MCC125 Wireless Link Project 2021- Software design

In the Tx

- ✓ Data to symbols
- ✓ Construct the frame
- ✓ Up-sampling
- ✓ Filtering
- ✓ To the DAC (the USRP)
- ✓ Up-conversion (0-50 MHz)
- ✓ To the Transmitter HW

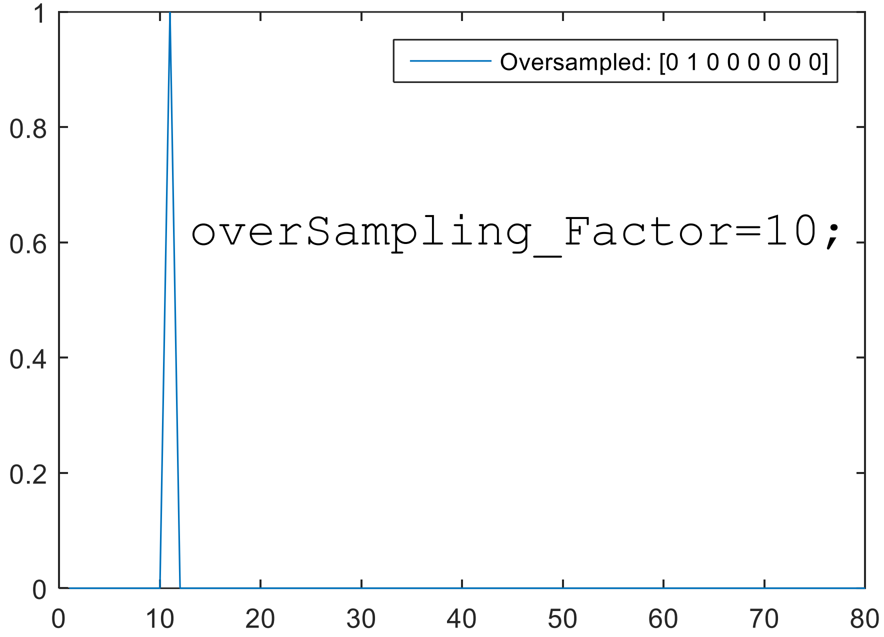
In the Rx

- ✓ Samples received from the USRP
- ✓ Down-conversion to baseband (perhaps together with coarse frequency correction?)
- ✓ Detection of message
- ✓ Filtering
- ✓ Down sampling + timing synchronization
- ✓ Frequency and phase correction
- ✓ Symbols to data
- ✓ Display of the message

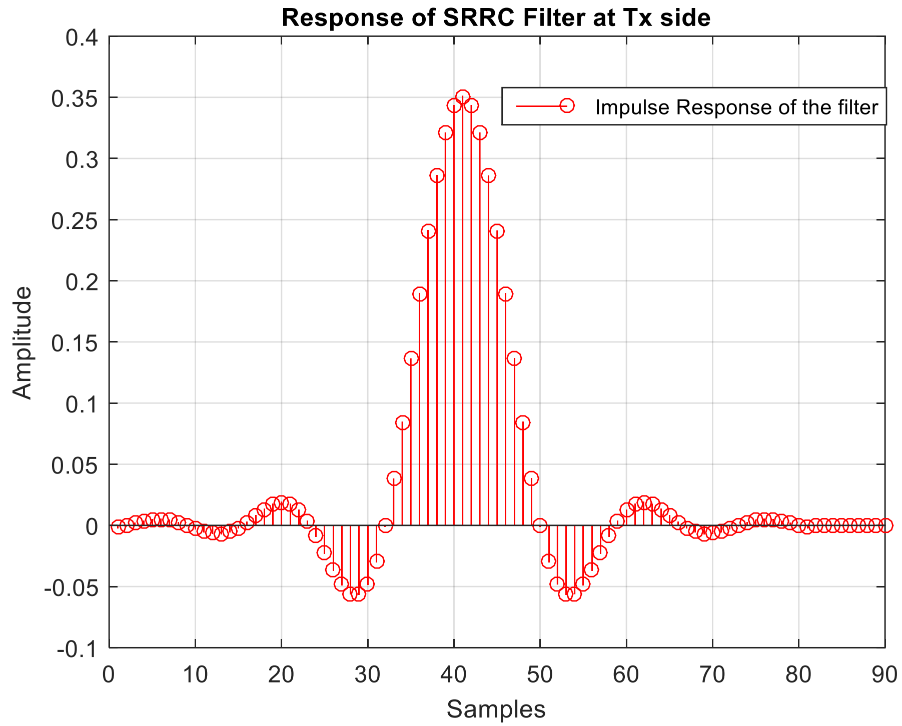
The challenge:
the LO sources in your Tx/Rx hardware
are not synchronized!

The steps above need not necessarily be performed in this sequence.

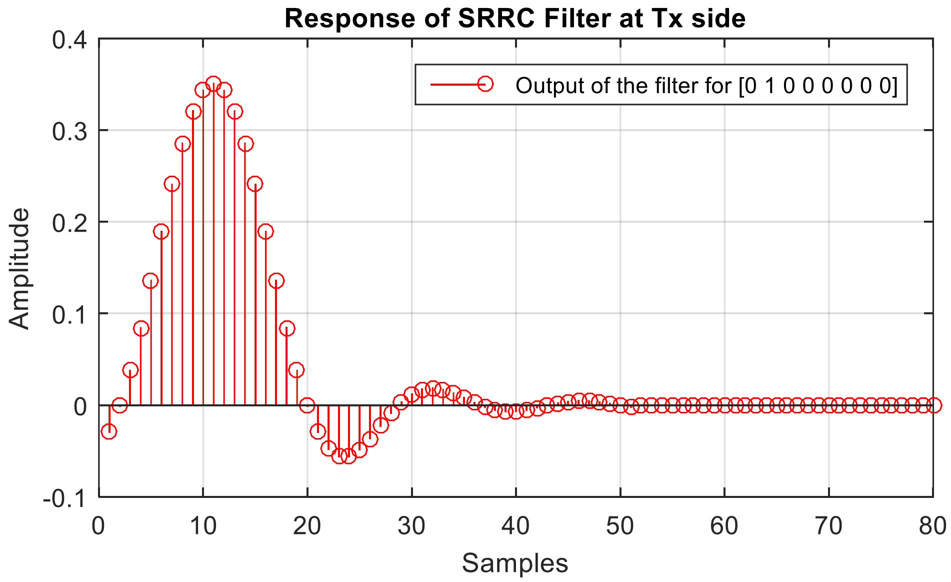
Filtering



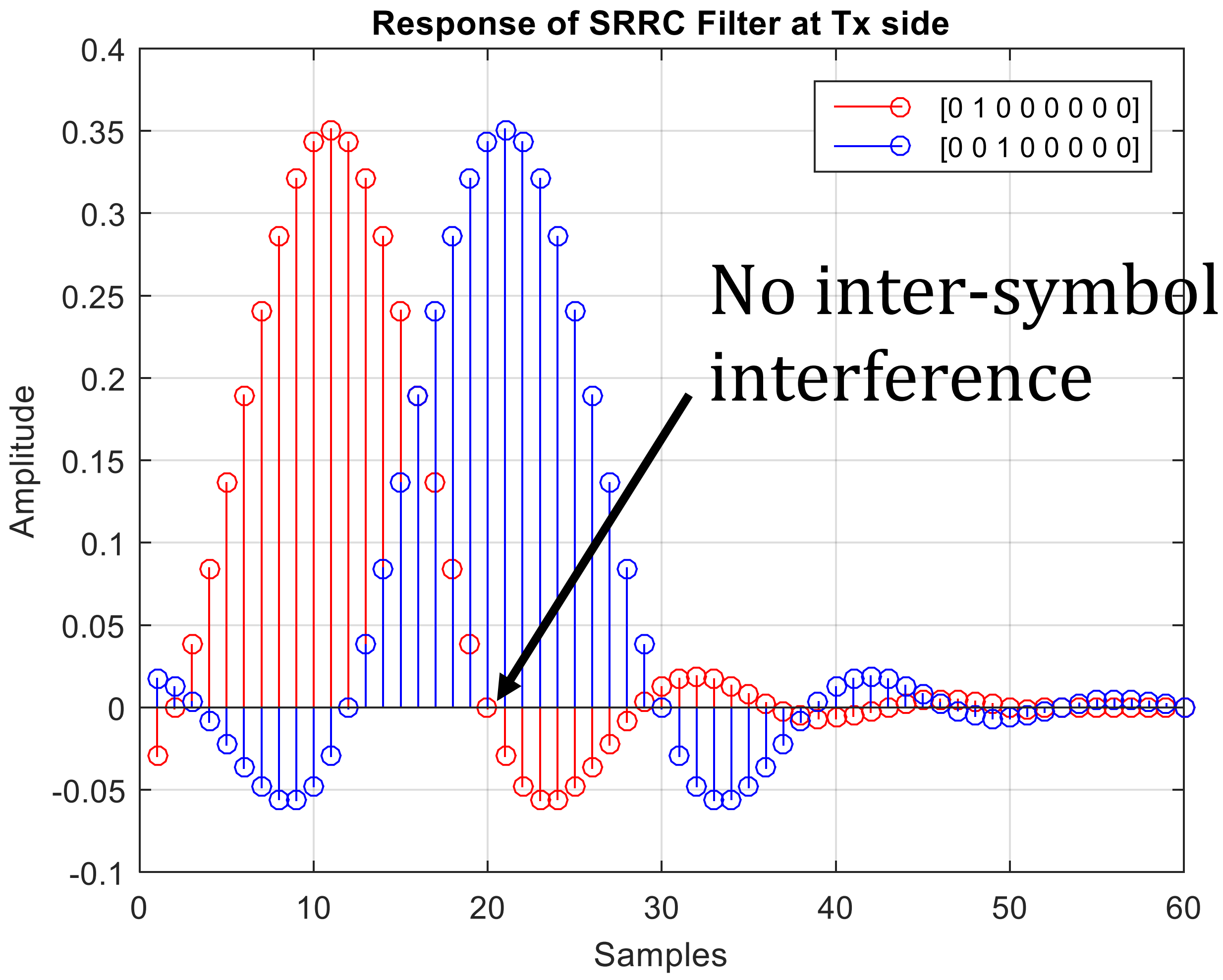

```
overSampling_Factor=10;
```

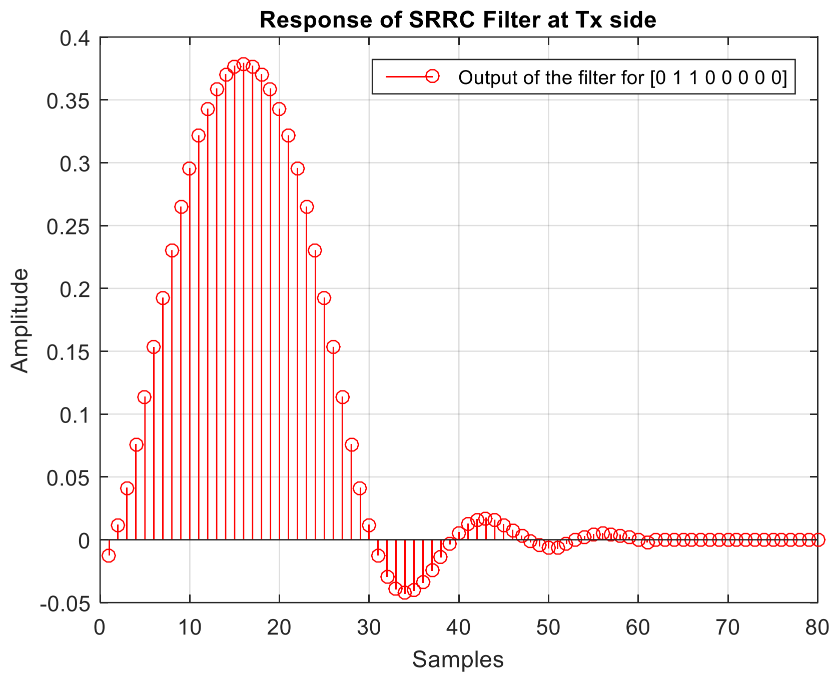


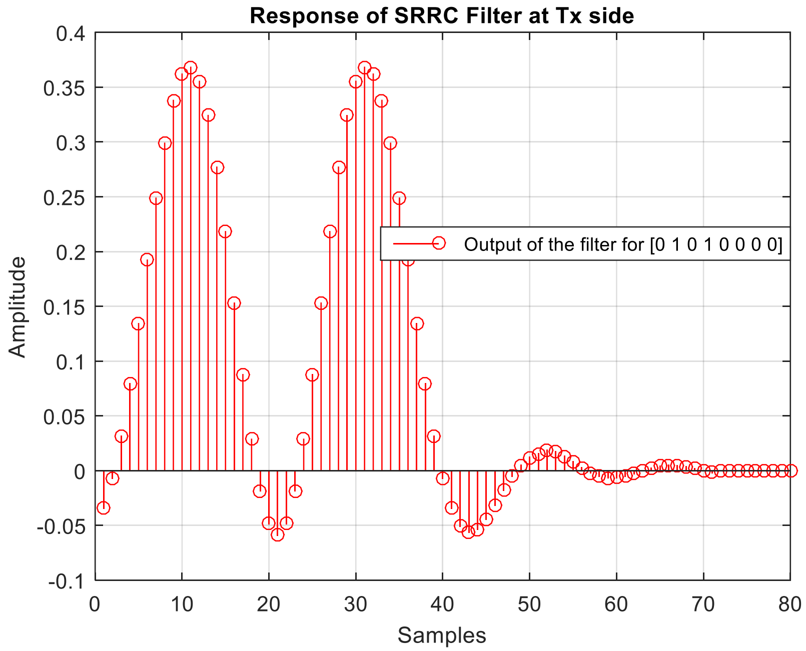

```
pt = rcosdesign(alpha,symb_span,s_per_symb);
output_of_srrc_filter = conv(Input_bit_os,pt,'same');
```

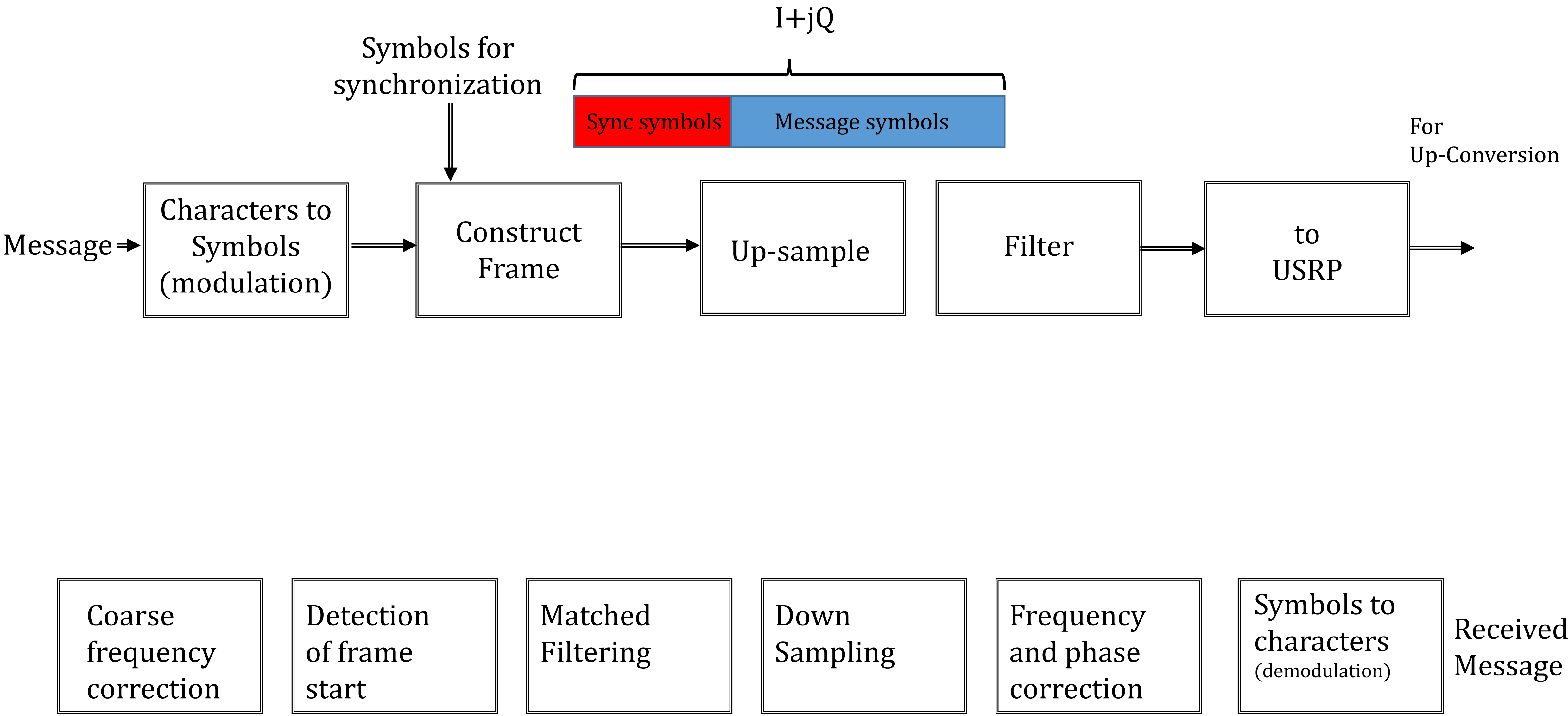


Filtering


Response of SRRC Filter at Tx side
[0 1 0 0 0 0 0 0]
[0 0 1 0 0 0 0 0]
No inter-symbol interference
Amplitude
Samples
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-0.05
-0.1
0
10
20
30
40
50
60


Response of SRRC Filter at Tx side
Output of the filter for [0 1 1 0 0 0 0 0]
Amplitude
Samples

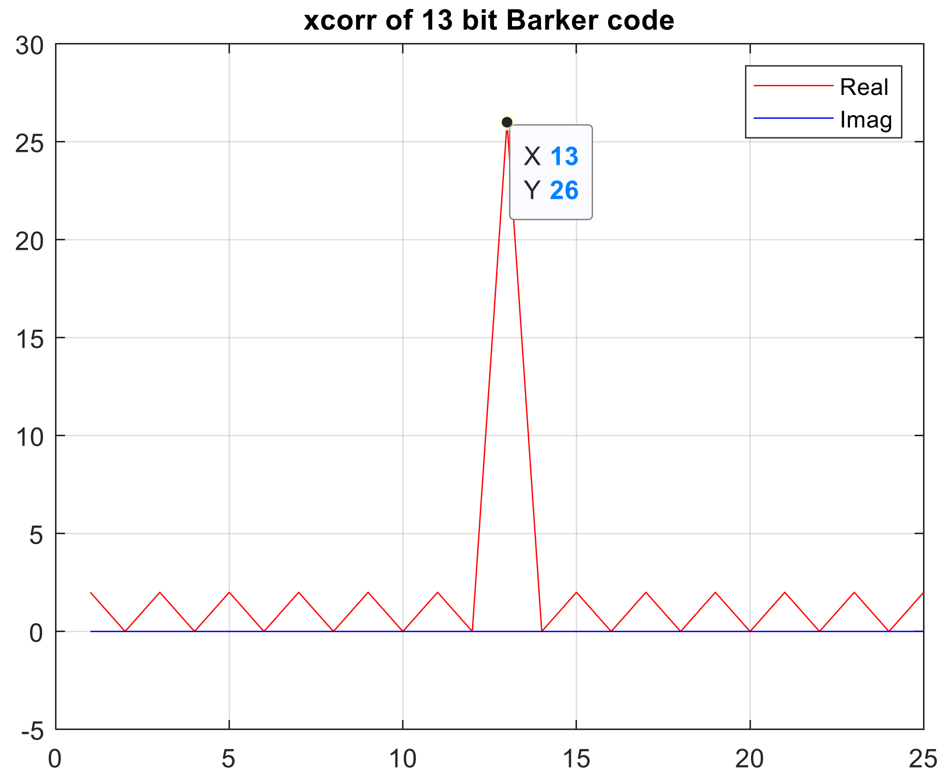

Response of SRRC Filter at Tx side
Output of the filter for [0 1 0 1 0 0 0 0]
Amplitude
Samples


I+jQ
Symbols for synchronization
Sync symbols
Message symbols
For Up-Conversion
Message
Characters to Symbols (modulation)
Construct Frame
Up-sample
Filter
to USRP
Coarse frequency correction
Detection of frame start
Matched Filtering
Down Sampling
Frequency and phase correction
Symbols to characters (demodulation)
Received Message

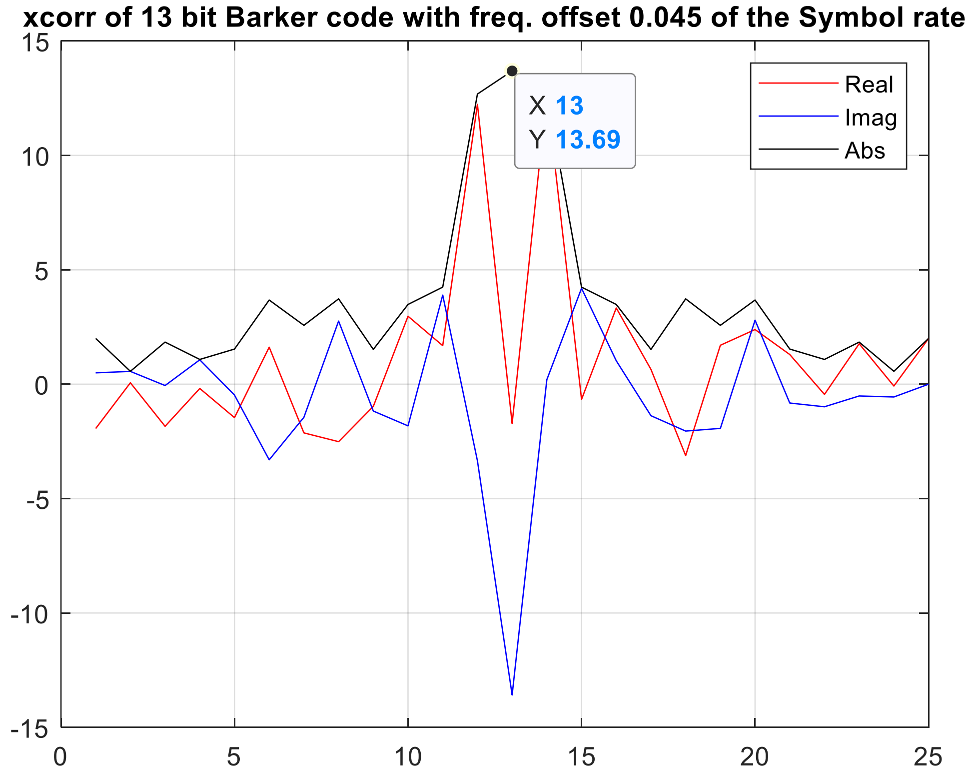
Detection of message/frame start

✓ Your receiver will collect a certain number of samples and it will look for the known "Sync" symbols by correlating your received symbols with the known synchronization symbols.

```
correlation=xcorr((symb_synk),(symb_received));
```

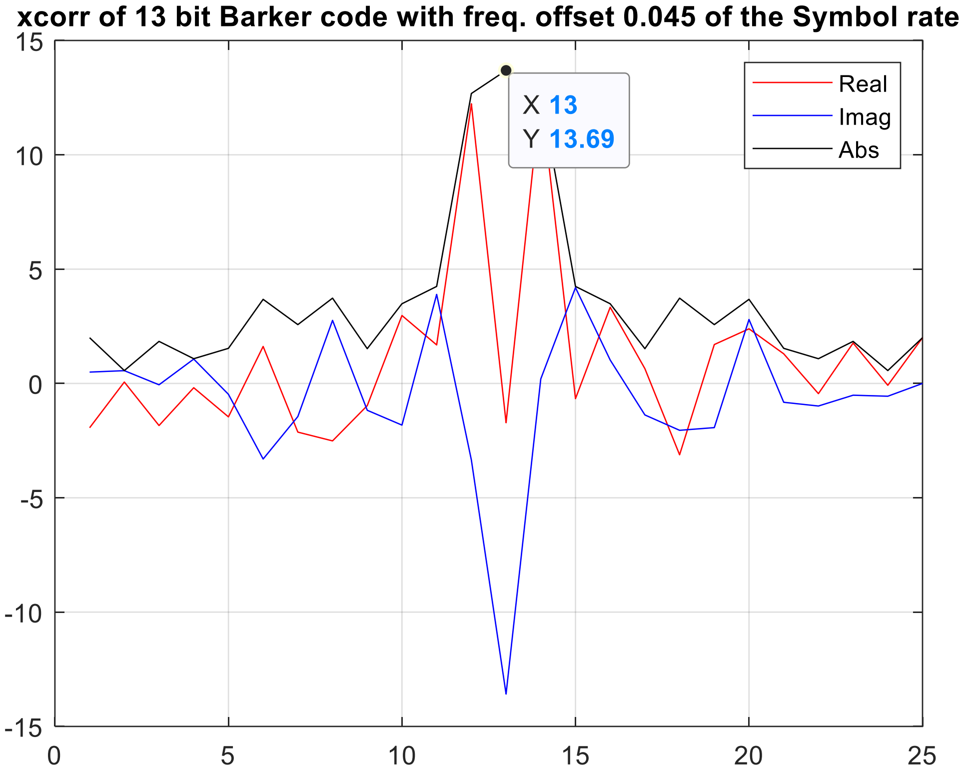


What happens if the received symbols have frequency offset?



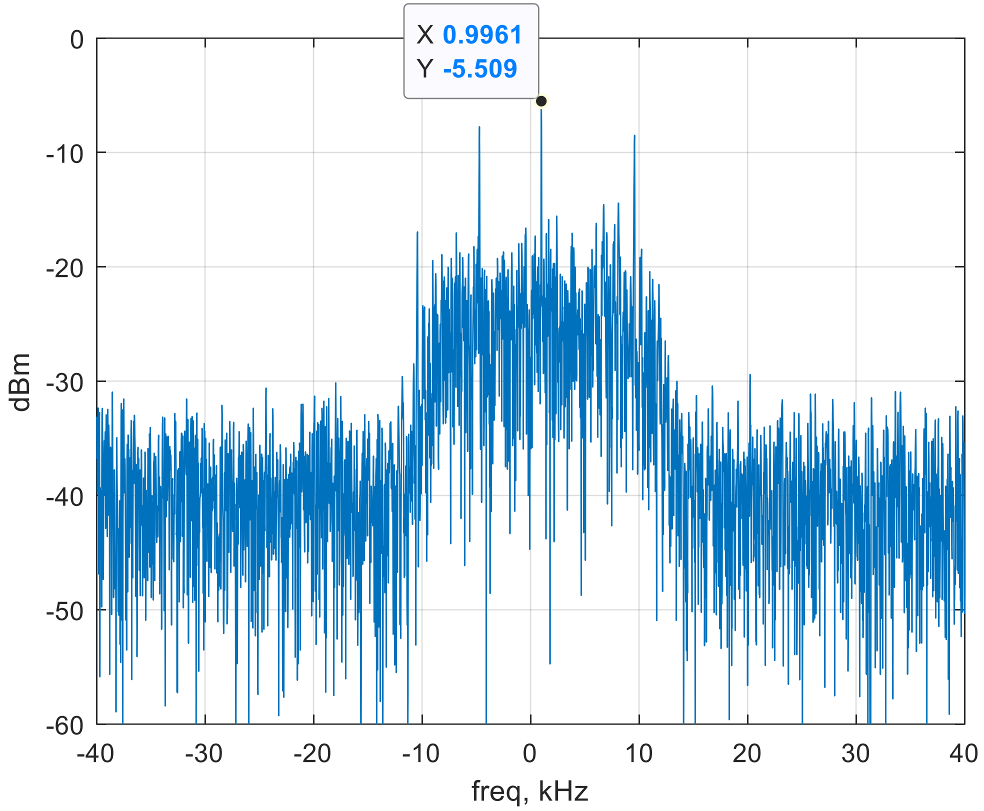
Detection of message, frame start

Using 13 bit Barker sequence, we can tolerate frequency offset up to 0.045 of the symbol rate! Why 0.045?
What can we do to tolerate higher frequency offset and still be able to detect the message?



C:\Users\em7vevas\Box\MCC125_WLP\HT21\Matlab\ Find_Message_Start

Coarse frequency correction

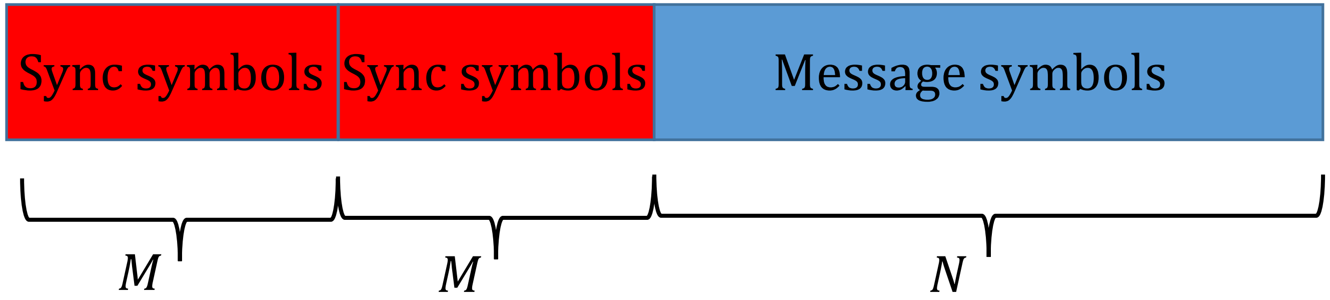


- ✓ It is a good idea to do a course frequency correction as a first step in the receiver.
- ✓ Due to DC offset in the DAC, the received spectrum will have a DC component, which will be located in the middle of the pass-band.
- ✓ Identifying the frequency of the DC component allows to correct for frequency offsets. This will make the detection of the message easier.
- ✓ If the DC component is not highest in the spectrum, one can enhance it by adding DC offset in the data.

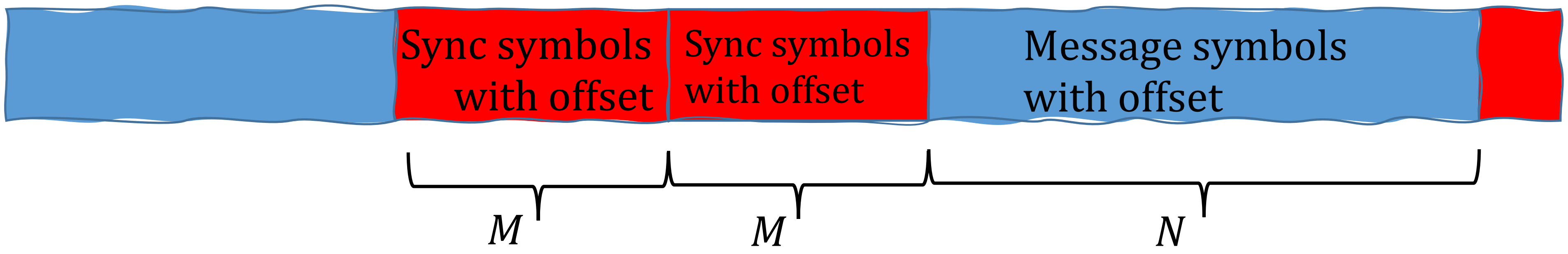
C:\Users\em7vevas\Box\MCC125_WLP\HT21\Matlab\TxRx11.m

Detecting the message in presence of arbitrary frequency offset (1)

Step 1: In the Tx you construct your frame so that the sync symbols are repeated at least 2 times.

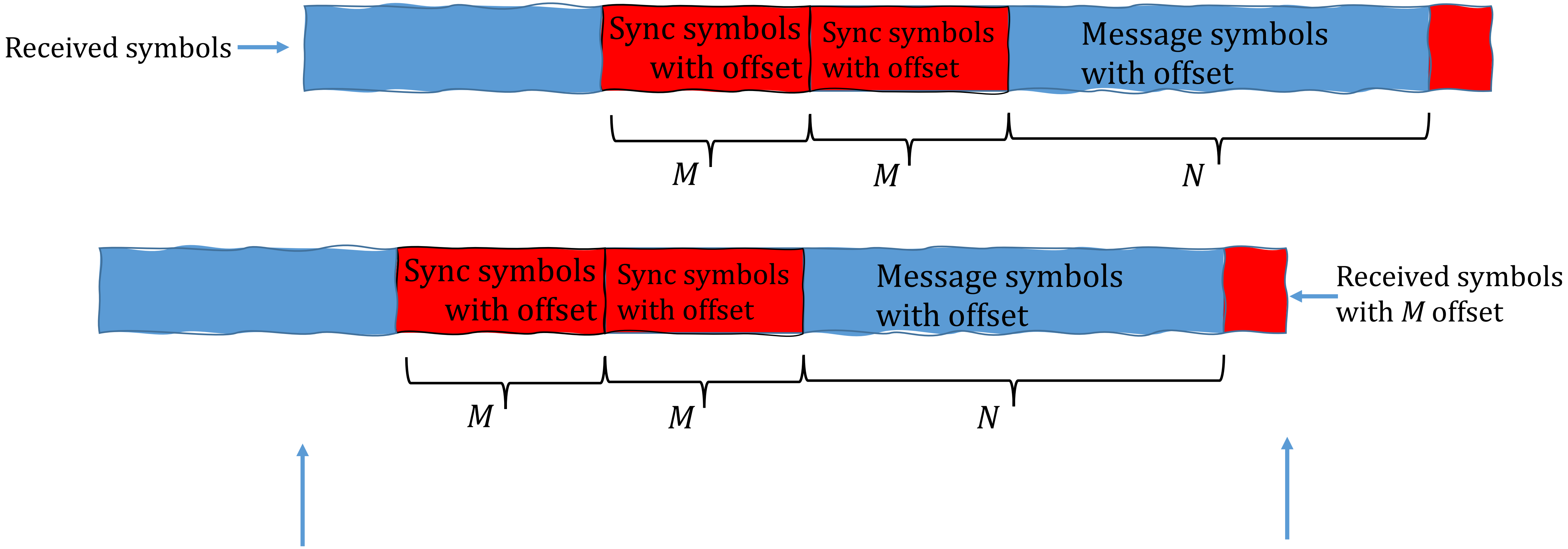


Step 2: In the Rx you receive for at least 3M+N symbols.



Detecting the message in presence of arbitrary frequency offset (2)

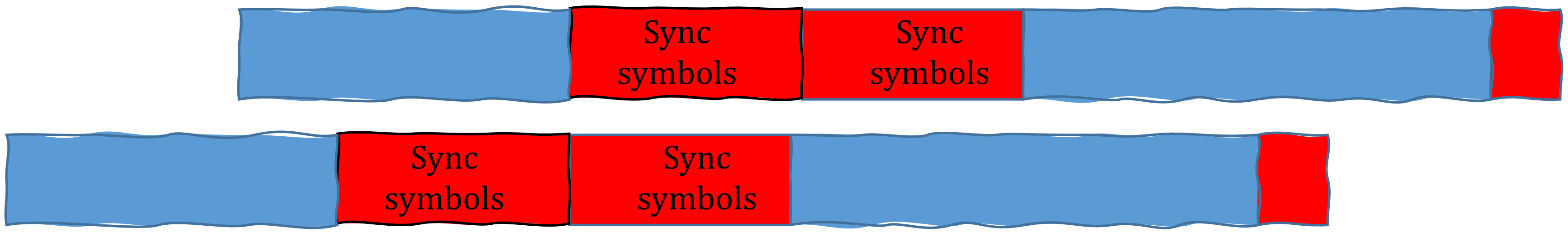
<u>Step 3</u>: You copy the received symbols ***conjugate*** them and ***offset*** them by M symbols



<u>Step 4</u>: You multiply both vectors element by element for the symbols where they overlap

Detecting the message in presence of arbitrary frequency offset (3)

Step 4: You multiply both vectors element by element for the symbols where they overlap



$$\overbrace{se^{j\theta}.\,e^{j(\omega t1+\psi)}}^{\text{Received symbol}}$$

$$*$$

$$\underbrace{se^{-j\theta}.\,e^{-j(\omega t2+\psi)}}_{\text{Received symbol conjugated}}$$

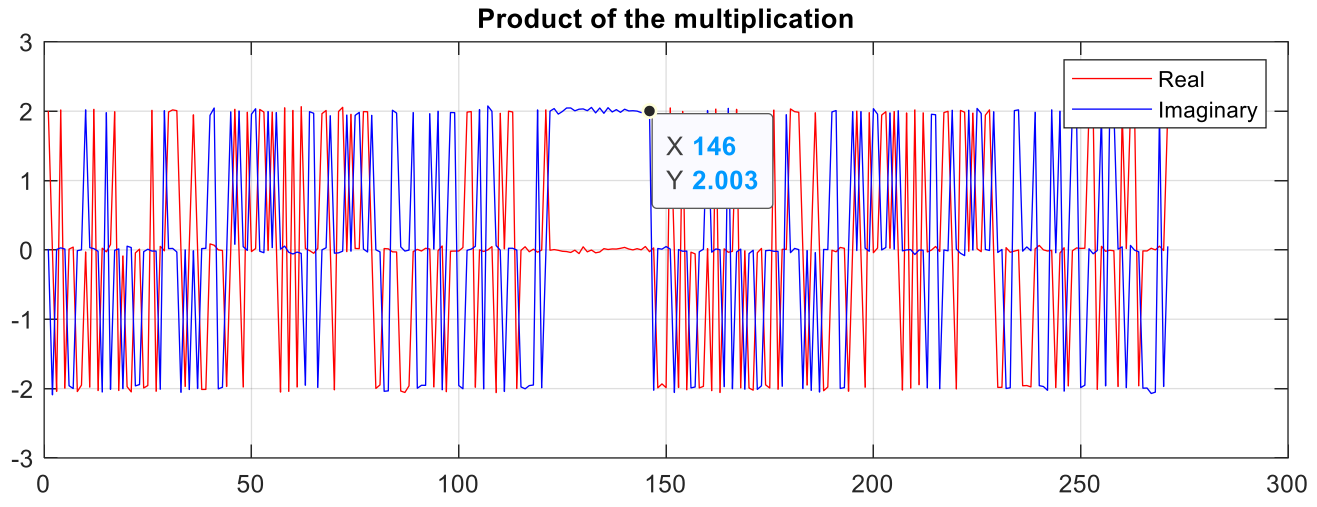
Only symbols that are ***the same*** will produce ...

$$\boxed{e^{j\omega(t1-t2)}}$$

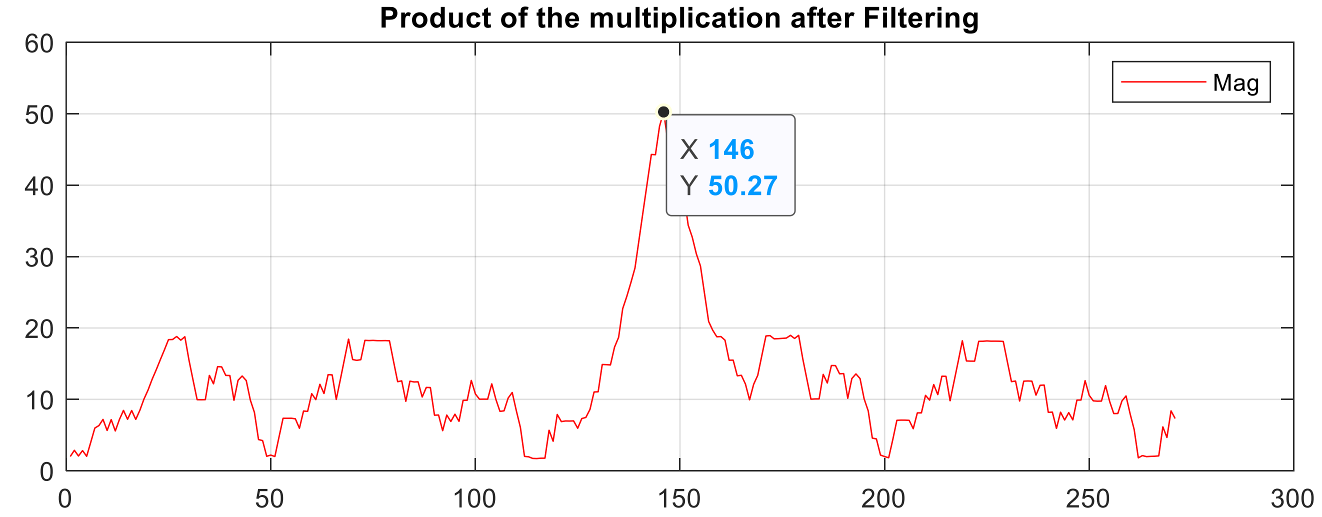
Detecting the message in presence of arbitrary frequency offset (5)

<u>Step 5</u>: You filter the product of the multiplication with a window of the same length as the Sync symbols

```
Product=(conj(Rx).*(Rx_offset));
```



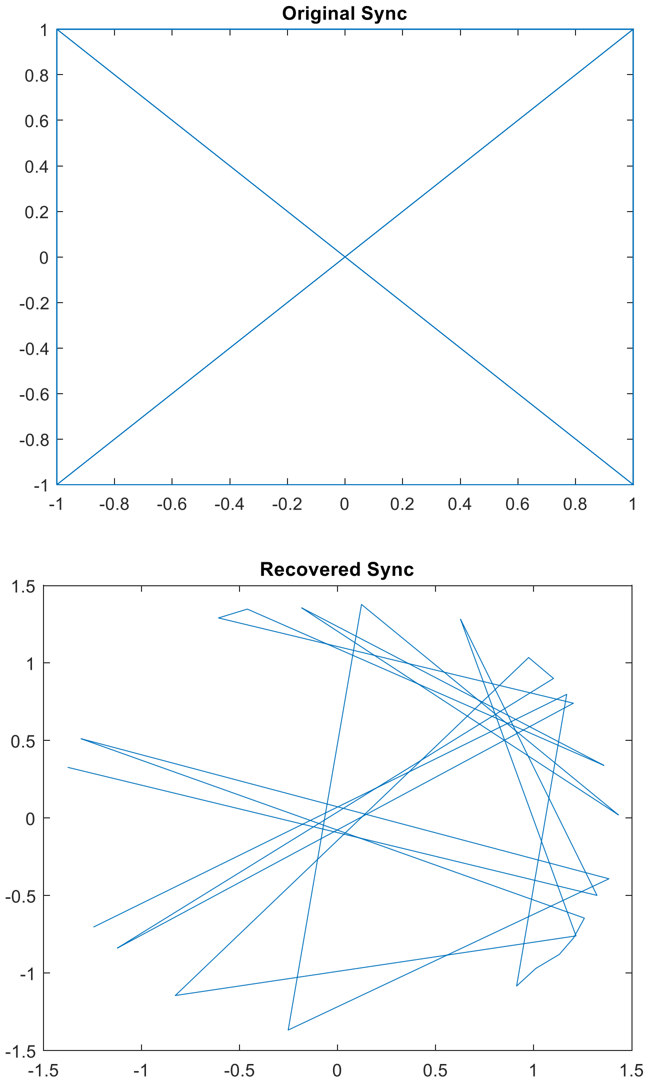

```
window = ones(1,M);
convv = filter(window,1,Product);
```

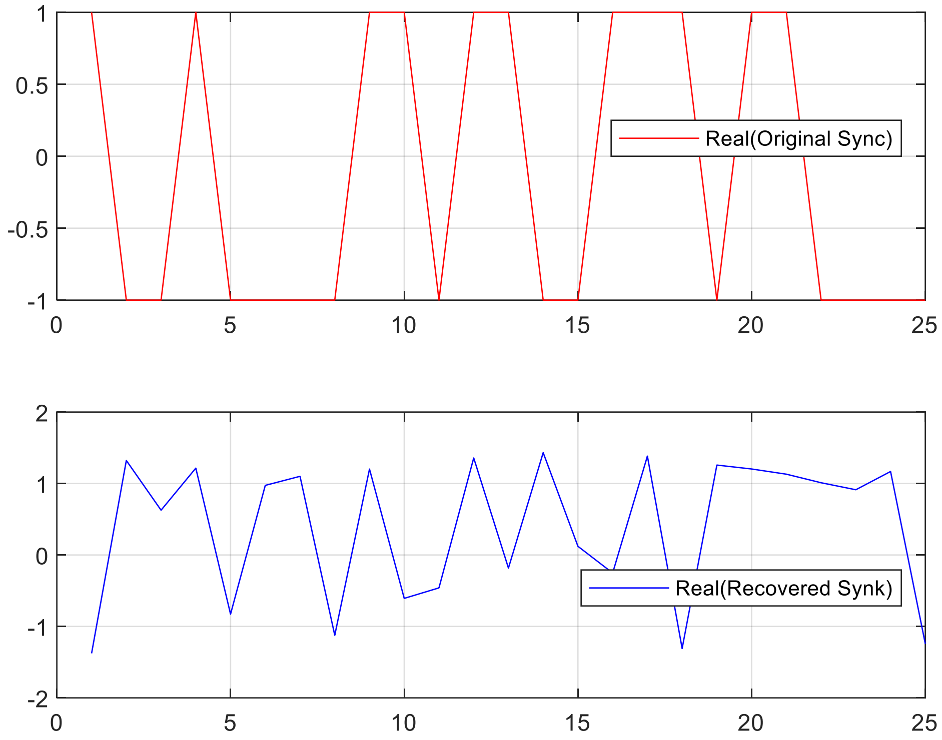


Find_Message_Start_Schmidl_Cox

Detecting the message in presence of arbitrary frequency offset (6)

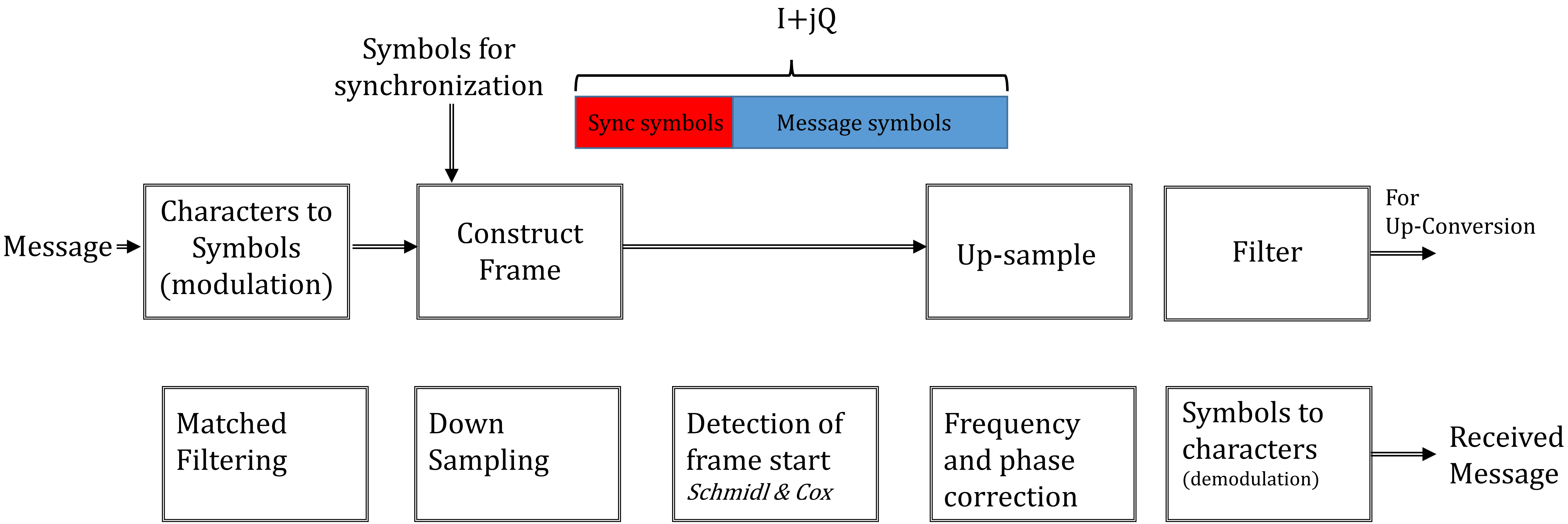
Step 6: You can now extract the Sync Symbols and use them to find the frequency and phase offset





Find_Message_Start_Schmidl_Cox

Frequency correction after down-sampling



Tx_Rx_Synk_f_Correction_after_DS_2

Frequency correction after down-sampling

Frequency and phase synchronization using the known synk symbols in the frame

1. Filter the incoming signal
2. Find the message start and isolate the synk symbols – `symb_synk_received` The frequency offset estimation is based on comparing the received to the known synk simbols `symb_synk`

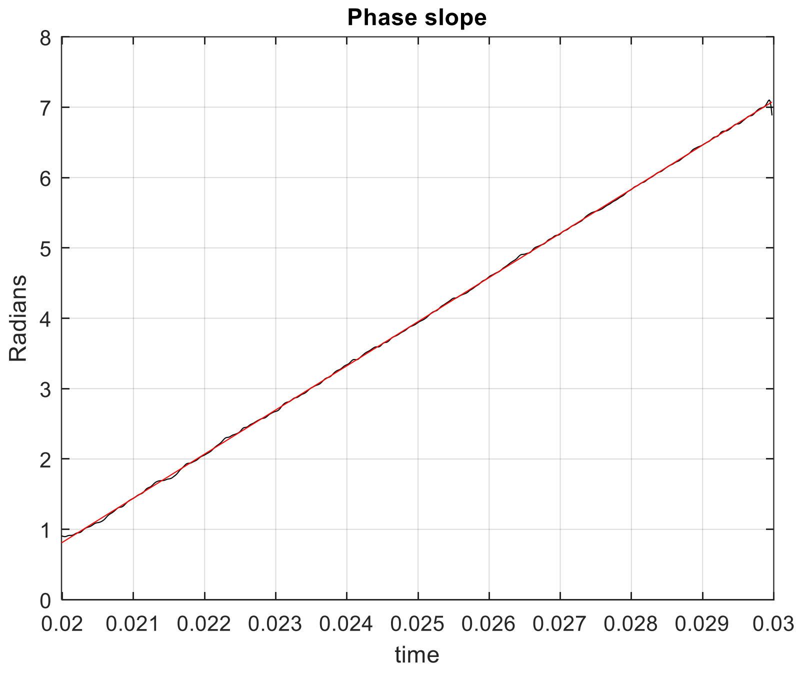
Original Sync symbols

*

Conj(Received Sync symbols f with offset)

=

Complex Number with increasing phase slope proportional to the freq. offset




```
phase_slope=unwrap(angle((symb_synk_received).*conj(symb_synk_us)));
```

Frequency correction after down-sampling

Frequency and phase synchronization using the known synk symbols in the frame

3. Fit a line to the phase slope: slope of the line = freq offst, value at t=0 equals the phase offset

```
p = polyfit(time_symb_synk,phase_slope,1);
phase_slope_fit = polyval(p,time_symb_synk);


freq_offset_estimated=p(1)/(2*pi)% In Hz
%phase_offset_estimated_deg=-360*p(2)/(2*pi)% In degrees
phase_offset_estimated_deg=360*(phase_slope(1))/(2*pi);
```

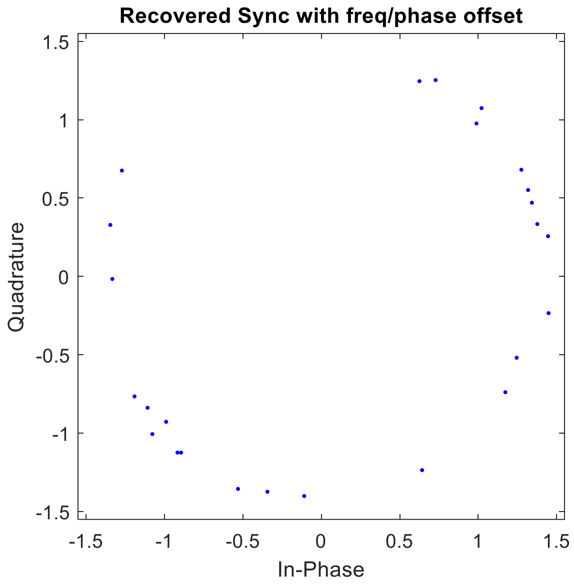
The “unwrap” function may not work on up-sampled symbols! Why?
The solution is to use the same value for all the synk symbols (for example 1+1j) – the drawback is that you introduce close to DC frequency component.

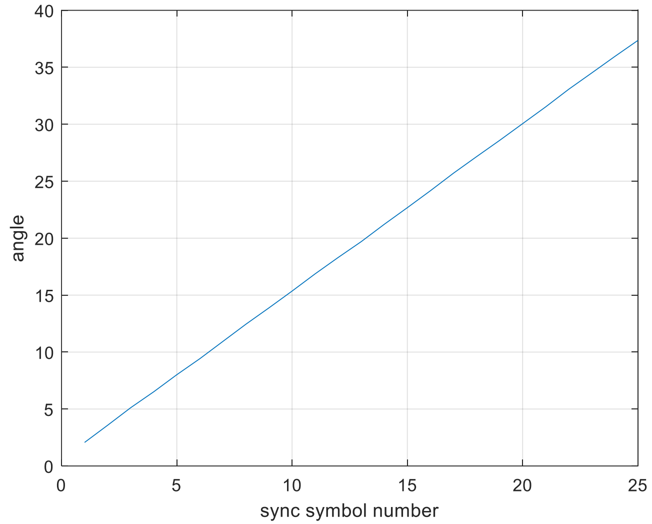
C:\Users\em7vevas\Dropbox\HT18\Matlab\Tx_Rx_Synk_f_Correction_2

Frequency correction after down-sampling

Find_Message_Start_Schmidl_Cox

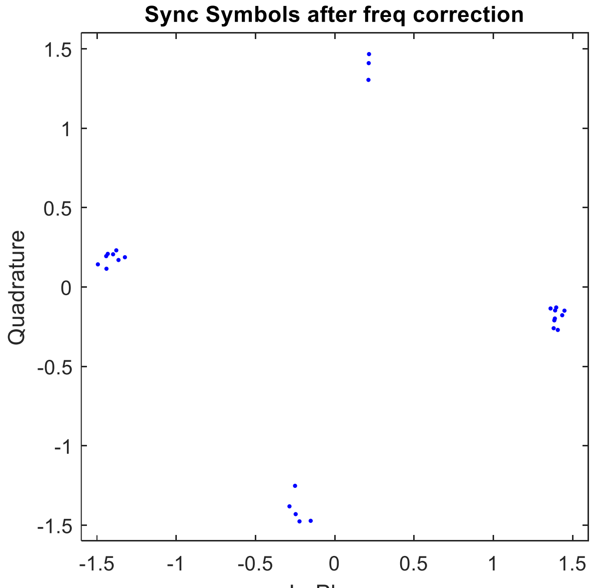
```
phase_slope=unwrap(angle(simb_synk_recovered)-angle(simb_synk));
```






```
freq_offset_detected=mean(diff(phase_slope))/(2*pi)
```

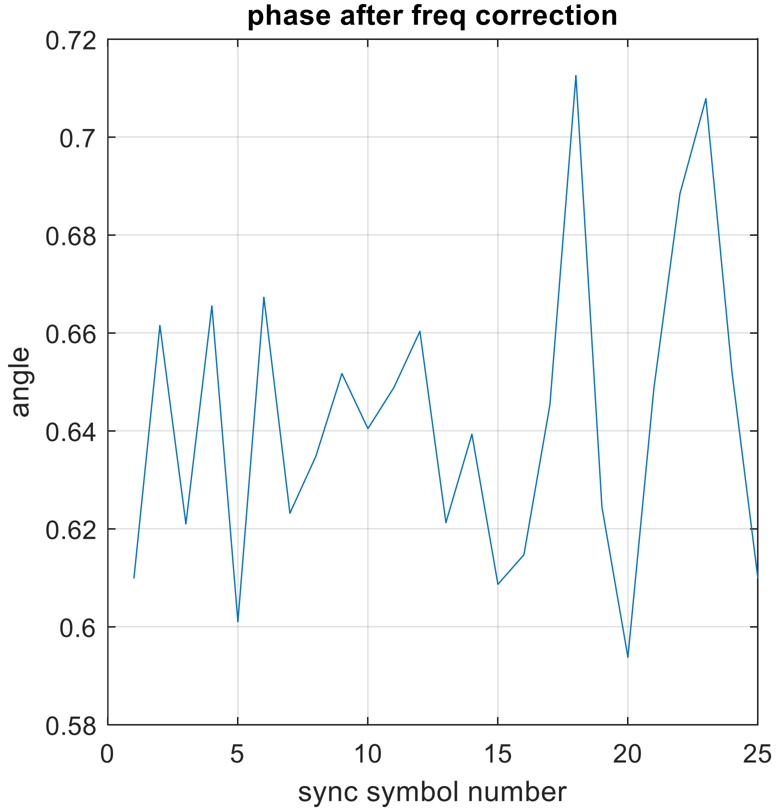
```
simb_synk_recovered=
simb_synk_recovered.*exp(i*(2*pi*freq_offset_detected*(1:1:length(simb_synk_reco
vered))));
```

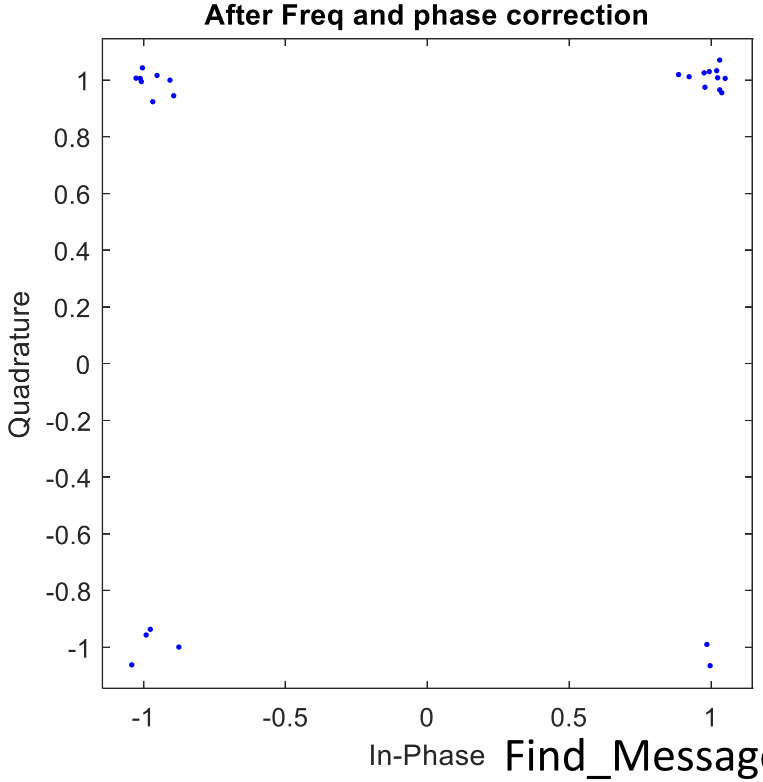


Phase offset correction after down-sampling

Symbols already corrected for
Freq. offset

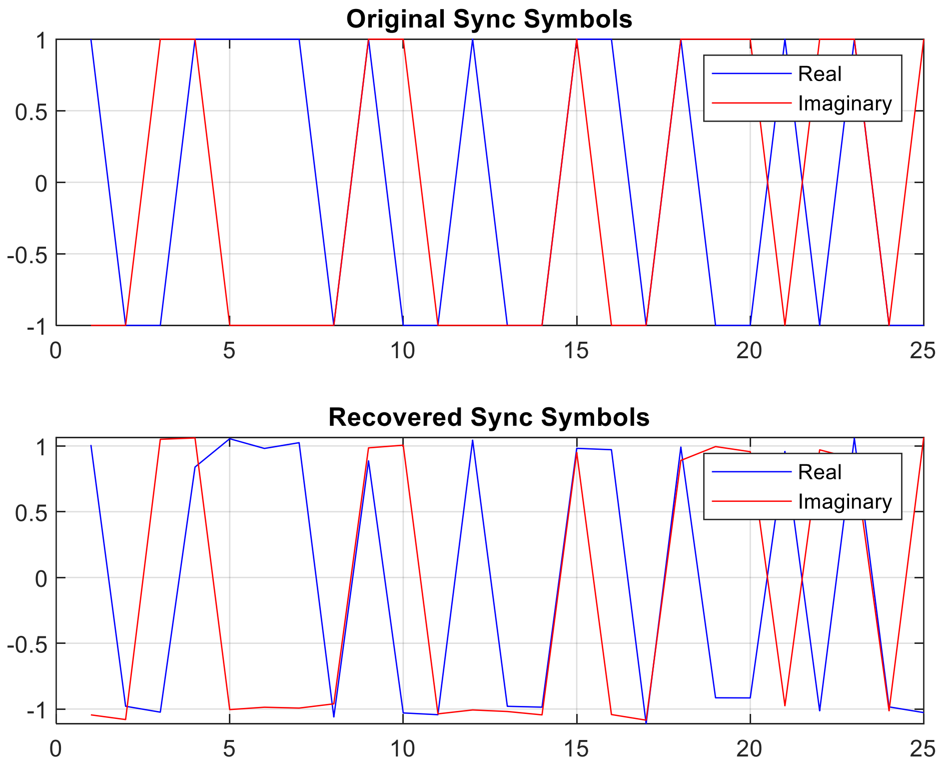
```
phase_offset=unwrap(angle(simb_synk_recovered)-angle(simb_synk));
phase_offset_detected=mean(phase_offset/(2*pi))
```



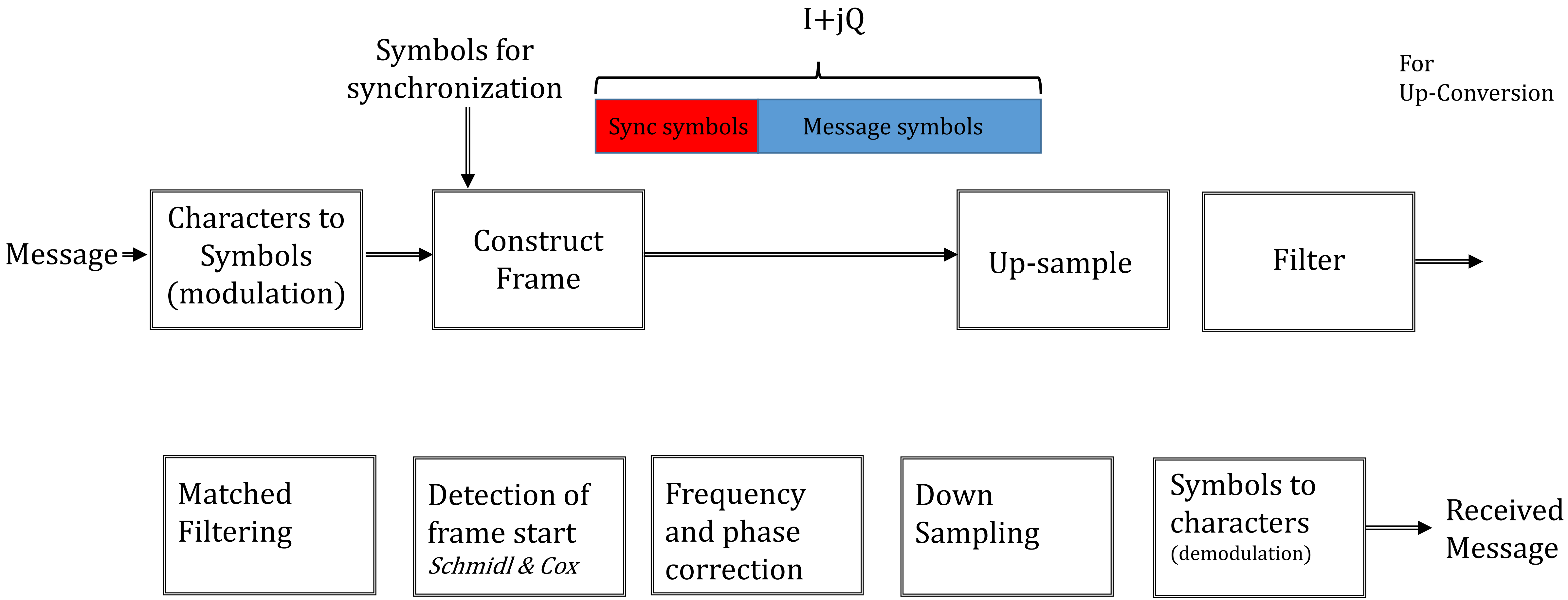


Find_Message_Start_Schmidl_Cox

Verification that the Sync Symbols are recovered



Frequency correction before down-sampling

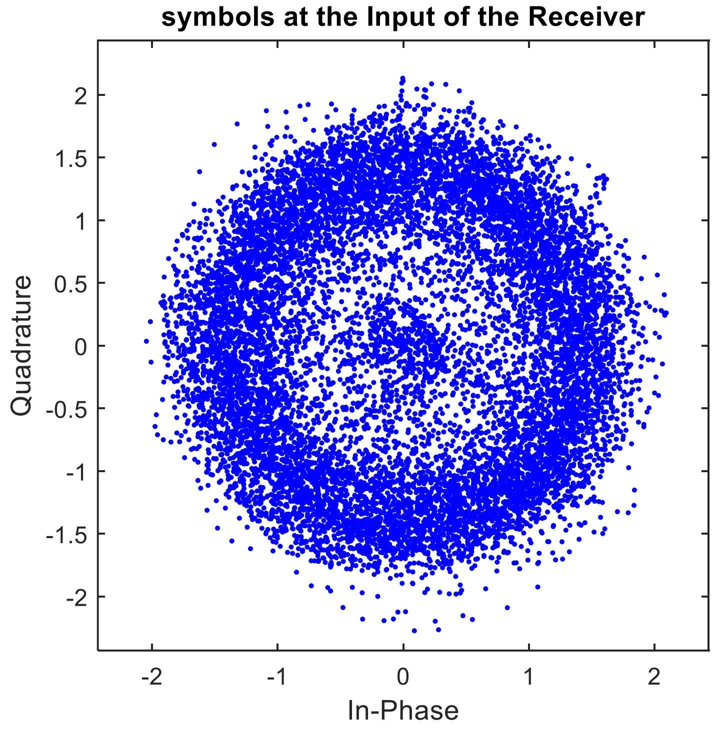


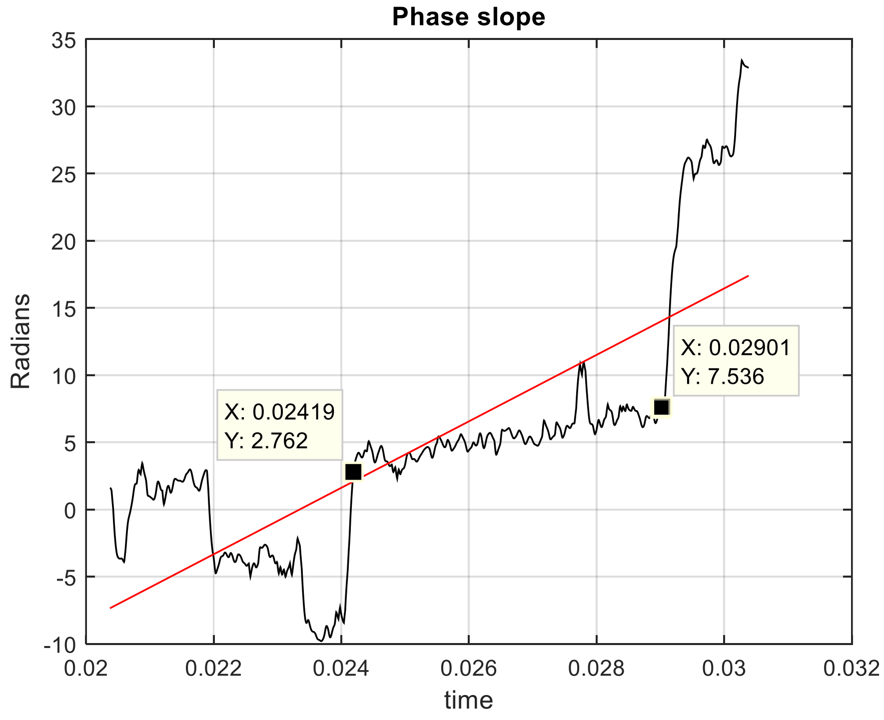
Tx_Rx_Synk_f_Correction_2.m

Frequency correction before down-sampling

Frequency and phase synchronization using the known synk symbols in the frame

Limitations: when used on an up-sampled signal and when the synk symbols are random there will be samples close to 0, and the unwrap function will not work properly and will produce jumps in the phase slope.

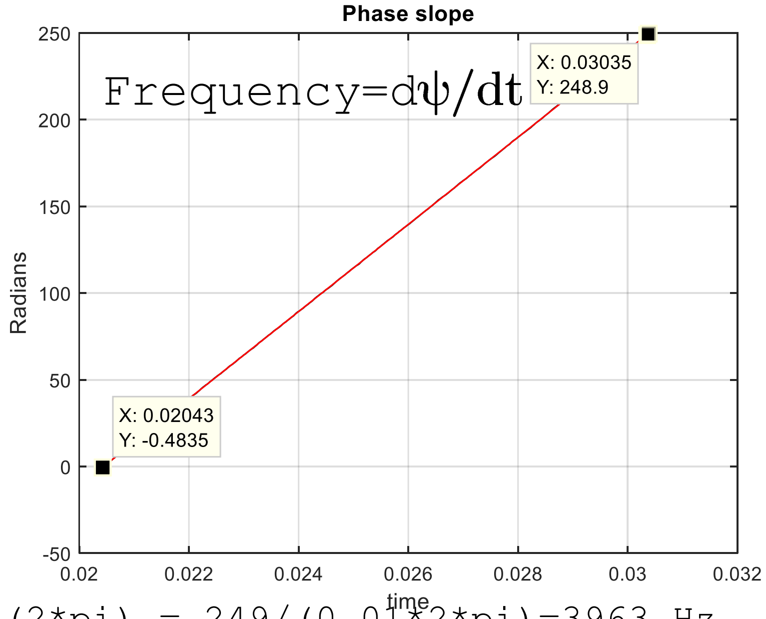




Frequency correction before down-sampling

Frequency and phase synchronization using the known synk symbols in the frame (before down sampling)

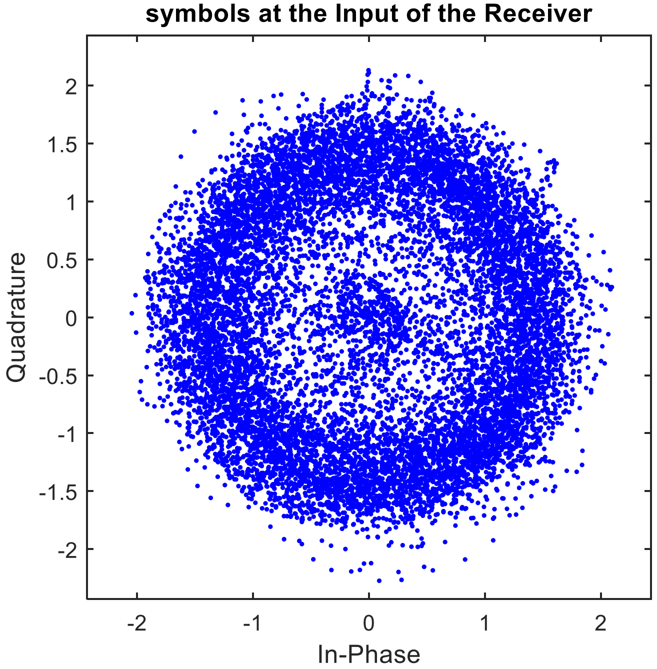
```
phase_slope=unwrap(angle(symb_synk_received)-angle(symb_synk_us));
```

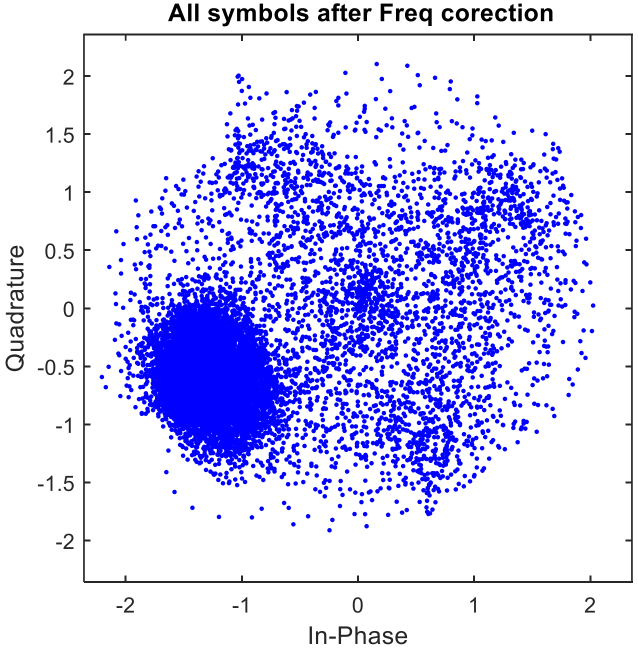


Example:

```
fsymb=20e3; %Symbol rate
s_per_symb=10; % Samples per symbol
fs=fsymb*s_per_symb;% Sampling rate
Phase_Offset=70; %In degrees
SNR=10; % In dB
Freq_Offset=4000; %In Hz
```

Frequency offset=estimated slope/(2*pi) = 249/(0.01*2*pi)=3963 Hz



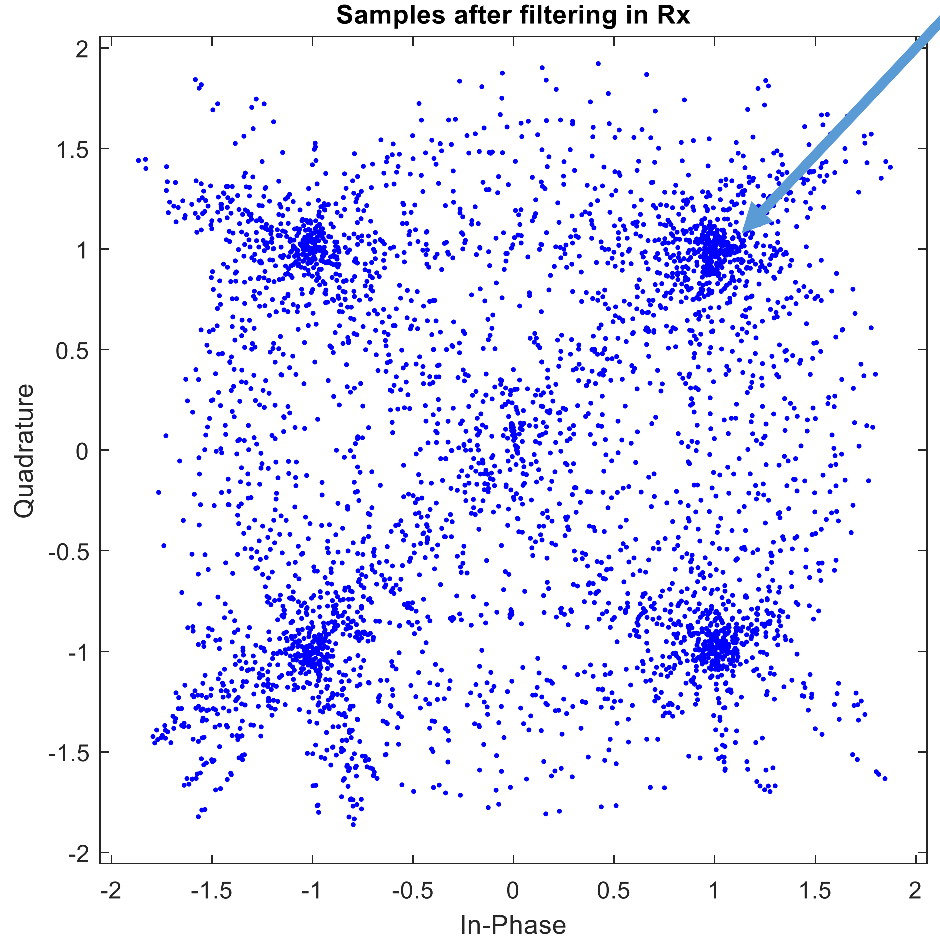


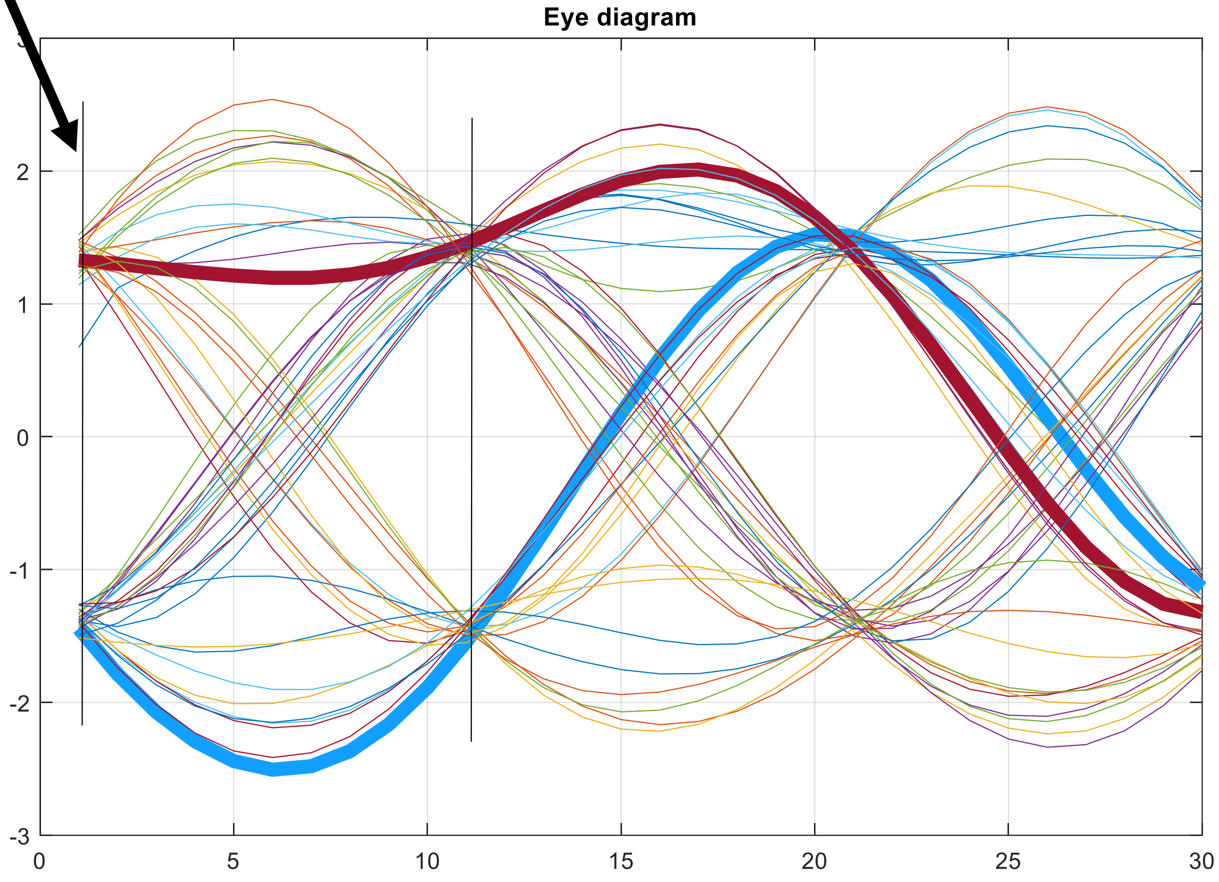
In this example all the synk symbols are 1+j1, thus the cloud

Down sampling (timing synchronization)

We need to take only one sample per Symbol

3 symbols, 10 samples/symbol

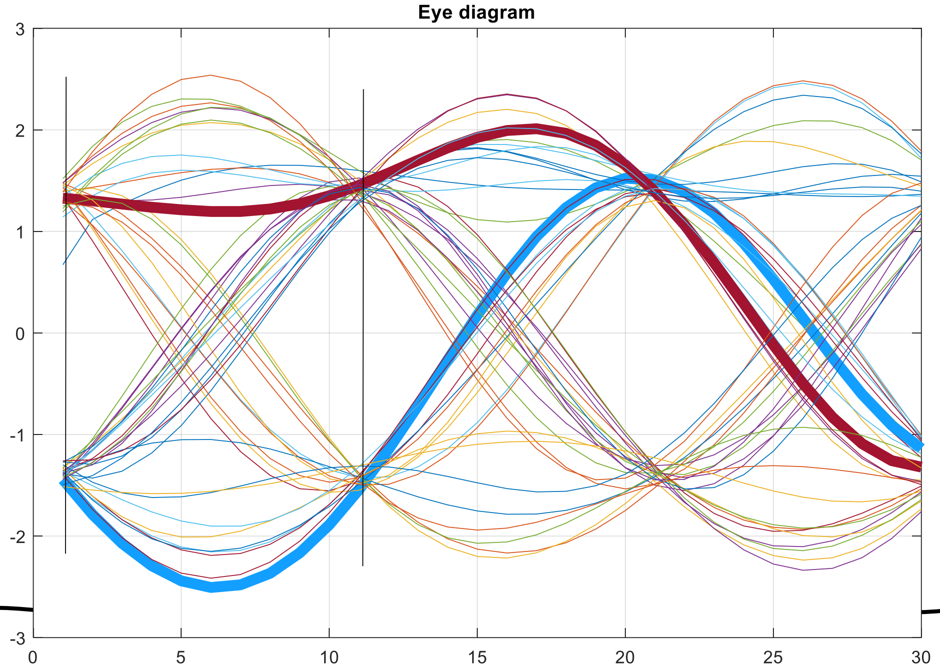


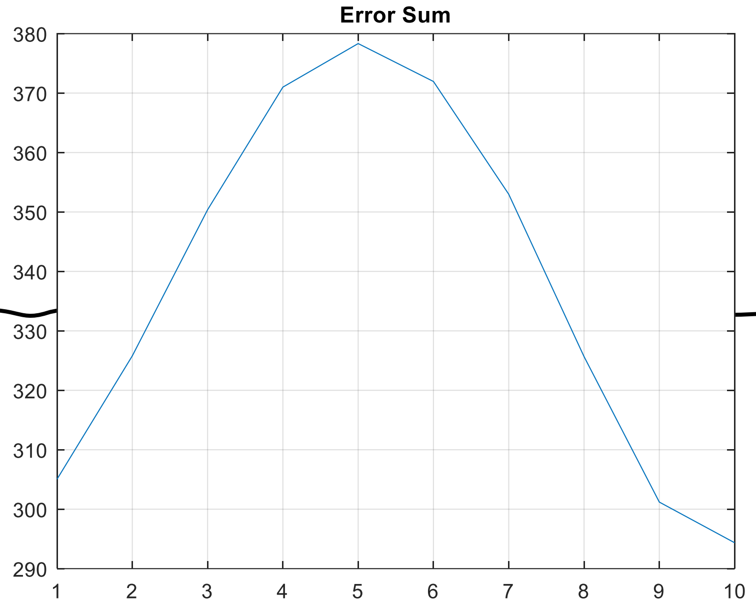



```
s_per_symb=10; % Samples per symbol
```

How to identify the sample that gives max eye opening?

Down-sampling






```
for k=2:1:length(symb)-1;
eI(k)=abs(real(symb(k-1))-real(symb(k+1)));
eQ(k)=abs(imag(symb(k-1))-imag(symb(k+1)));
e(k)=eI(k)+eQ(k);
end

error=reshape(e,s_per_symb,[]); %[s_per_symb length(symb)/s_per_symb]
error_sum=sum(error,2);% [samples_per_symbol 1]
```

Optimum sample is found at the minimum of the error sum

Down-sampling works even in presence of freq/phase offset