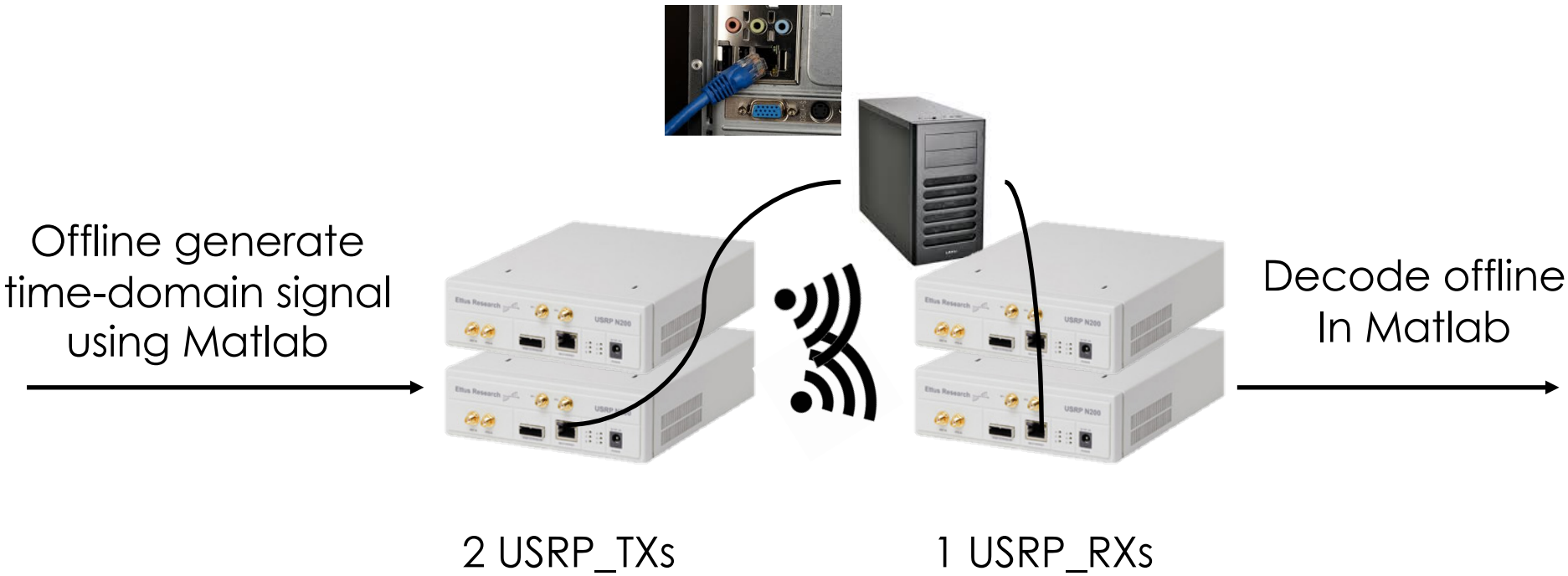
Wireless Communication Systems
@CS.NCTU

Lab4: Interference Nulling on USRP

Outline

- Intro
  - Environment
- Tasks
  - Channel estimation
  - Precoding
  - Decoding (MATLAB)
- Grading Criteria

Environment



- USRP Testbed in LAB / office
- Access through ssh (nulling.cpp/nulling.h)
- Run Matlab in your own machine

Build MIMO Using USRPs

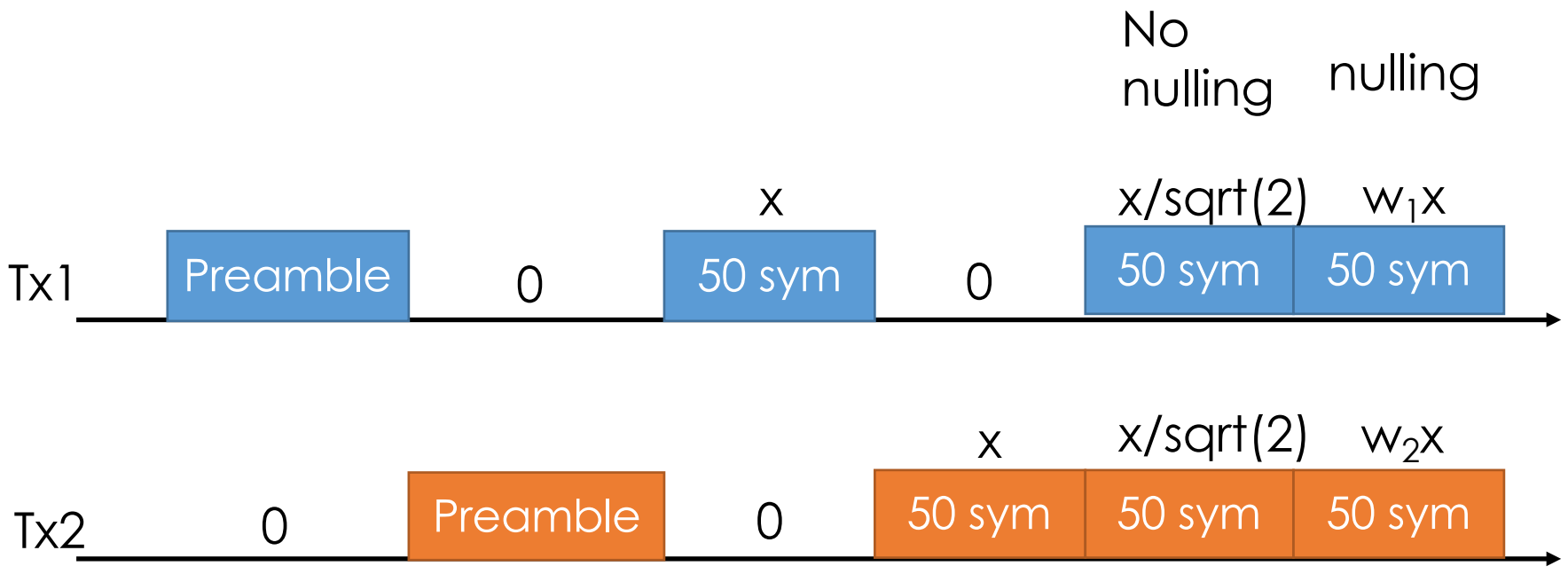
- Connect two USRPs using an external clock

- Configure clock info (have been implemented in the example file)
  - Usrp_tx->set_clock_config(uhd::clock_config_t::external());
    usrp_tx->set_time_next_pps(uhd::time_spec_t(0.0));

USRP Server

- ssh wcs-g#@140.113.203.6
  - e.g., wcs-g1@140.113.203.6

Packet Format



Flow

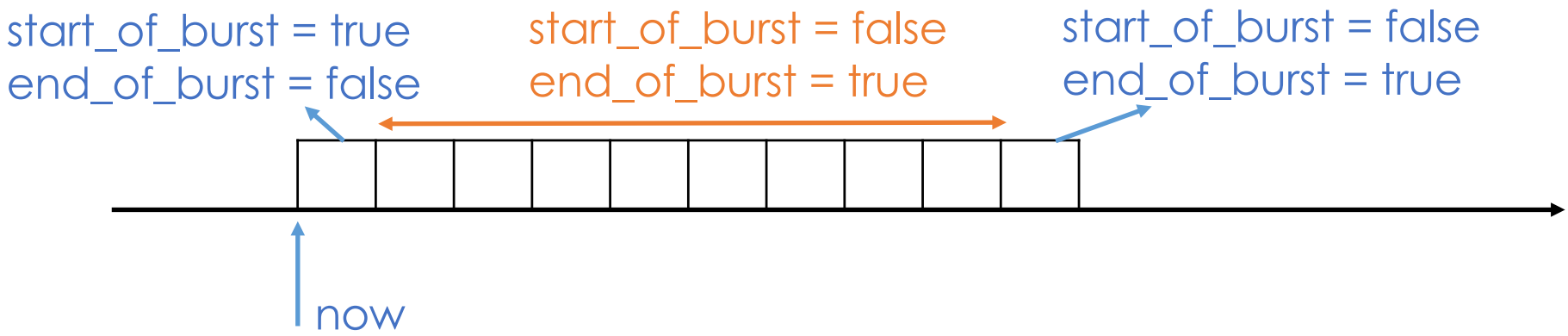
1. init_sys() `// read freq. samples`
2. init_usrp() `// configure usrp`
3. sync_clock() `// sync to external clock`
   - `usrp->set_clock_config(uhd::clock_config_t::external());`
   - `usrp->set_time_next_pps(uhd::time_spec_t(0.0));`
4. init_stream() `// setup rx time`
5. Send zero samples (line 246-242)
6. Rx garbage sample (line 264-272)
7. Configure tx/Rx time (line 282-291)
8. Tx/Rx

Metadata

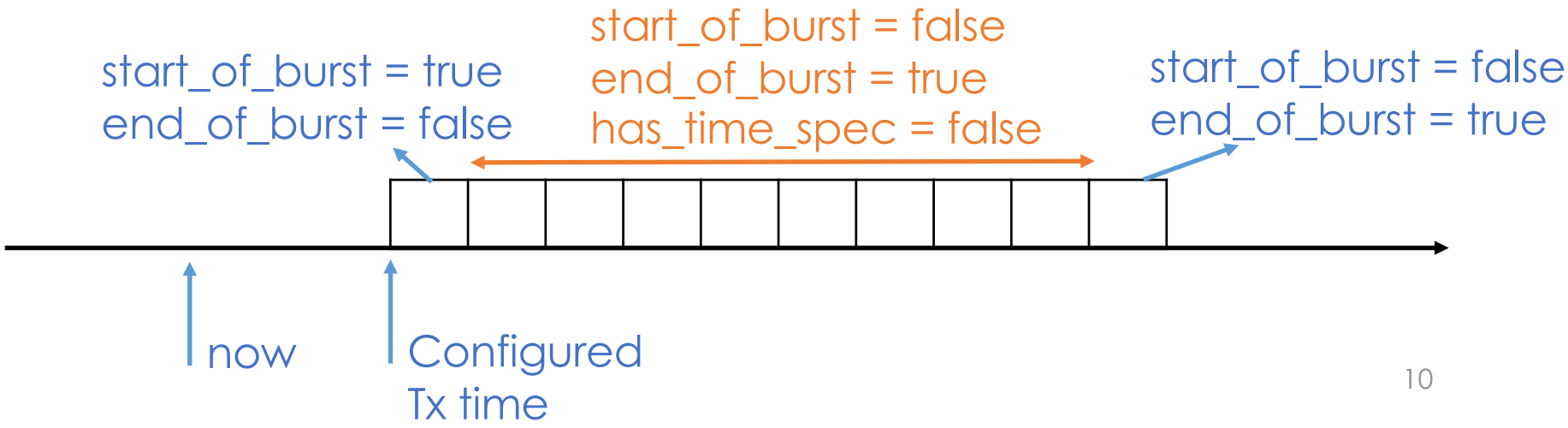
- tx_metadata, rx_metadata
- metadata.time_spec: info. about time
  - tx_metadata.time_spec: scheduled tx time
  - time_spec.get_real_secs(): convert to real seconds
  - Time_spec.get_tick_count(rate): convert to # tick
- metadata.start_of_burst
  - Binary indicating whether this is the first sample
- metadata.end_of_burst
  - Binary indicating whether this is the last sample
- metadata.has_time_spec
  - true: send at a particular time (specified in metadata)
  - false: send immediately

Metadata

- has_time_spec = false



- has_time_spec = true

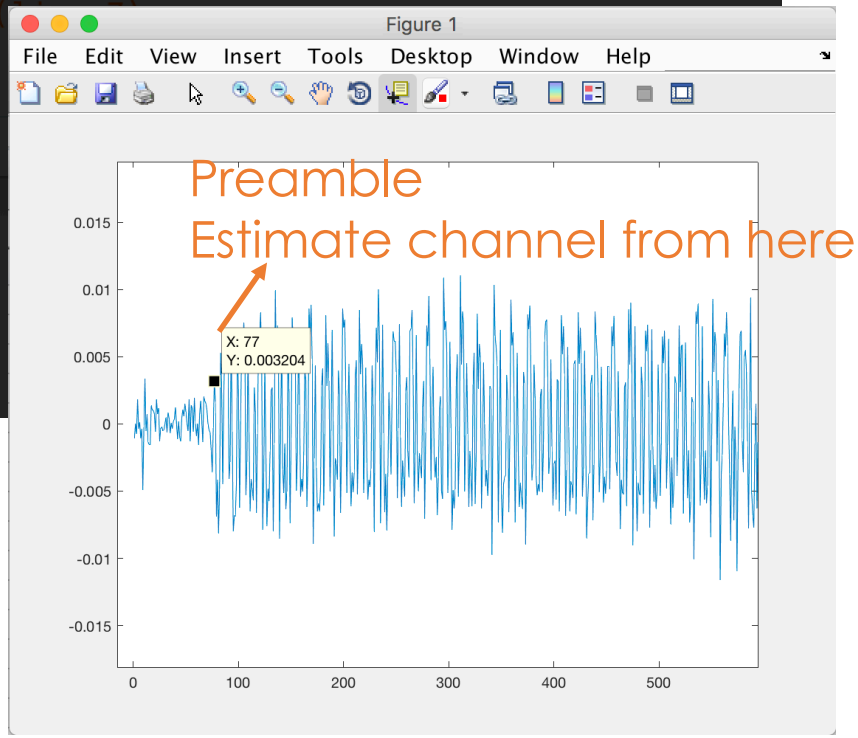


What in the Example Code?

- Read freq.-domain signals from the input file
- ifft function
- While
  - Tx1 sends preamble (time-domain)
  - Tx2 sends preamble (time-domain)
  - TODO here: precoding at freq.-domain
  - Do ifft for data symbols
  - Tx1 and Tx2 send data symbols together
  - Rx signals
  - Write signals if the received time is larger than the configured tx time
    - TODO here: channel estimation if the rx signals are the preamble

Results of the Example Code

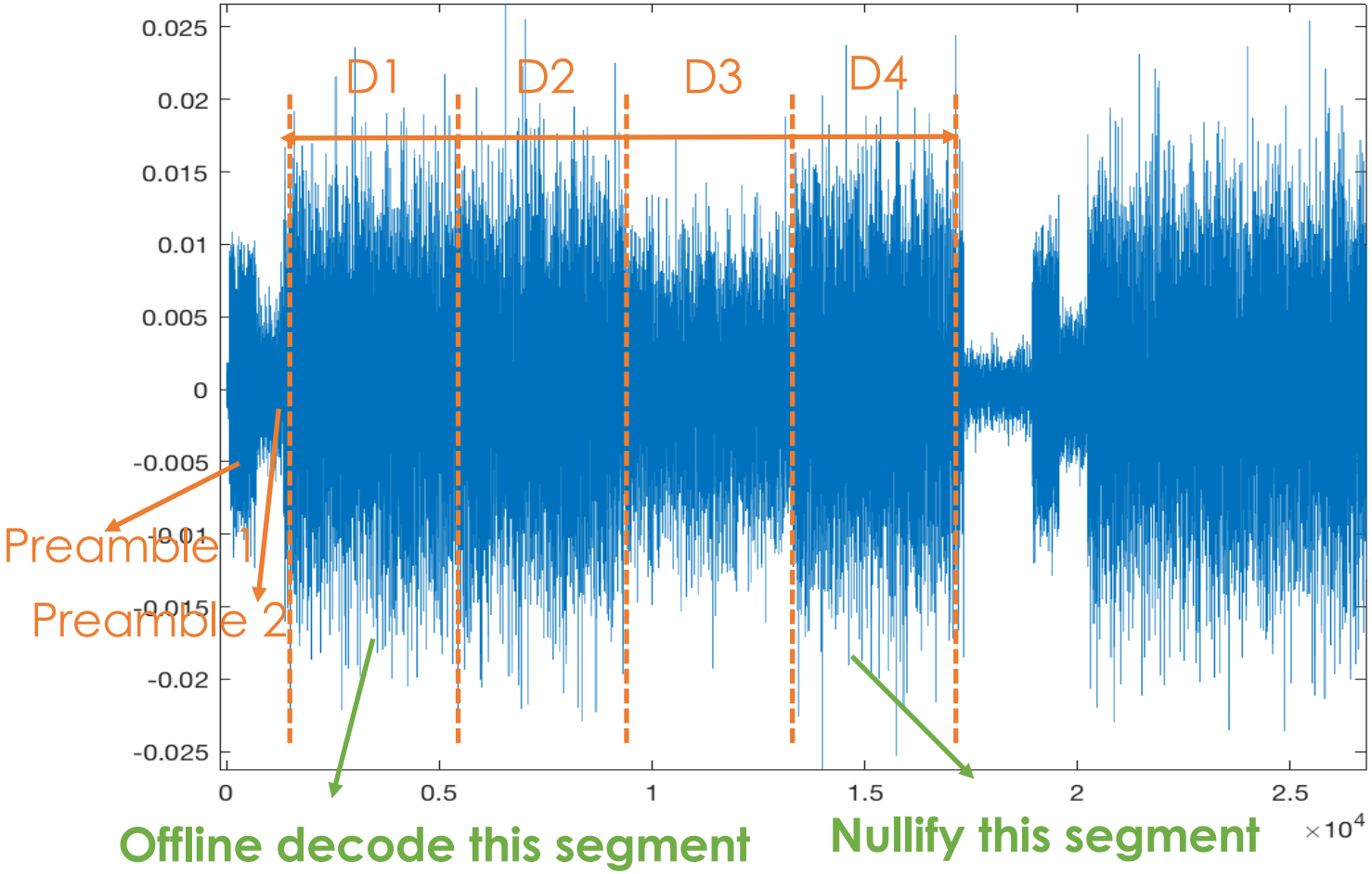
```
UHD Warning:
    Unable to set the thread priority. Performance may be negatively affected.
    Please see the general application notes in the manual for instructions.
    EnvironmentError: OSError: error in pthread_setschedparam
SYNC Clock
Time to start receiving: 1.58224
Press Ctrl + C to stop streaming...
UCurrent clock time: 2.58304
Configured Tx time: 3.5832
Configured Tx tick: 455628
start recving tick: 455578
start recving 80 samples at 3.58314
finish recving 250000 samples at 3.90304

Dump signals
Delete receiving signal buffer
^C^C^C^C^C^CTerminate systems ...
```



In this example, you will receive (455628 – 455578) = 50 samples of noise. However, in reality, the offset may be larger, e.g., 77 in this case.

Results of the Example Code

Tx1	x	0	x/sqrt(2)	x
Tx2	0	x	x/sqrt(2)	x



Preliminary – FFT library Installation

- Library : FFTW3
  - http://www.fftw.org/
- TODO: Add linking library to Cmake file (important)
  - Edit /uhd/host/example/CMakeLists.txt
  - Add fftw3 and m to link library (around line 52)

```
FOREACH(example_source ${example_sources})
    GET_FILENAME_COMPONENT(example_name ${example_source} NAME_WE)
    ADD_EXECUTABLE(${example_name} ${example_source})
    TARGET_LINK_LIBRARIES(${example_name} uhd ${Boost_LIBRARIES})
    …
    TARGET_LINK_LIBRARIES(${example_name} uhd fftw3 m ${Boost_LIBRARIES})
ENDFOREACH(example_source)
```

  - Clean all things in build directory, and cmake again according to the lab1 slide (page 12)

Debug

- If you want to check
  - Whether iFFT is correct
  - The received raw signals

- Modify in nulling.h
  - const size_t DEBUG = 1;

Tasks

1. Channel Estimation
2. Precoding
3. Offline Decoding (Matlab)

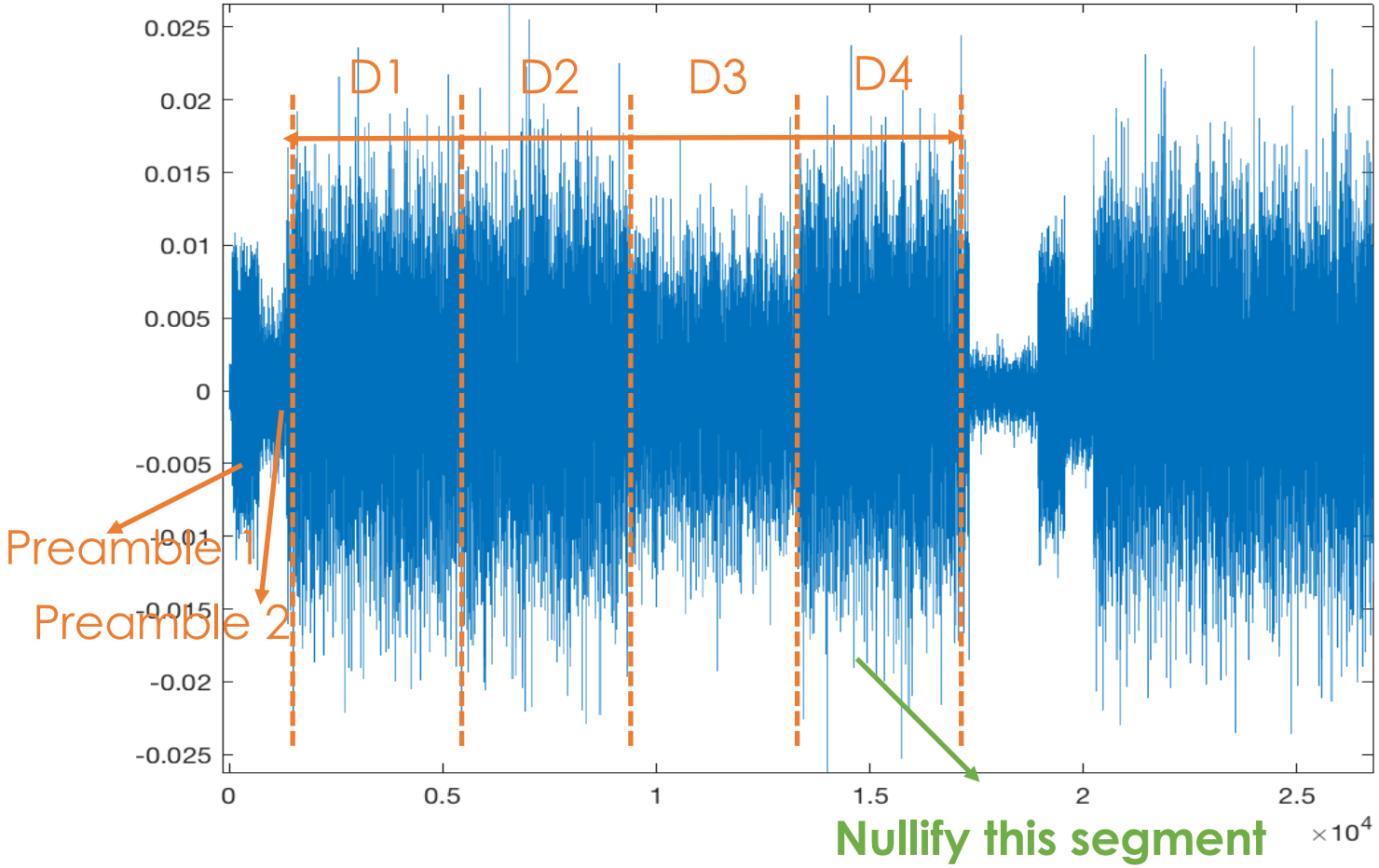
Task 1: Channel Estimation

- Calculate the number of the received noise
- Estimate the index of the first preamble sample
  - With consideration of the clock offset
- Perform channel estimation
  - Convert the received signals to freq.-domain
  - Create a fft() function
    - the same with ifft(), just modify a parameter to "FfT_W_FORWARD"
    - p = fftw_plan_dft_1d(SC_LEN, in, out, **FFTW_FORWARD**, FFTW_ESTIMATE);
  - H[0][k] = Y[0][k] / preamble[0][k] `// for tx1`
  - H[1][k] = Y[1][k] / preamble[1][k] `// for tx2`

Task 2: Precoding

Tx1	x	0	x/sqrt(2)	**w1**x
Tx2	0	x	x/sqrt(2)	**w2**x

Nulling for the 4-th segment



Task 2: Precoding

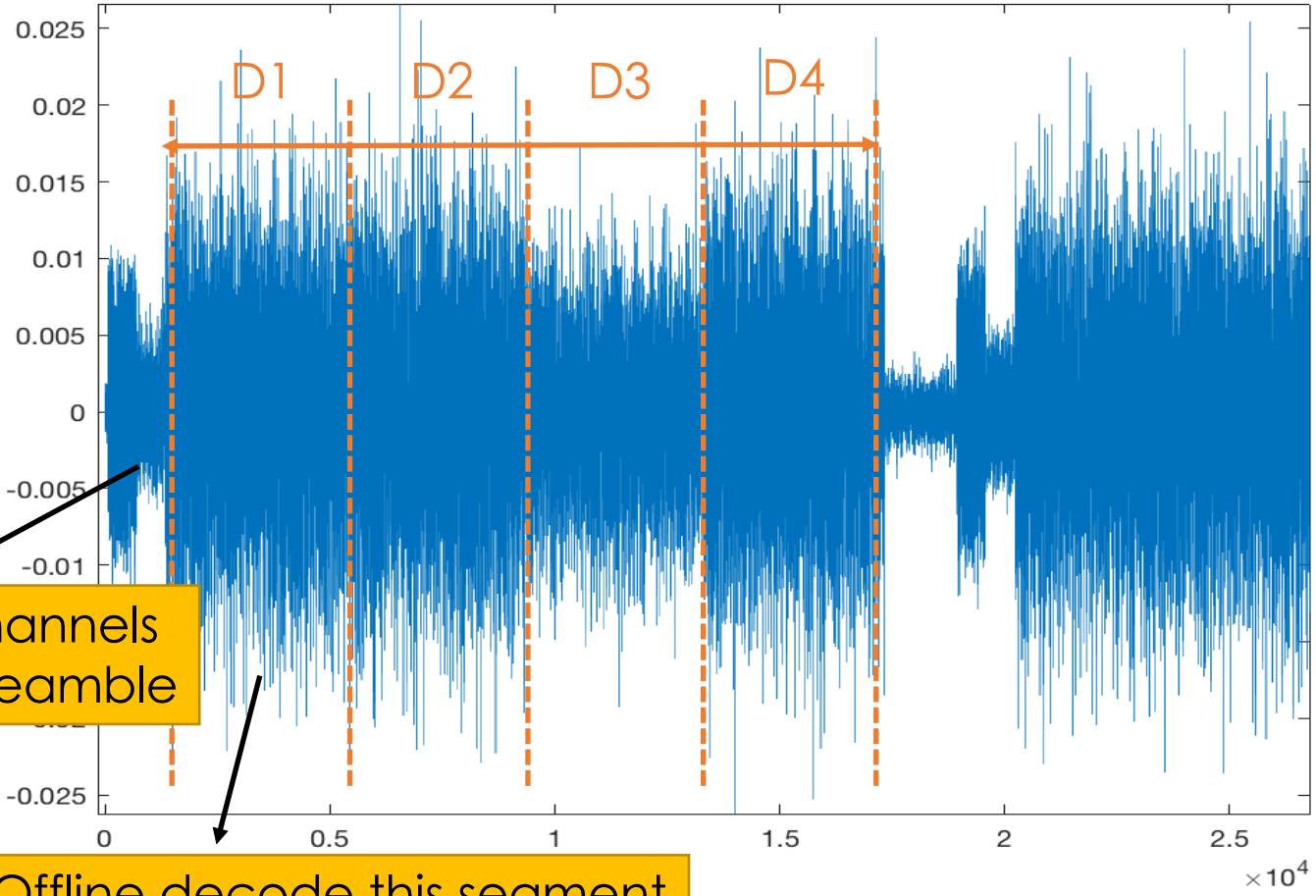
- Calculate precoder w1 and w2 based on the estimated channels H[0] and H[1]
- Normalized the precoder to a unit power
  - $W1[k]^2 + W2[k]^2 = 1$
- Perform precoding
  - Before ifft (freq.-domain),
    - Tx1: pkt_tx_ifft[k] = pkt_tx[k] * w1[k];
    - Tx2: pkt_tx_ifft[k] = pkt_tx[k] * w2[k];
  - ifft(pkt_tx_ifft) `// input the precoded samples to ifft`
  - ifft() will put the results in `pkt_tx_time` (global variable)
    - Send `pkt_tx_time` directly

Task 3: Offline Decode

- Estimate the channels
- Only need to decode the first segment
- Channel estimation in Matlab
- Calculate the average SNR of the first data segment

Task 3: Offline Decode

Tx1	x	0	x/sqrt(2)	w1x
Tx2	0	x	x/sqrt(2)	w2x

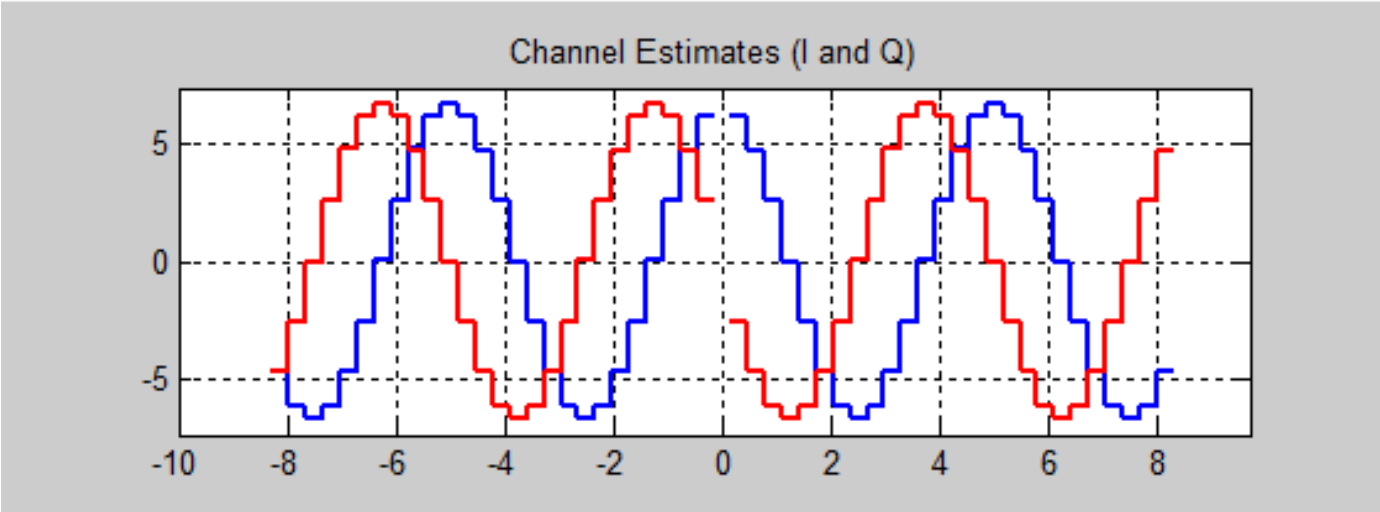


Estimate channels from the preamble

Offline decode this segment

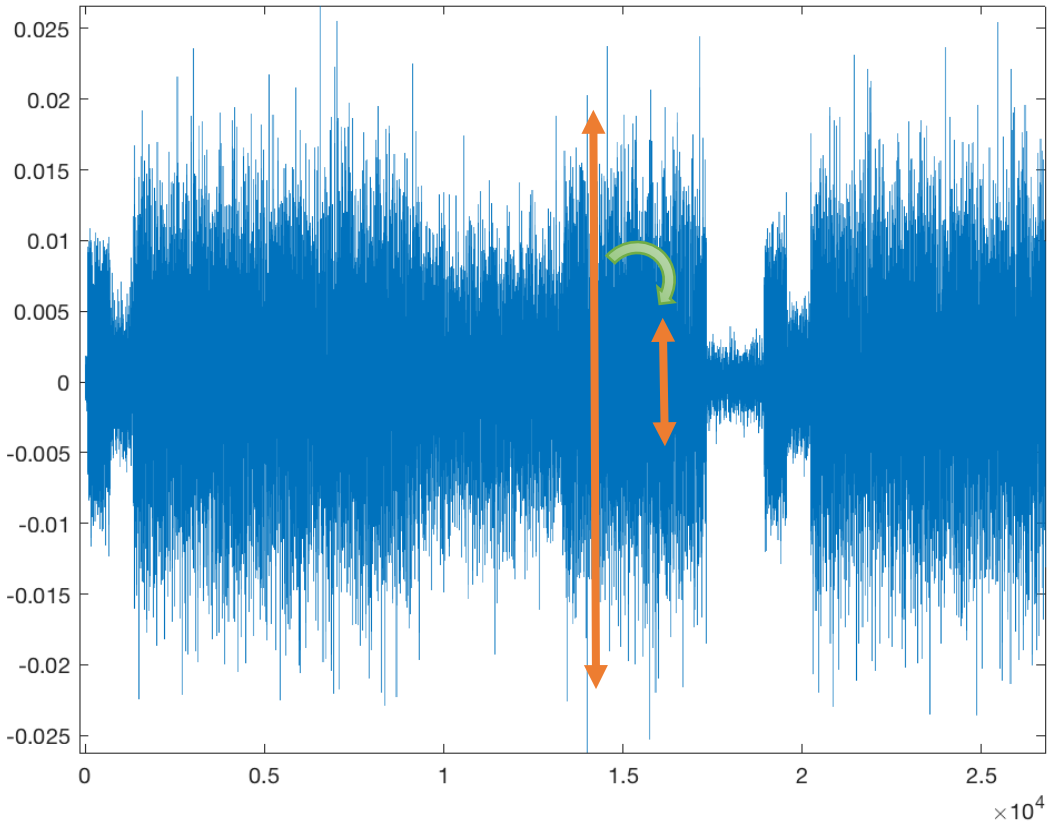
Report

Figure: Real part of the estimated channel from both USRP and Matlab



Report

Figure: Raw received signal



Result: decoding SNR

Grading

- Channel estimation: 25%
- Precoding: 25%
- Offline decoding: 20%
- Report: 30%