific pilots to be used for time offset estimation
              possible methods are 'scattered' and 'P2'

  Outputs:
    time_offset -- time offset value (value can be rounded to integer)
```

The test bench that tests estimation of time offset in frequency domain under different conditions is `testbench_TOinFreq.m`. Based on receiver chain, test bench provides: P1 detection coarse and fine, T2-frames extraction, fine time synchronization on P2 and data symbols with fine CFO estimation, P2 and data symbols extraction with correction of estimated fine CFO. For estimation of time offset based on symbol pilots, following procedures are implemented: demodulation of P2 and data symbols, decoding of subcarrier order, generation of pilot sets, pilot extraction.

Fine carrier frequency offset on continual pilots. When possible, integer carrier frequency offset (CFO) is estimated and corrected, estimation of the remaining fine (fractional) carrier frequency offset is performed. Basic idea of fine CFO estimation is to examine phase rotation of pilots in a unit time interval. Correlating pilots of two consecutive symbols at the same carrier index gives a relation of constant phase difference and thus a relation to fine CFO. Averaging of this measure gives desired fine CFO estimate. The estimator uses Li and Zhang's algorithm [4].

Note that continual pilots are extracted according to Annex G of [1] but no priority of scattered pilot amplitudes has been taken to notice, i.e. all continual pilots are assumed to be continual. Also demodulation of the received continual pilots is omitted.

Matlab function, where continual pilots method is selected:

```
function epsl = rx_fineCFOinFreq(s_conf, rxp, txp, method)

  Inputs:
    s_conf -- configuration structure; see rx_param() function
    rxp    -- RX pilots matrix (vector), pilots extracted from received symbols
    txp    -- TX pilots matrix (vector), transmitted pilots
              (size of these matrices depends on selected method)
    method -- specific pilots to be used for time offset estimation
              possible methods are 'scattered' and 'P2'
  Outputs:
    epsl -- fine CFO estimate
```

The test bench that tests estimation of fine CFO in frequency domain under different conditions is `testbench_fineCFOinFreq.m`. Based on receiver chain, test bench provides: P1 detection coarse and fine, T2-frames extraction, fine time synchronization on P2 and data symbols with fine CFO estimation, P2 and data symbols extraction with correction of estimated fine CFO. For estimation of integer CFO based on symbol pilots, following procedures are implemented: demodulation of P2 and data symbols, decoding of subcarrier order, generation of pilot sets, pilot extraction.

Integer carrier frequency offset on continual pilots. Integer carrier frequency offset cause a cyclic shift of subcarrier indices in OFDM symbol. Whereas *integer* means that amount of this frequency offset is equal to an integer multiple of subcarrier bandwidth. With the effort to reveal integer CFO, pilot based methods search for a correlating peak over subsequently shifted received pilots. Maximum of all correlations, each one for a given pilot shift, leads to a shift index which is equal to a shift caused by integer CFO and thus the value of Integer CFO. The estimator uses Li and Zhang's algorithm [4].

Note that only continual pilots are used, i.e. an evaluation of intersection with scattered pilots and consequent discarding of scattered pilots is performed.

Matlab function, where continual pilots method (variant 3) is selected:

```
function [icfo, Lam] = rx_integerCFOinFreq(s_conf, input, pilot_indx, ...
                         pilot_value, symbol_number, method, varargin)

  Inputs:
    s_conf        -- configuration structure; see rx_param() function
    input         -- consecutive received data symbols
                     (size of this matrix depends on selected method)
    pilot_indx    -- pilot indices given by rx_pilot_address.m
    pilot_value   -- pilot values given by rx_pilot_address.m
    symbol_number -- position of the 1st symbol in the input sequence
                     (position of a symbol in T2 frame)
    method        -- specific pilots to be used for CFO estimation
                     possible methods are: 'scattered', 'continual' and 'p2'
    version       -- specific version of the selected method
                     posssible versions (default 'V0') are:
                       Continual: V0 ... V4
                       Scattered: V0 ... V3
                       P2:        V0 and V1

  Outputs:
    icfo -- CFO estimate
    Lam  --  metric values for each correlation shift
```

The test bench that tests estimation of integer CFO in frequency domain under different conditions is `testbench_intCFOinFreq.m`. Based on receiver chain, test bench provides: P1 detection coarse and fine, T2-frames extraction, fine time synchronization on P2 and data symbols with fine CFO estimation, P2 and data symbols extraction with correction of estimated fine CFO. For estimation of integer CFO based on symbol pilots, following procedures are implemented: demodulation of P2 and data symbols, decoding of subcarrier order, generation of pilot sets, pilot extraction.

3.2.4 Other Methods Available in Frequency Domain

To provide a wide range of synchronization methods, following pilot-based methods operating in frequency domain were implemented and evaluated in Matlab:

- Time offset on P2 pilots

- Time offset on scattered pilots
- Fine carrier frequency offset on continual pilots
- Fine carrier frequency offset on scattered pilots
- Integer carrier frequency offset on P2 or continual or scattered pilots

For time-offset estimation, the scattered pilots can be used. Matlab function implementing time-offset estimation is `rx_fineTOinFreq(..., 'scattered')`. The test bench that tests the time offset estimation in frequency domain under different conditions is `testbench_TOinFreq.m`.

For fine CFO estimation, the scattered pilots can be used. Matlab function implementing fine CFO estimation is `rx_fineCFOinFreq(..., 'scattered')`. The test bench that tests fine CFO estimation in frequency domain under different conditions is `testbench_fineCFOinFreq.m`.

For integer CFO estimation, both scattered and P2 pilots can be used. Matlab function implementing integer CFO estimation is `rx_integerCFOinFreq(...)` , where the proper method and its version have to be selected. In summary possible methods are: 'scattered', 'continual' or 'p2', where the methods have the following versions: Continual – V0 . . . V4, Scattered – V0 . . . V3, P2 – V0 and V1.

In particular, for the continual pilots:

- **V0** is the simple Li, Zhang's method
- **V1** is the simple Li, Zhang's method with demodulation
- **V2** is the Li, Zhang's method with spectra split in two halves
- **V3** is the Li, Zhang's method with spectra split in two halves and demodulation
- **V4** is the McNair's method

In particular, for the scattered pilots:

- **V0** is the simple Li, Zhang's method
- **V1** is the Li, Zhang's method with timing offset elimination
- **V2** is the Li, Zhang's method with timing offset and channel influence elimination
- **V3** is the McNair's method

In particular, for the P2 pilots:

- **V0** is the simple correlation method
- **V1** is the McNair's method

The test bench that tests estimation of integer CFO in frequency domain for all methods above is `testbench_intCFOinFreq.m`.

Note that some of the implemented methods, except for highlighted ones at the beginning of this section, have shown poor results in experiments and their use is not recommended.

3.3 OFDM Demodulation and Channel Equalization

OFDM demodulation and channel equalization consists of functions for OFDM demodulation using FFT (containing the removal of unused frequency band for normal and extended mode), pilot pattern generation (continuous and scattered), channel estimation in frequency domain for P2 and data symbol and channel interpolation in frequency.

3.3.1 OFDM demodulation

Matlab functions performing OFDM demodulation is employs internal FFT routine called in the main script. Demodulation can be achieved using the following sequence of commands:

```
  % reshape one frame to the matrix format -- each column one OFDM symbol
  Data_t=reshape(T2_frame(:,frame_no),Sconf.Symbol_size,Sconf.No_symbols+1);
  % remove cyclic prefix
  Data_t=Data_t(Sconf.GI_len+1:end,2:end);
  % demodulate using FFT
  Data_decoded=fft(Data_t)*1/Sconf.K_norm/(Sconf.FFT_size);
  % reorder logical vs. phisical carriers -- provides fftshift
  Data_decoded=[Data_decoded(end/2+1:end,:); Data_decoded(1:end/2,:)];
```

3.3.2 Channel equalization

For channel estimation LS method based on actual data measurements is implemented. For estimation, P2 and scattered pilots (and edge pilots) are used. Then, the linear (`rx_chanEst_lin`) or second order (`rx_chanEst`) interpolation in frequency domain can be used. There is no need for interpolation in time domain. The equalization techniques are evaluated in test benches `test_eq3`, `test_eq4` and `test_eq5`.

With respect to input data, the function `chanEst` works in three different modes (parameter `pattern`):

- **P2** is based on pilots contained in the only P2 symbol
- **scattered** is based on several (usually four) consecutive data symbols where scattered pilots form full equidistant grid
- **single** is based on one data symbol; this method seems to be the most robust with less memory requirements

Matlab function:

```
function [chanEst] = chanEst(Sconf, input, pattern, varargin)

  Inputs:
    Sconf         -- configuration structure
    input         -- input signal, OFDM symbol without cyclix prefix
    pattern       -- estimation method: 'P2', 'single' or 'scattered'
    pn_seq        -- PN sequence for generation of scattered pilots
    symbol_number -- symbol index in T2 frame

  Outputs:
    chanEst -- LS estimate of the channel
```

3.4 Frequency De-Interleaver

Frequency de-interleaver reorders data cells in received symbols. Data cell addresses are given by permutation address generator $\mathrm{H(p)}$ described in [1], Section 8.5. The de-interleaver uses this permutation generator to determine reverse (de-interleaving) process. Permutation address generator and the de-interleaver are configurable to all possible FFT sizes and both symbol types (data or P2).

Integration into the receiver chain exploits two functions: de-interleaver function and permutation address generator. It is assumed that all received symbols are synchronized, FFT-demodulated and their subcarrier indices are transformed from physical to logical indices. To de-interleave a symbol, there is a need to know and scatter all non-data subcarriers that carry pilots, PAPR or to extend bandwidth in extended mode.

Process of de-interleaving data cells follows these steps:

1. Enumeration of permutation sequences (even and odd $\mathrm{H(p)}$) using generator—deciding factors here are the type of symbol (P2 or data) and FFT size.
2. Demodulated data or P2 symbol with logical subcarriers ordering is freed of all possible pilots, reserved or bandwidth extending subcarriers.
3. Correct number of remaining cells is verified as this number have to be equal to the length of corresponding permutation sequence.
4. Symbol is processed by the de-interleaver, where the symbol number l determines either even or odd sequence.
5. De-interleaved data cells are ready.

3.4.1 Address Permutation Sequence Generator

Address generator, provided by the Matlab function `rxtx_freqDeinterleaverHpGenerator`, enumerates even and odd permutation sequences $\mathrm{H0(p)}$ and $\mathrm{H1(p)}$. Exact addresses in sequences depend on the de-interleaved symbol type (P2 or data). Length of $\mathrm{H(p)}$ sequences also depends on DVB-T2 system setup, namely on the number of available data cells `Cdata`. In case of the mode with 32k size FFT, there is only one $\mathrm{H(p)}$ sequence, i.e. $\mathrm{H1} = \mathrm{H0}$, and the differences between even and odd symbol interleaving are solved further at the de-interleaver.

Matlab function:

```
function [H0 H1] = rxtx_freqDeinterleaverHpGenerator(s_conf, symbol_number)

  Inputs:
    s_conf        -- configuration structure
    symbol_number -- position of symbol in T2-frame, l=0...NP2+Ldata-1

  Outputs:
    H0 -- permutaion sequence H(p), even symbols
    H1 -- permutaion sequence H(p), odd symbols
```

3.4.2 De-Interleaver Function

De-interleaver, provided by the Matlab function `rx_freqDeinterleaver`, takes the permutation sequences $H(p)$ for even and odd symbol and transforms indices of received data cells to de-interleaved indices according to formula specified in the DVB-T2 standard.

Matlab function:

```
function deinter_data = rx_freqDeinterleaver(s_conf, H0, H1, ...
                          symbol_number, tos, data)

  Inputs:
    s_conf        -- configuration structure
    H0            -- permutaion sequence H(p), even symbols
    H1            -- permutaion sequence H(p), odd symbols
    symbol_number -- position of symbol in T2-frame, l=0...NP2+Ldata-1
    tos           -- type of symbol: 'P2' or 'data'
    data          -- interleaved data cells

  Outputs:
    deinter_data -- frequency-deinterleaved data cells
```

4 Selected test benches

test_p1_coarse_SNR4 Evaluation of P1 detection and decoding for flat SISO channel and a range of SNRs. The P1 is detected using the Phase Shift property method (see Section 3.1) and decoded. As input, the grabbed input data in BBC mode are used. The data are shortened to contain seven P1 symbols, but only part of T2-frame. Subsequently, a non-constant frequency offset is applied on input data. **Note:** The grabbed data contains already a minor frequency offset.

The following properties are evaluated:

- P1 detection (coarse time synchronization)
- P1 decoding, estimation of L1-pre parameters (S1, S2)
- estimation of coarse (P1-based) frequency offset

S1 and S2 estimation is evaluated using the “Probability of correct detection”, where '1' means no error and '0' means all estimates are false.

The test set-up parameters:

- SNR – range of SNR to evaluate, for example `SNR=10:2:20;`
- F_offset – frequency offset to insert into the input signal
- beta – a random walk parameter for the evaluation of frequency offset

test_freq_sync_SNR1 Simulates time and frequency synchronization. The synchronization is performed in several steps:

1. First frame:
   (a) P1 is detected.

   (b) Fractional frequency offset is estimated and corrected.
   (c) P1 is decoded, integer frequency offset is estimated.
   (d) Frame parameters S1 and S2 are estimated.
2. Frequency offset correction set-up for the second frame.
3. Second frame:
   (a) P1 is detected etc. as in the first frame.
   (b) Guard interval is estimated on data symbols.
4. Frequency offset correction for the following frame.
5. Repeat.

To lower the memory requirements, the calculation is based on batches of T2-frame size. For that reason, adaptive threshold calculation is reset ones per frame, which may cause that the first P1 symbol in the batch is not detected and the script ends with '`index out of bound`' error. This can occur namely for low SNRs (less than 0).

The test set-up parameters:

- SNR – range of SNR to evaluate, for example `SNR=10:2:20;`
- F_offset – frequency offset to insert into the input signal
- beta – a random walk parameter for the evaluation of frequency offset

test_eq5 Evaluation of equalization and channel estimation methods. To shorten the computation time, the test uses pre-computed BBC data where positions of P1 symbol is stored in the `.mat` file. The grabbed data with SNR=50dB and flat fading channel are used.

The P2 and data symbols are demodulated and LS channel estimation and interpolation in frequency is performed. Finally, original QAM symbols are plotted. Estimation and interpolation is done using a single data symbol. See the test `test_eq3` for the channel estimation based on all pilots.

testbench_freq_deinterleaver Complete test of the frequency de-interleaver defined in the DVB-T2 standard, Section 8.5. The following functions are tested in this test bench:

- `rxtx_freqDeinterleaverHpGenerator` – see Section 3.4.1 for parameters
- `rx_freqDeinterleaver` – see Section 3.4.2 for parameters

Based on receiver chain, the test bench provides: P1 detection coarse and fine, T2-frames extraction, fine time synchronization on P2 and data symbols with fine CFO estimation, P2 and data symbol extraction with correction of estimated fine CFO. The following procedures are implemented: demodulation of P2 and data symbols, decoding of subcarrier order, generation of pilot sets, pilot extraction.

References

[1] European Telecommunications Standards Institute, 650 Route des Lucioles, F-06921 Sophia Antipolis Cedex, France. *ETSI EN 302 755 V1.2.0b: Digital Video Broadcasting (DVB); Frame structure channel coding and modulation for a second generation digital terrestrial television broadcasting system (DVB-T2)*, draft edition, July 2008. Reference DEN/JTC-DVB-228.

[2] European Telecommunications Standards Institute, 650 Route des Lucioles, F-06921 Sophia Antipolis Cedex, France. *ETSI TR 102 831 V0.9.6: Digital Video Broadcasting (DVB); Implementation guidelines for a second generation digital terrestrial television broadcasting system (DVB-T2)*, draft edition, January 2009. Reference <none>.

[3] Bruno Jahan, Marc Lanoiselee, Gabriel Degoulet, and Rodrigue Rabineau. Full synchronization method for OFDM/OQAM and OFDM/QAM modulations. In *Spread Spectrum Techniques and Applications, 2008. ISSSTA '08. IEEE 10th International Symposium on*, pages 344–348, Aug. 2008.

[4] Mingqi Li and Wenjun Zhang. A novel method of carrier frequency offset estimation for ofdm systems. *Consumer Electronics, IEEE Transactions on*, 49(4):965–972, Nov. 2003.

[5] Xianbin Wang, Yiyan Wu, J.-Y. Chouinard, Sili Lu, and B. Caron. A channel characterization technique using frequency domain pilot time domain correlation method for DVB-T systems. *Consumer Electronics, IEEE Transactions on*, 49(4):949–957, Nov. 2003.

[6] Hanli Zou, B. McNair, and B. Daneshrad. An integrated OFDM receiver for high-speed mobile data communications. volume 5, pages 3090–3094 vol.5, 2001.

A DVD-ROM Content

All Matlab scripts and functions are located in the `src` directory. The following list of files provides a brief overview of functions that can be found on the DVD-ROM.

A.1 DVB-T2 functions

```
rx_GI_detect.p
rx_P1_decode.p
rx_P1_detect.p
rx_P1_detect2.p
rx_P1_sync.p
rx_chanEst.p
rx_chanEst_lin.p
rx_coarseEst_GI.p
rx_cp_groups.p
rx_fineCFOinFreq.p
rx_fineTOinFreq.p
rx_fine_time.p
rx_freqDeinterleaver.p
rx_integerCFOinFreq.p
rx_pilot_address.p
rx_param.p
rx_pn.p
rx_prbs.p
tx_P1_gen.p
rxtx_freqDeinterleaverHpGenerator.p
tx_freqDeinterleaver_dataGenerator.p
freq_deinterleaver_Hp_generator.p
reserved_carriers_helper.p
continual_pilots_helper.p
```

A.2 Auxliliary functions

```
channel.p
channel_siso.p
edgedetect.p
expavg.p
expavgfix.p
find_pieck_max.p
load_points.m
skip_glitches.p
treshold_estimate.p
treshold_estimate_init.p
wgn_octave.m
awgn_octave.m
```

A.3 Test benches

```
test_eq3.m
test_eq4.m
test_eq5.m
test_p1_v2.m
test_P1_coars_SNR4.m
test_freq_sync_SNR1.m
testbench_TOinFreq.m
testbench_fineCFOinFreq.m
testbench_freq_deinterleaver.m
testbench_intCFOinFreq.m
```