5G WIRELESS TECHNOLOGIES

Timing Recovery

Deniss Kolosovs
Deniss.Kolosovs@rtu.lv

Riga Technical university

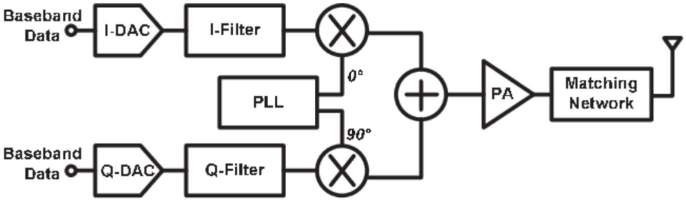
27th April 2021

Operations to perform in analog domain

Transmitter

Tasks for the transmitter:

- Digital-analog conversion;
- Smoothing-filter application;
- Quadrature modulation;
- Power amplification.



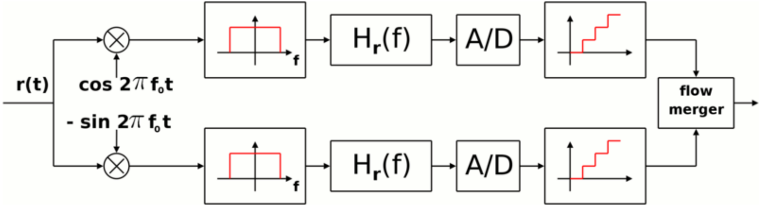
(a)

Operations to perform in analog domain

Receiver

Tasks for the receiver:

- Low-noise amplification;
- Automatic gain control;
- Image channel suppression and mixig (if necessary);
- Quadrature demodulation;
- Double-frequency component suppression;
- Anti-aliasing filter application;
- Analog-to-digital conversion.



Operations to perform in digital domain

- Tasks for the transmitter:
    - User data mapping to symbols and zeros insertion;
    - Pulse-shaping filtration;
    - Pre-distortions.
- Tasks for the receiver:
    - Distorting effect mitigation;
    - Timing and carrier recovery;
    - Matched filtration;
    - Detection.

Timing recovery

Reason and compensation objectives

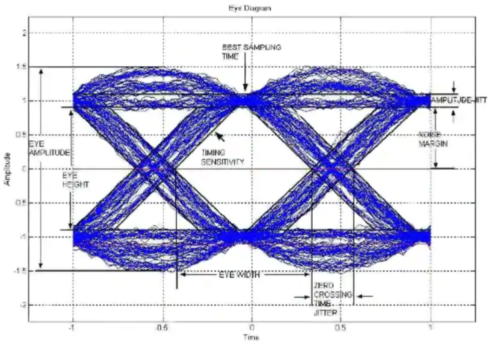
Time processes to be reconstructed:

- FPGA clock phase and frequency;
- Symbol position among other samples.

The difference in clock frequencies means:

- Data generation and acquision are performed with different frequencies;
- ADC and DAC are clocked with different clocks;
- Different sampling frequencies.

Usually, the difference is small (5–50ppm), but recovery is needed!



Timing recovery

Visual evidences

- In time domain, the signal is either shrinked or expanded—well visually witnessable from the timing diagrams of $I(t)$ and $Q(t)$. **Draw sampling process!**
- On constellation, appears as noisy points or a complete ravel.
- Considering frequency response, from the properties of the Fourier transform:

$$\begin{aligned} s(t) &\Leftrightarrow \dot{S}(\omega) \\ s(at) &\Leftrightarrow \frac{1}{a}\dot{S}\left(\frac{\omega}{a}\right). \end{aligned}$$

  Thus, the spectrum becomes wider or narrower.
- Definetely needs to be recovered! Performed in the modem, combined with the slicing procedure (because of the cost function).

Problem definition

Assume n-th baseband sample $x[n]$ should be delayed by $\tau[n]$. Depending on the form of $\tau[n]$, the following situations are possible:

- Constant integer delay $\tau[n] = t_0 \mod K \in Z$, where K is number of samples per symbol. Implies wrong sample symbol assumption as a symbol.
- Constant fractional delay $\tau[n] = t_0 < 1$ is a constant phase shift between transmitter and receiver clock signal generators.
- Linearly growing delay $\tau[n] = f_0 n$ describes the frequency difference f_0 between transmitter and receiver generators.
- Stochastic delay $\tau[n] = w[n]$ denotes frequency jitter.

Objective: introduce time-varying fractional delay $\tau[n]$ into a signal.

Simulation model

The reason for the distortion is the clock signal differences.

- Perform generation, transition D/A and A/D with different clocks. Shows the essence of the phenomenon, though impractical in simulation.
- Emulate this process resampling signal in a simulation model. Accurate theoretical model, but too calculation-consuming; non-real-time approach.
- Linear interpolation between samples (coefficients α and $1-\alpha$) in baseband. Simple implementation, but too inaccurate.

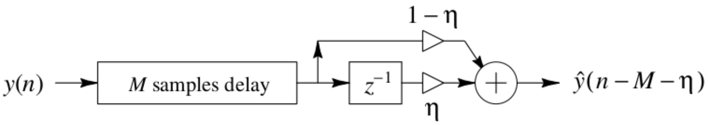
Objective clarification: Accurate interpolation (equivalent to the time shift) in baseband.

Linear interpolation

- Assume we have a discrete signal $y[n]$, which corresponds to continuous-time signal $y(t)$;
- To calculate signal's value at $t = n + \tau$, $(\tau < 1)$, one can use linear interpolation:

$$y(n - \tau) = (1 - \tau)y[n] + \tau y[n - 1];$$

- Filter-like structure;
- Frequency response—narrow-band low-pass filter. Inaccurate for wide-band signals, which have meaningful components near the Nyquist frequency.



Sinc interpolation

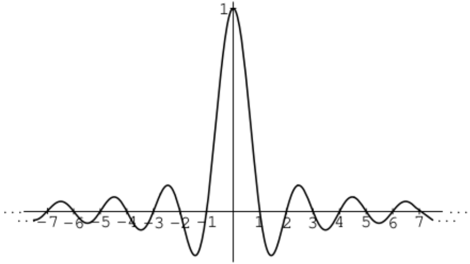
Ideal filter

- An interpolation output can be considered an FIR filter;
- Ideal filter is the one that ensures the signal recovery in the analog domain:

$$y(t) = \sum_{n=-\infty}^{\infty} y(nT)h_{id}(t - nT),$$

where T is a sampling step;
- To obtain an analog signal, the signal should be passed through square pulse; therefore,

$$h_{id}(t) = \operatorname{sinc}\left(\frac{\pi t}{T}\right).$$



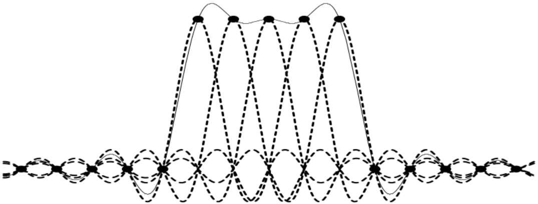
Sinc interpolation

Digital-to-analog conversion

- Substituting ideal filter impulse response into convolution expression:

$$y(t) = \sum_{n=-\infty}^{\infty} y(nT)\,\mathrm{sinc}\left(\frac{\pi}{T}(t - nT)\right);$$

- In the time domain, the recovered signal is a weighted sum of sinc-functions;
- Gives us a possibility to calculate values in between samples;
- Needs infinite sum of sinc-functions—can not be implemented in a real-world application.



Sinc interpolation

Delay filter I

Requirements for ideal delay filter:

- Constant amplitude-frequnecy response in range $(-f_N, f_N)$, i.e., square filter;
- Linear phase-frequency response with the slope $-2\pi\tau$.

That leads to:

- The requency response has only a phase-frequency component:

$$H_{id}(f) = \mathrm{e}^{j\varphi(f)} = \mathrm{e}^{-j2\pi f\tau};$$

- Group delay is equal:

$$\tau_g(f) = -\frac{1}{2\pi}\frac{\mathrm{d}\varphi(f)}{\mathrm{d}f} = \tau.$$

Sinc interpolation

Delay filter II

- The frequency response has only a phase-frequency component

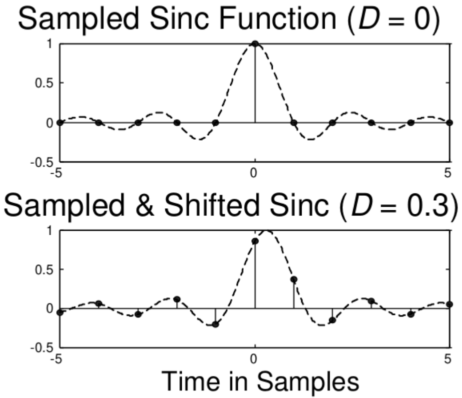
$$H_{id}(f) = \mathrm{e}^{j\varphi(f)} = \mathrm{e}^{-j2\pi f\tau};$$

- From the Fourier transfrom properties:

$$\begin{aligned} s(t) \quad &\Leftrightarrow \quad \dot{S}(\omega) \\ s(t-\tau) \quad &\Leftrightarrow \quad \dot{S}\left(\omega\right)\mathrm{e}^{-j\omega\tau}; \end{aligned}$$

- Thus, an impulse response of the ideal fractional delay τ filter is:

$$h(t) = \operatorname{sinc}\left(\frac{\pi(t-\tau)}{T}\right).$$



Sinc interpolation

Nyquist filter

- Instead, we can use any Nyquist filter with a band wider than the signal:

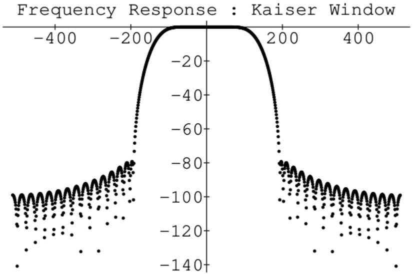
$$h(nT) = \begin{cases} 1; & n = 0 \\ 0; & n \neq 0 \end{cases}$$

- Truncation of sinc-function to N samples:

$$y(t) = \sum_{n=-N/2}^{N/2-1} y(nT)\,\mathrm{sinc}\left(\frac{\pi}{T}(t - nT)\right);$$

- Windowed $w(n)$ sinc-function:

$$y(t) = \sum_{n=-N/2}^{N/2-1} y(nT)w(n-\tau)\,\mathrm{sinc}\left(\frac{\pi}{T}(t - nT)\right).$$



Kaiser-Windowed Sinc Transform

Lagrange interpolation

Lagrange polynomial

- Polynomial interpolation for which N-th order polynomial interpolates $N+1$ points—zero-delay corresponds to the point itself;
- Assume we have $N+1$ points: $(x_0, y_0), \ldots, (x_j, y_j), \ldots, (x_k, y_k)$;
- The task is to find unique set of polynomials of the order N, which interpolates $y(x)$ in a form of linear combination:

$$y(x) = \sum_{j=0}^{k} y_j \ell_j(x),$$

where $\ell_j(x)$ is a Langrage polynomial:

$$\ell_j(x) = \prod_{\substack{0 \le m \le k \\ m \ne j}} \frac{x - x_m}{x_j - x_m} = \frac{(x - x_0)}{(x_j - x_0)} \cdots \frac{(x - x_{j-1})}{(x_j - x_{j-1})} \frac{(x - x_{j+1})}{(x_j - x_{j+1})} \cdots \frac{(x - x_k)}{(x_j - x_k)}.$$

Lagrange interpolation

Lagrange polynomial properties

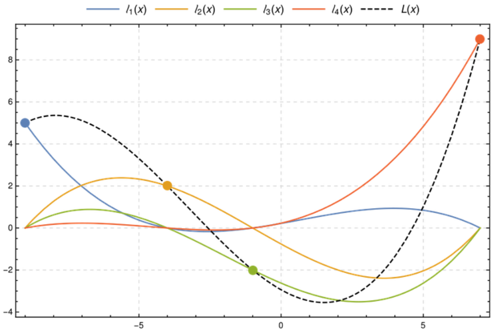
- The j-th Langrage polynomial:

$$\ell_j(x) = \prod_{\substack{0 \le m \le k \\ m \ne j}} \frac{x - x_m}{x_j - x_m};$$

- The numerator is 0 for all $j \neq k$, i.e.,

$$\ell_k(x_j) = \begin{cases} 1; & j = k \\ 0; & j \neq k \end{cases};$$

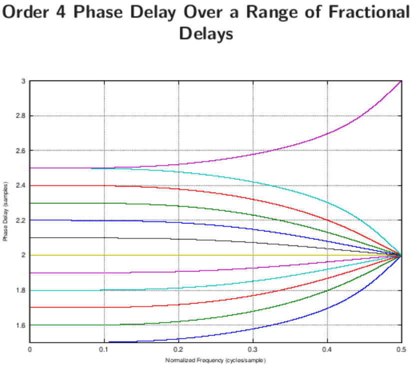
- For equally spaced samples x_k and infinite N, becomes a sinc-function;
- In this case, the value of $\ell_k(x)$ is equal to $\mathrm{sinc}(x - x_k)$;
- Maximally flat at DC.



Lagrange interpolation

Lagrange polynomial 4th order

$\Delta = 1.5:0.1:2.5$

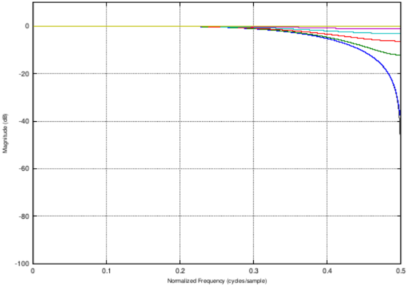


$\Delta = 1.5:0.1:2.5$ plus 2.499

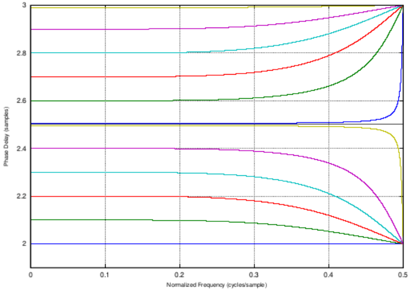
Lagrange interpolation

Lagrange polynomial 5th order

Order 5 Amplitude Response Over a Range of Fractional Delays



Order 5 Phase Delay Over a Range of Fractional Delays



$\Delta = 2.0 : 0.1 : 3.0$ plus 2.495 and 2.505

Interpolation FIR filter I

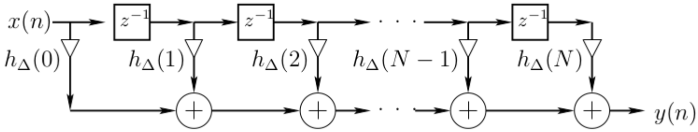
- Assume samples are spaced with T, i.e., with sampling step;
- An interpolated value of $y(t)$ is equal to:

$$y(n+\tau) = \sum_{k=0}^{K} y(nT)\ell_k(\tau) = \sum_{k=0}^{K} y(nT)h_\tau(k),$$

where the impulse response is equal to:

$$h_\tau(n) = \prod_{\substack{k=0 \\ k\neq n}} \frac{\tau - k}{n - k};$$

- FIR filter structure:



Interpolation FIR filter II

- The impulse response is equal to:

$$h_\tau(n) = \prod_{\substack{k=0 \\ k \neq n}} \frac{\tau - k}{n - k};$$

- Calculating coefficients:

Order	$h_\tau(0)$	$h_\tau(1)$	$h_\tau(2)$	$h_\tau(3)$
$N=1$	$1-\tau$	τ		
$N=2$	$\frac{(\tau-1)(\tau-2)}{2}$	$-\tau(\tau-2)$	$\frac{\tau(\tau-1)}{2}$	
$N=3$	$-\frac{(\tau-1)(\tau-2)(\tau-3)}{6}$	$\frac{\tau(\tau-2)(\tau-3)}{2}$	$-\frac{\tau(\tau-1)(\tau-3)}{2}$	$\frac{\tau(\tau-1)(\tau-2)}{6}$

- Create coefficients calculating function in Matlab.

Farrow structure

Optimization idea

- The impulse response can be expressed as:

$$h_\tau(n) = \sum_{m=0}^{N_c} c_n(m)\tau^m;$$

- Then z-transform is:

$$H_\tau(z) = \sum_{n=0}^{N_h} h_\tau z^{-n} = \sum_{n=0}^{N_h}\left[\sum_{m=0}^{N_c} c_n(m)\tau^m\right] z^{-n} = \sum_{m=0}^{N_c}\left[\sum_{n=0}^{N_h} c_n(m) z^{-n}\right]\tau^m = \sum_{m=0}^{N_c} C_m(z)\tau^m;$$

- Applying Horner's method:

$$Y_\tau(z) = X(z)\sum_{m=0}^{N_c} C_m(z)\tau^m = C_0(z)X(z) + \tau\left[C_1(z)X(z) + \tau\left[\cdots + \tau C_{N_c-1}(z)X\right]\right].$$

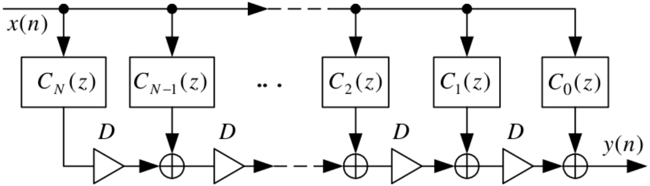
Farrow structure

Filter structure

- The output of the Farrow structure is:

$$Y_\tau(z) = X(z)\sum_{m=0}^{N_c} C_m(z)\tau^m = C_0(z)X(z) + \tau\left[C_1(z)X(z) + \tau\left[\cdots + \tau C_{N_c-1}(z)X\right]\right];$$

- The structure of the Farrow filter:



Farrow structure

Polynomial calculation

- Polynomial $C_m(z)$ coefficients can be calculated from:

$$z^{-\tau} = \sum_{m=0}^{N_c} C_m(z)\tau^m \text{ for } \tau = 0, \cdots N_c;$$

- Constructing system of the linear equations in matrix form $\mathbf{MC} = \mathbf{z}$ for $N_c = 2$:

$$\begin{bmatrix} C_0(z)0^0 + C_1(z)0^1 + C_2(z)0^2 \\ C_0(z)1^0 + C_1(z)1^1 + C_2(z)1^2 \\ C_0(z)2^0 + C_1(z)2^1 + C_2(z)2^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} C_0(z) \\ C_1(z) \\ C_2(z) \end{bmatrix} = \begin{bmatrix} 1 \\ z^{-1} \\ z^{-2} \end{bmatrix};$$

- Polynomicals $C(z)$ can be found as:

$$\mathbf{C} = \mathbf{M}^{-1}\mathbf{MC} = \mathbf{M}^{-1}\mathbf{z};$$

- As a result, our goal is simply to calculate inverse matrix $\mathbf{M}^{-1}$!

Farrow structure

Filter construction

- Calculating inverse matrix, obtain:

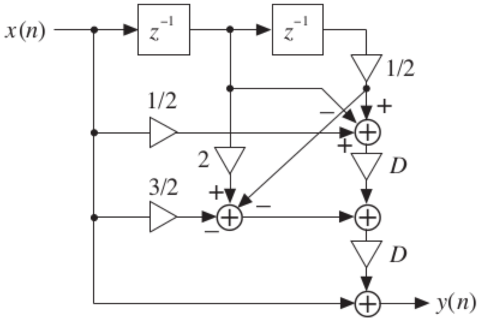
$$\mathbf{M}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -3/2 & 2 & -1/2 \\ 1/2 & -1 & 1/2 \end{bmatrix};$$

- Thus, polynomials are:

$$\begin{aligned} C_0(z) &= 1; \\ C_1(z) &= -\tfrac{3}{2} + 2z^{-1} - \tfrac{1}{2}z^{-2}; \\ C_2(z) &= \tfrac{1}{2} - z^{-1} + \tfrac{1}{2}z^{-2}; \end{aligned}$$

- Recall, filter's frequency response is in the form:

$$H_\tau(z) = \sum_{m=0}^{N_c} C_m(z)\tau^m \text{ for } \tau = 0, \cdots N_c.$$



Farrow structure

Modified Farrow filter

- For our implementation, there were no restrictions on τ value, i.e., $\tau \in (0, N_c)$;
- Let us limit its range to $\tau - \frac{N_c+1}{2} \in (0, 1)$;
- It can be done (without proof), introducing a transformation matrix $\mathbf{T}$, whose each element is defined as:

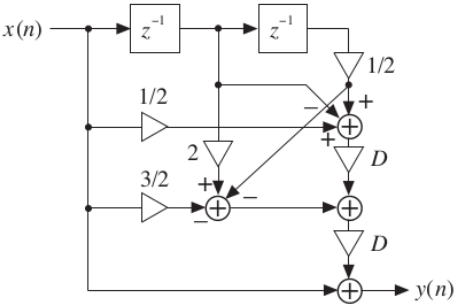
$$T_{n,m} = \begin{cases} \left\lceil \dfrac{N_c}{2} \right\rceil^{n-m} \dbinom{n}{m} & \text{if } n \geqslant m \\ 0 & \text{if } n < m; \end{cases}$$

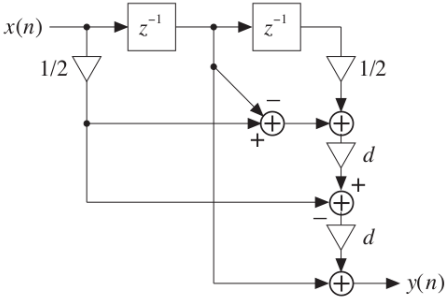
- To apply the transformation, we have to substitute $\mathbf{M}^{-1}$ by $\mathbf{TM}^{-1}$;
- For the $N_c = 2$ polynomials order example, we have:

$$\mathbf{TM}^{-1} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ -3/2 & 2 & -1/2 \\ 1/2 & -1 & 1/2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1/2 & 0 & 1/2 \\ 1/2 & -1 & 1/2 \end{bmatrix}.$$

Farrow structure

Original Farrow filter vs Modified Farrow filter



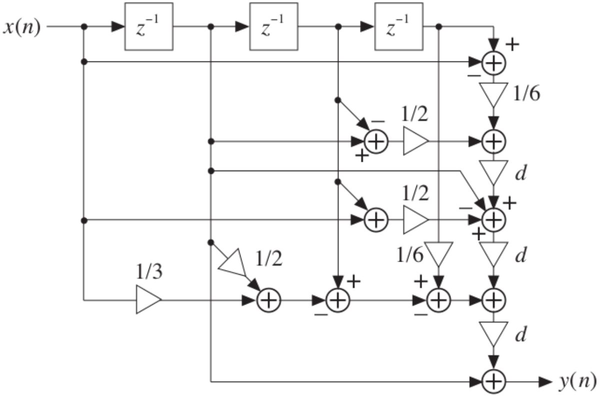


Farrow structure

3rd order example

- Matrix of the polynomials' coefficients:

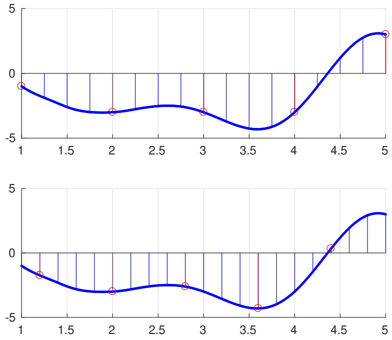
$$\mathbf{TM}^{-1} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{1}{2} & -\frac{1}{2} & 1 & -\frac{1}{6} \\ \frac{1}{2} & -1 & \frac{1}{2} & 0 \\ -\frac{1}{6} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{6} \end{bmatrix}.$$



Interpolation usage for timing desynchronization

Phase difference

- Positive constant phase shift τ:
  - Apply filter of order N that implements delay τ;
  - Delete $N/2$ first output signal samples to compensate filter's integer delay;
- Negative constant phase shift τ:
  - Apply filter of order N that implements delay $1-\tau$;
  - Delete $N/2-1$ first output signal samples to compensate filter's integer delay;
- Positive/negative frequency difference.



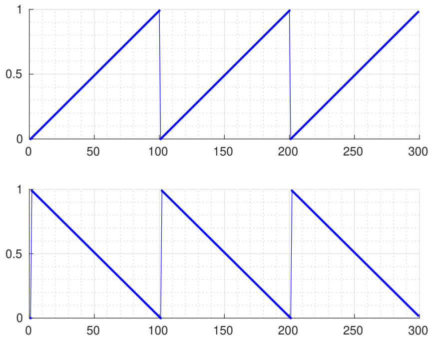
Interpolation usage for timing desynchronization

Frequency difference

- Calculate unique delay (and filter) for each sample:

$$\tau[n] = f_0 n \mod 1;$$

- Apply different filters at each clock cycle;
- At the phase jumps:
  - For higher frequency (lower period), use the same inputs twice;
  - For lower frequency (higher period), skip current input sample set;
- Delete $N/2$ first output signal samples to compensate filter's integer delay.
- Draw increments and see in Matlab.



Interpolation usage for timing desynchronization

Lower frequency example

```
phi =

         0    0.2300    0.4600    0.6900    0.9200    0.1500    0.3800    0.6100    0.8400    0.0700    0.3000    0.5300


ans =

    0.0000    0.0000    0.0000    0.0000    0.0000    1.0000    0.0000    0.0000    0.0000    1.0000    0.0000    0.0000


x =

    0.3166    1.1867    1.2422   -0.5702   -0.5873   -0.7546    0.1898   -0.4181   -0.4842    1.1303   -1.1664    0.7609


y =

    0.3166    1.1867    1.2422   -0.5702   -0.5873   -0.7546    0.1898   -0.4181   -0.4842    1.1303   -1.1664    0.7609
         0    0.3166    1.1867    1.2422   -0.5702   -0.5873   -0.7546    0.1898   -0.4181   -0.4842    1.1303   -1.1664
         0         0    0.3166    1.1867    1.2422   -0.5702   -0.5873   -0.7546    0.1898   -0.4181   -0.4842    1.1303
         0         0         0    0.3166    1.1867    1.2422   -0.5702   -0.5873   -0.7546    0.1898   -0.4181   -0.4842


ans =

    0.3166    1.1867    1.2422   -0.5702   -0.5873    0.1898   -0.4181   -0.4842   -1.1664    0.7609    0.7950   -1.4576
         0    0.3166    1.1867    1.2422   -0.5702   -0.7546    0.1898   -0.4181    1.1303   -1.1664    0.7609    0.7950
         0         0    0.3166    1.1867    1.2422   -0.5873   -0.7546    0.1898   -0.4842    1.1303   -1.1664    0.7609
         0         0         0    0.3166    1.1867   -0.5702   -0.5873   -0.7546   -0.4181   -0.4842    1.1303   -1.1664
```

Interpolation usage for timing desynchronization

Higher frequency example

```
phi =

        0    0.7100    0.4200    0.1300    0.8400    0.5500    0.2600    0.9700    0.6800    0.3900    0.1000    0.8100


ans =

    0.0000    1.0000    0.0000    0.0000    1.0000    0.0000    0.0000    1.0000    0.0000    0.0000    0.0000    1.0000


x =

    0.3359   -1.8788    1.2296    0.9412   -1.1044    0.1236   -0.7855   -1.7368   -0.1915   -0.2251    1.0809   -1.0540


y =

    0.3359   -1.8788    1.2296    0.9412   -1.1044    0.1236   -0.7855   -1.7368   -0.1915   -0.2251    1.0809   -1.0540
         0    0.3359   -1.8788    1.2296    0.9412   -1.1044    0.1236   -0.7855   -1.7368   -0.1915   -0.2251    1.0809
         0         0    0.3359   -1.8788    1.2296    0.9412   -1.1044    0.1236   -0.7855   -1.7368   -0.1915   -0.2251
         0         0         0    0.3359   -1.8788    1.2296    0.9412   -1.1044    0.1236   -0.7855   -1.7368   -0.1915


ans =

    0.3359    0.3359   -1.8788    1.2296    0.9412    0.9412   -1.1044    0.1236   -0.7855   -0.7855   -1.7368   -0.1915
         0         0    0.3359   -1.8788    1.2296    1.2296    0.9412   -1.1044    0.1236    0.1236   -0.7855   -1.7368
         0         0         0    0.3359   -1.8788   -1.8788    1.2296    0.9412   -1.1044   -1.1044    0.1236   -0.7855
         0         0         0         0    0.3359    0.3359   -1.8788    1.2296    0.9412    0.9412   -1.1044    0.1236
```

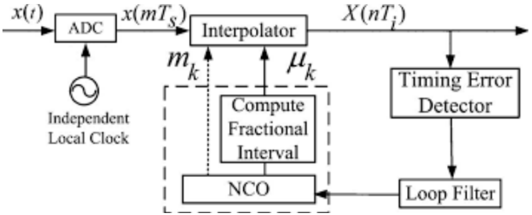
Main dynamic system construction stages

The following requirements must be fulfilled to create a dynamic system:

- Define a way how to compensate a distortion if its amount in the signal is static precisely known—compensation application;
- Understand how distortion affects a signal and find a mathematical expression that shows the consequences of its presence—cost function;
- Find an algorithm to calculate an amount of compensation from the cost function—compensation algorithm.

Timing recovery block implementation

- **Goal:** recover samples that correspond to symbols and create a marker that indicates which one is a symbol;
- Correction application: interpolation and skip-sample management;
- Cost function:
  - Gardner algorithm;
  - Early-late algorithm;
  - Band-edge-based cost function.
- Compensation algorithm derivation uses a stochastic gradient approach.



Gardner scheme

Cost function definition

- An application of the correction looks like:

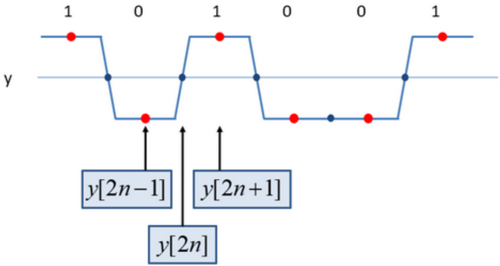
$$y[n] = f(x[n], \tau[n-1]) = x[n - \tau[n-1]];$$

- Cost function that minimizes error at the moment of the transition between symbols $y[n]$:

$$J[n] = E\left[(y[n] - \hat{y}[n])^2\right]$$

- Then increment of the delay value $\tau[n]$ is expressable through:

$$\begin{aligned} \tau[n] &= \tau[n-1] - \mu \frac{\mathrm{d}J[n]}{\mathrm{d}\tau[n-1]} = \\ &= \tau[n-1] + 2\mu(y[n] - \hat{y}[n])y'[n] \end{aligned}$$



Gardner scheme

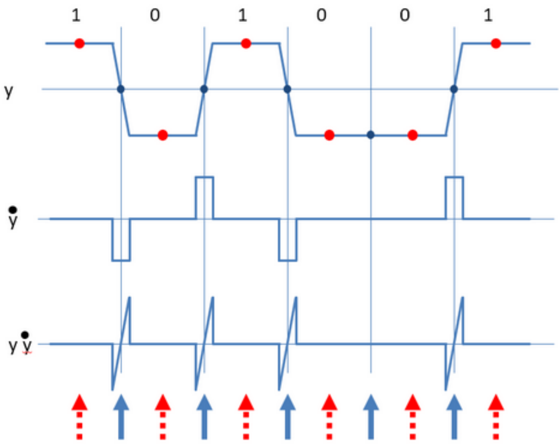
Timing error explanation

- The output of the dynamic block at time moment n is $y[n]$; previous and next samples are $y[n-1]$ and $y[n+1]$, correspondingly;
- The cost function in the case of $\hat{y}[n] = 0$ is a multiplication of the sample and its derivative:

$$e[n] = y[n]y'[n];$$

- As the precise value is not known, we approximate it:

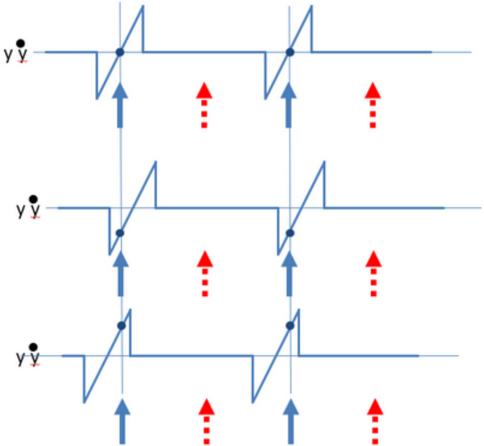
$$e[n] \simeq y[n](y[n+1] - y[n-1]).$$



Gardner scheme

Timing error examples

- If sampling moment corresponds to transition through zero ($y[n] = 0$), error also is equal to zero $e[n] = 0$;
- If sampling moment advances transition through zero, derivative and sampling result are of different signs, and error is negative $e[n] < 0$;
- If sampling moment is delayed after transition through zero, derivative and sampling result is of the same sign, and error is positive $e[n] > 0$.



Early-late detector

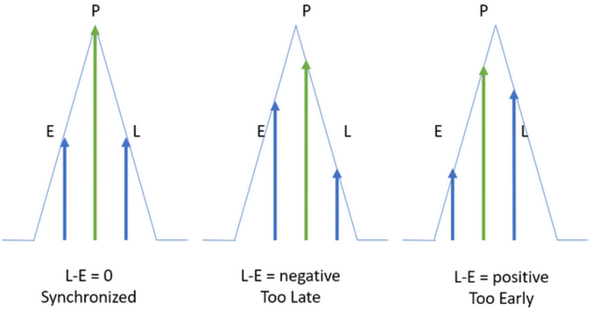
Timing error examples

- **Goal** is to find place in the signal, where derivative is zero;
- Symbol point is $y[n]$—compare to Gardner scheme;
- Cost function approximation:

$$\frac{\mathrm{d}y^2[n]}{\mathrm{d}\tau[n-1]} \simeq \frac{y^2[n+1]-y^2[n-1]}{2};$$

- In this way, instantaneous delay is:

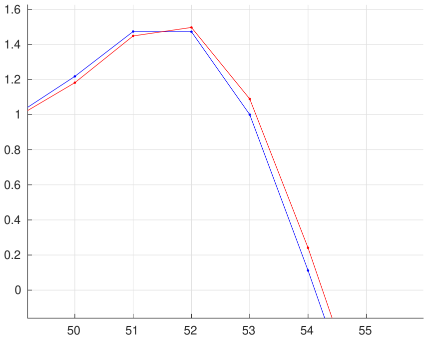
$$\tau[n] = \tau[n-1] - \mu\frac{y^2[n+1]-y^2[n-1]}{2}.$$



Application to QAM timing recovery

- In the case of PAM, process signal; in the case of QAM—absolute values;
- Mueller–Müller algorithm for only symbols—not covered in this course;
- Gardner scheme cost function and decision-aided recovery;
- The cost function tracks symbol position $y[n]$, not a sample between symbols;
- A value $\hat{y}[n] \neq 0$ now is not equal to zero;
- Error function becomes:

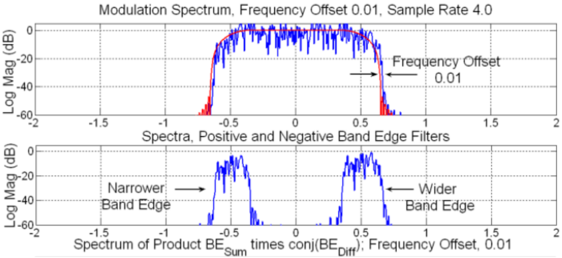
$$e[n] \simeq (y[n] - \hat{y}[n])(y[n+1] - y[n-1])$$



Band-edge cost function

Idea

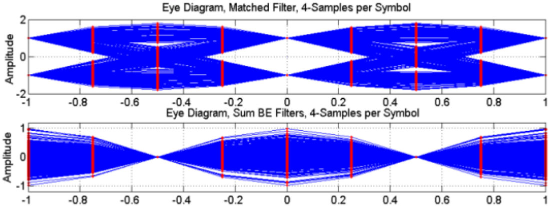
- In parallel to the data flow, it is possible to introduce additional processing;
- At the frequency that corresponds to the symbols sequence, there is information on the data rate, but no data itself;
- Filter out the edge of the signal frequency band.



Band-edge cost function

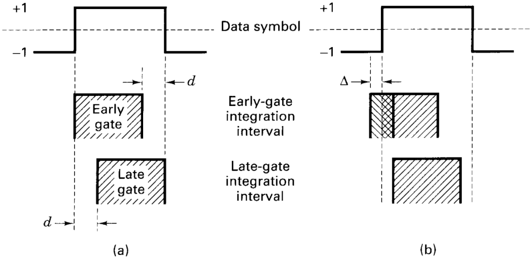
Application

- Maximal value corresponds to the symbol; zero—to the transition;
- Symmetrical pulses;
- Use early-late algorithm;
- Advantage: detection is not needed, there are no wrong adjustments;
- Drawbacks: either noisy or high latency; additional circuitry.



Binary symbol synchronization

- Cost function is equality of two strobes;
- Distance from the rising edge of the early strobe to the falling edge of the late strobe must be equal to the bit duration;
- If rising edges of bit and early strobe coincide, strobe values are equal—correct synchronization;
- If rising edges of bit and early strobe do not coincide, strobe values are different—control sign to move to the correct direction;
- Direction is dependent on the bit value.



Tracking controller

- In the steady-state—no changes;
- Different frequency requires constantly rising/falling delay $\tau[n]$;
- Use PI (proportional and integral) controller;
- TED—time error discriminator;
- PI controller consists of proportional α and integral β branches;
- Intergral accumulator and $\tau[n]$ containin accumulator describe frequency differenc