Principles of Communications

ECS 332

Dr. Prapun Suksompong

prapun@siit.tu.ac.th

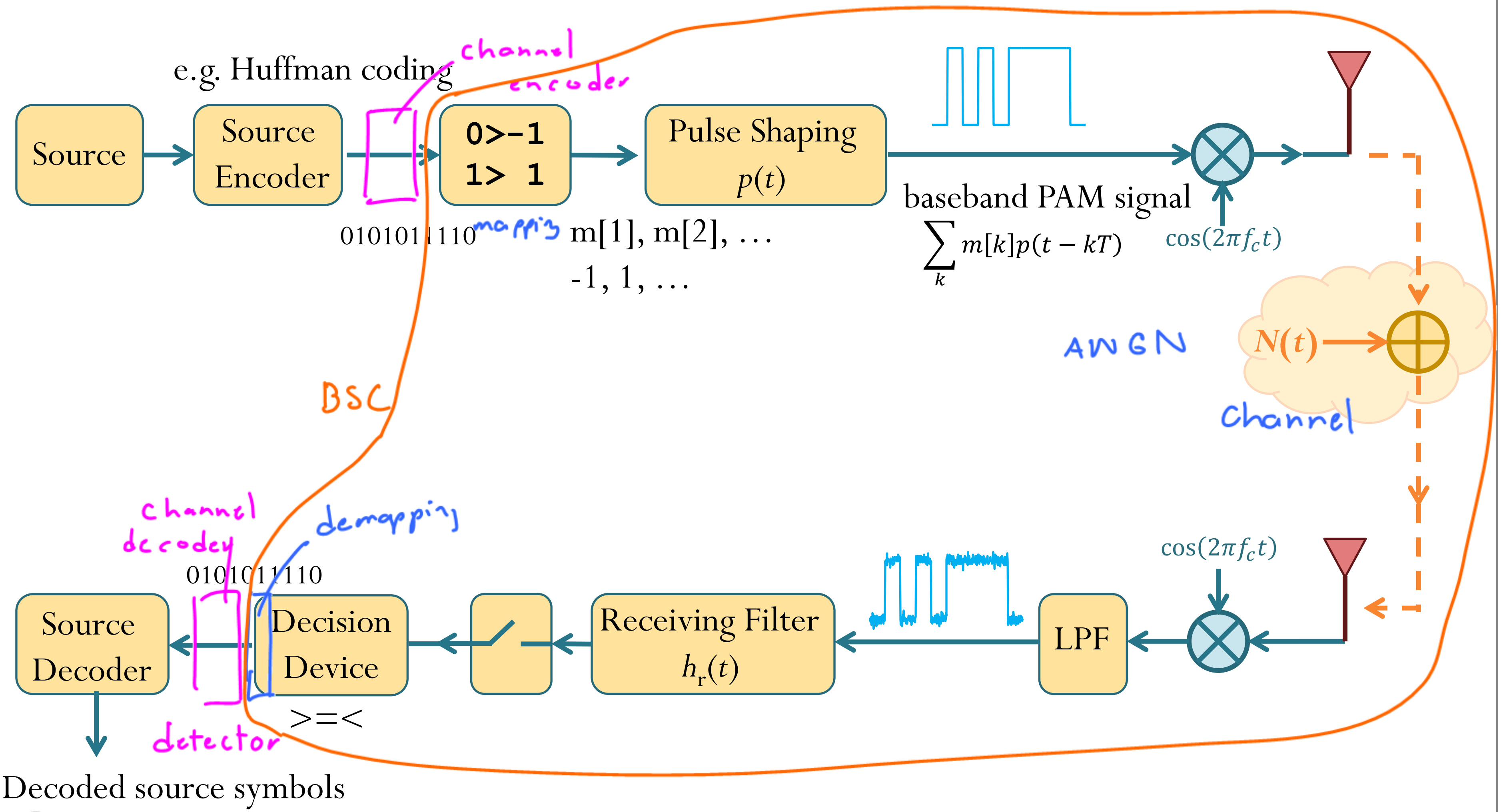
Digital Communication in the Presence of Noise

Office Hours:

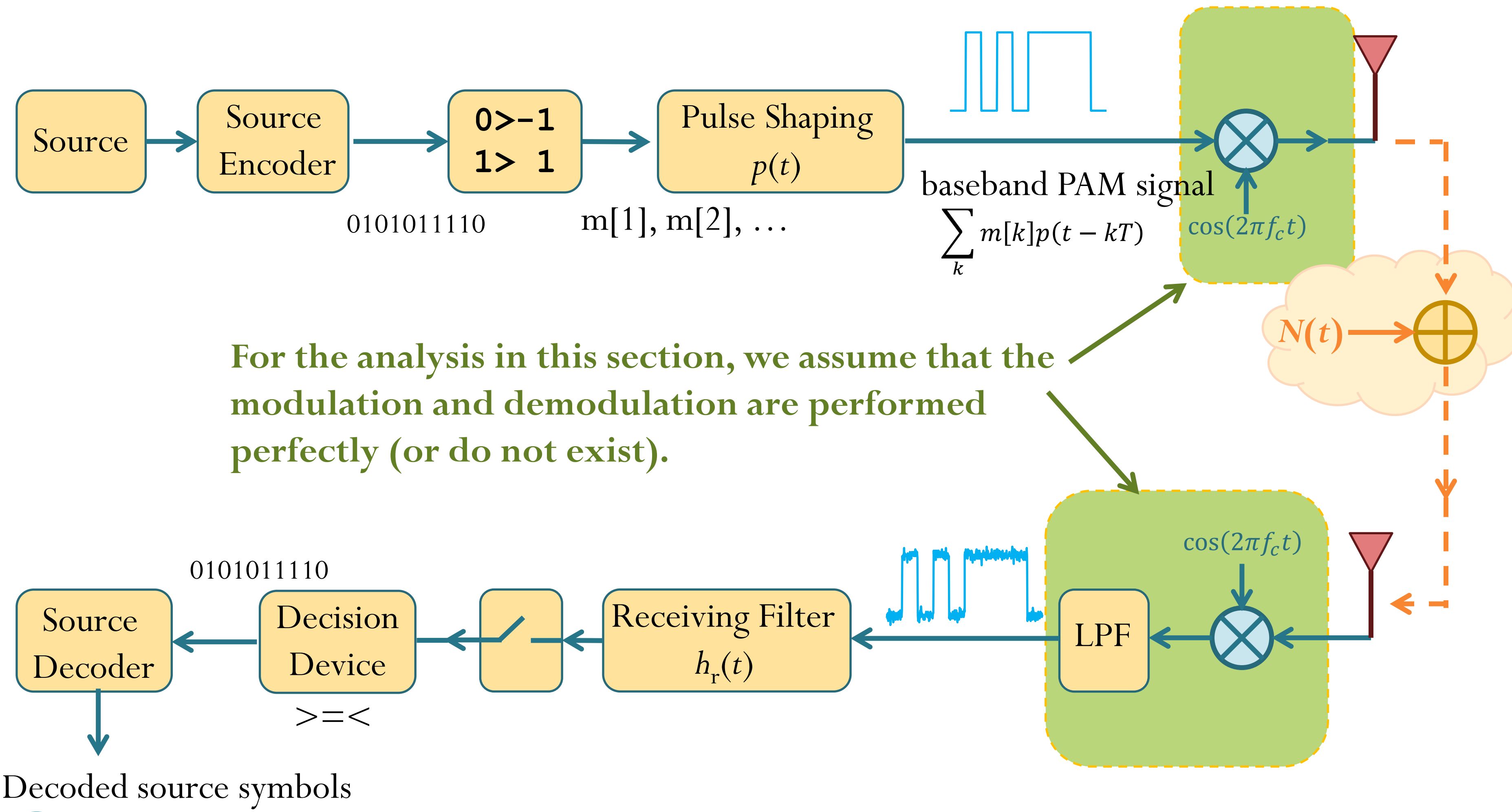
BKD 3601-7

Monday	**14:40-16:00**
Friday	**14:00-16:00**

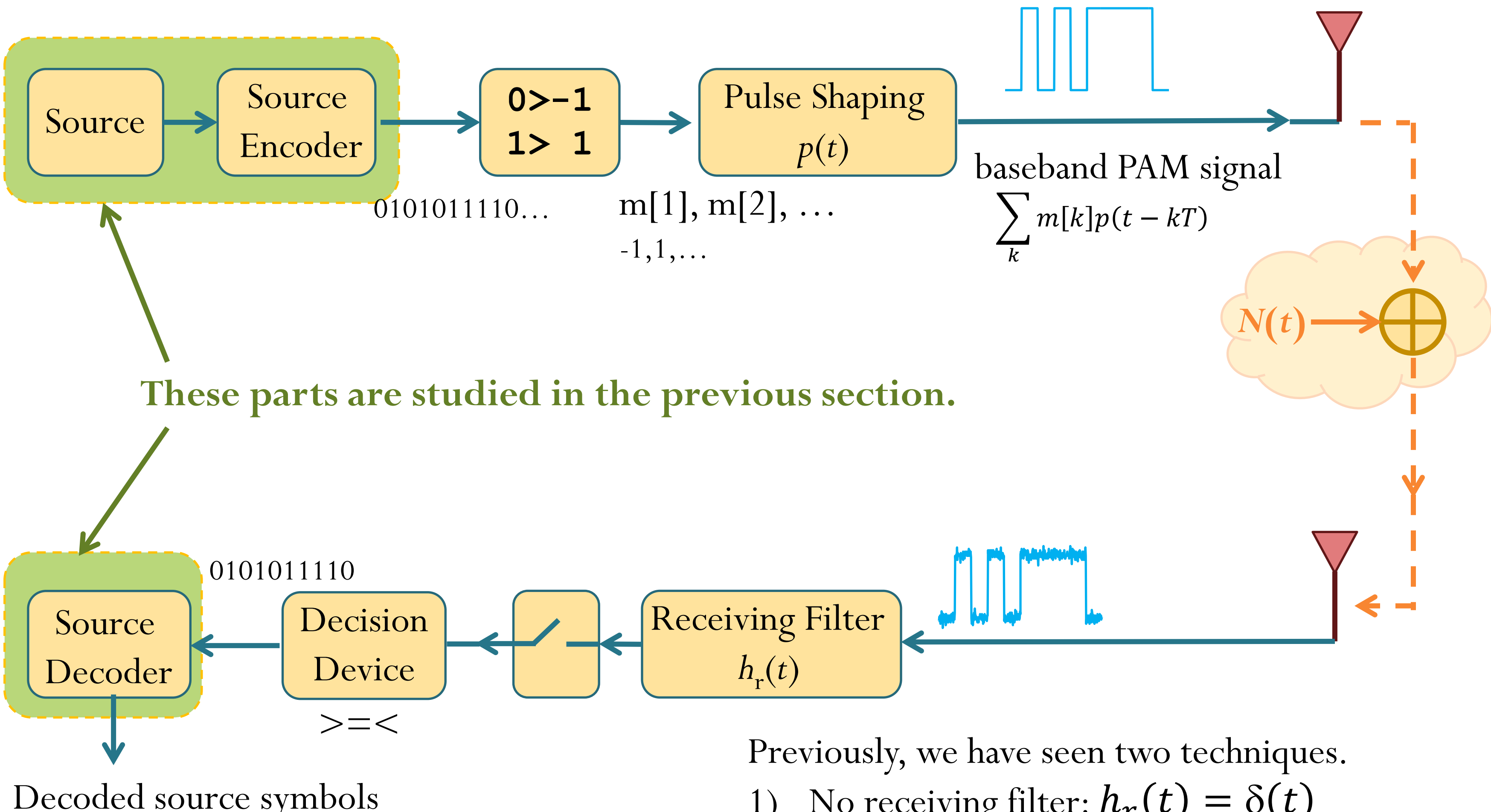
Big Picture (so far)



Big Picture: Assumption



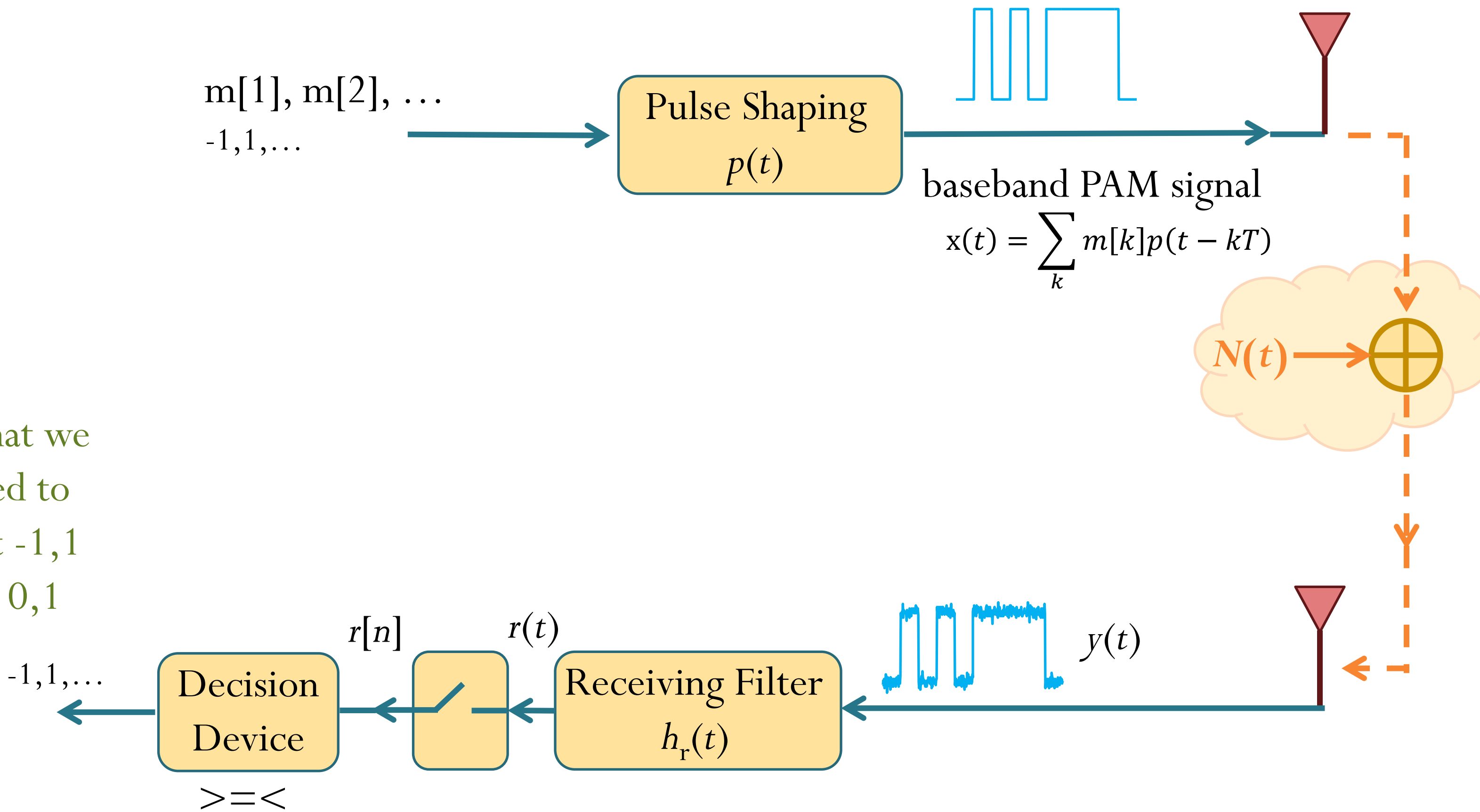
Big Picture: Simplified



Previously, we have seen two techniques.

1) No receiving filter: $h_r(t) = \delta(t)$
2) Matched filter: $h_r(t) = p^*(T - t)$

System under consideration



Note that we still need to convert -1,1 back to 0,1

Previously, we have seen two techniques.

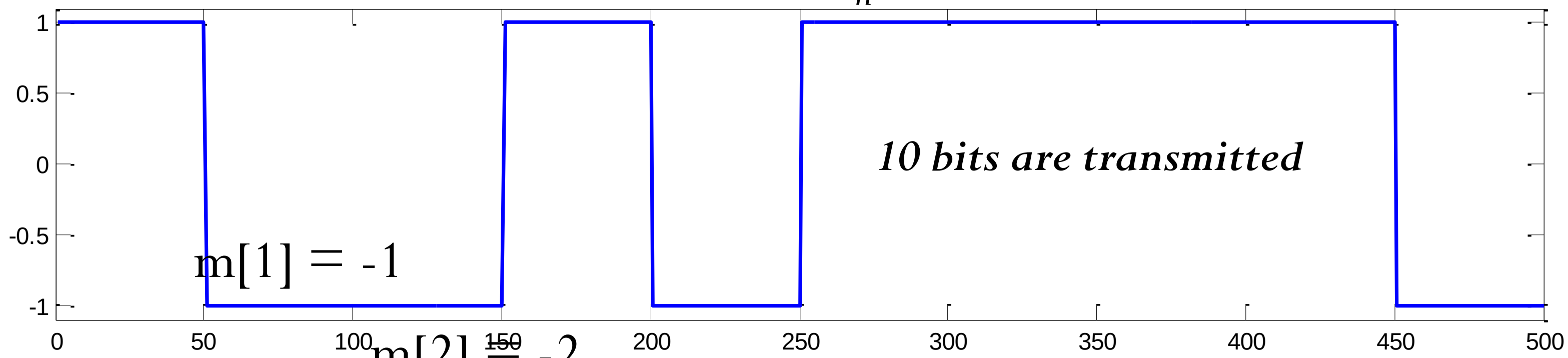
1) No receiving filter: $h_r(t) = \delta(t)$
2) Matched filter: $h_r(t) = p^*(T - t)$

Old Example (1)

$$m[n] \in \underbrace{\{-1, 1\}}_{\mathcal{A}}$$

$$x(t) = \sum_n m[n]\, p(t - nT)$$

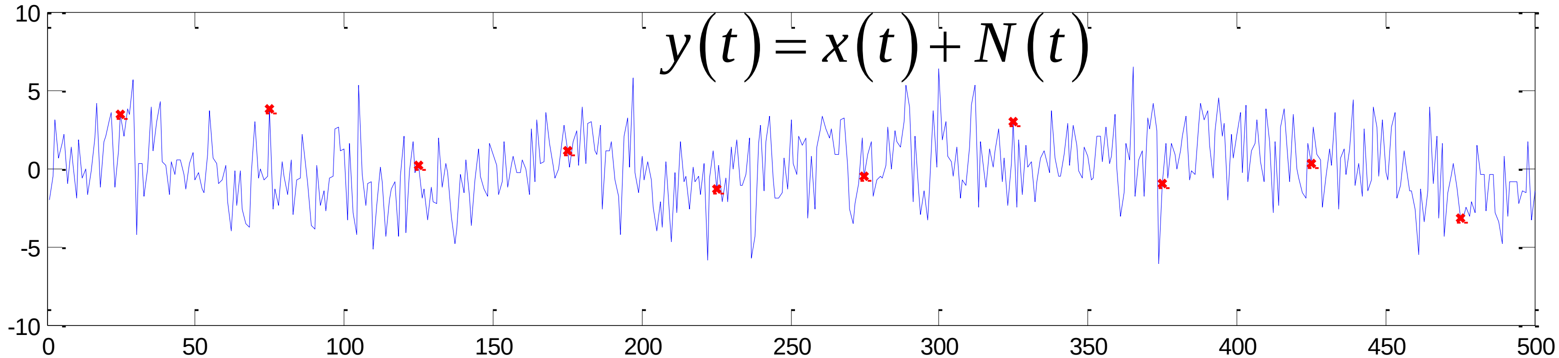
m[0] = 1



m[1] = -1

m[2] = -2

$$y(t) = x(t) + N(t)$$



Technique 1: consider

$$r[n] = y\left(nT + \frac{T}{2}\right)$$

$$r[n] \geq 0 \Rightarrow \text{ decode as } \hat{m}[n] = 1$$

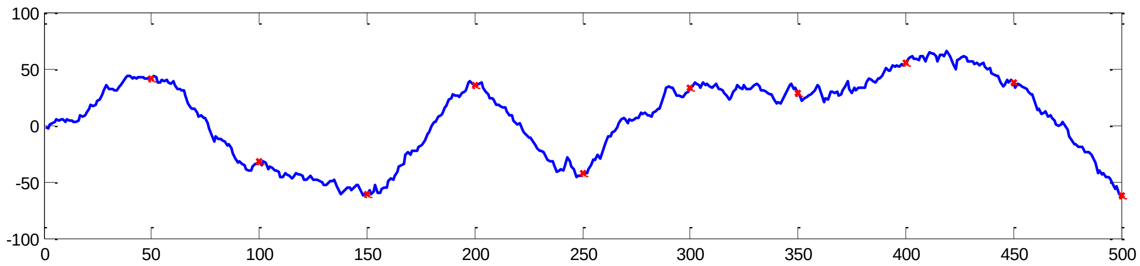
$$r[n] < 0 \Rightarrow \text{ decode as } \hat{m}[n] = -1$$

Old Example (2)

Technique 2: $r(t) = \int_{t-T}^{t} y(\tau)d\tau = y(t) * \underbrace{h_r(t)}$

$h_r(t) = p^*(T-t)$ **Matched filter**

$r(t)$



To decode, consider $r[n] = y(nT + T)$

$$r[n] \geq 0 \Rightarrow \text{ decode as } \hat{m}[n] = 1$$

$$r[n] < 0 \Rightarrow \text{ decode as } \hat{m}[n] = -1$$

BER (Bit Error Rate)

- Technique 1

n	0	1	2	3	4	5	6	7	8	9
$m[n]$	1	-1	-1	1	-1	1	1	1	1	-1
$\hat{m}[n]$	1	1	1	1	-1	-1	1	-1	1	-1

$$BER = \frac{4}{10} = 0.4$$

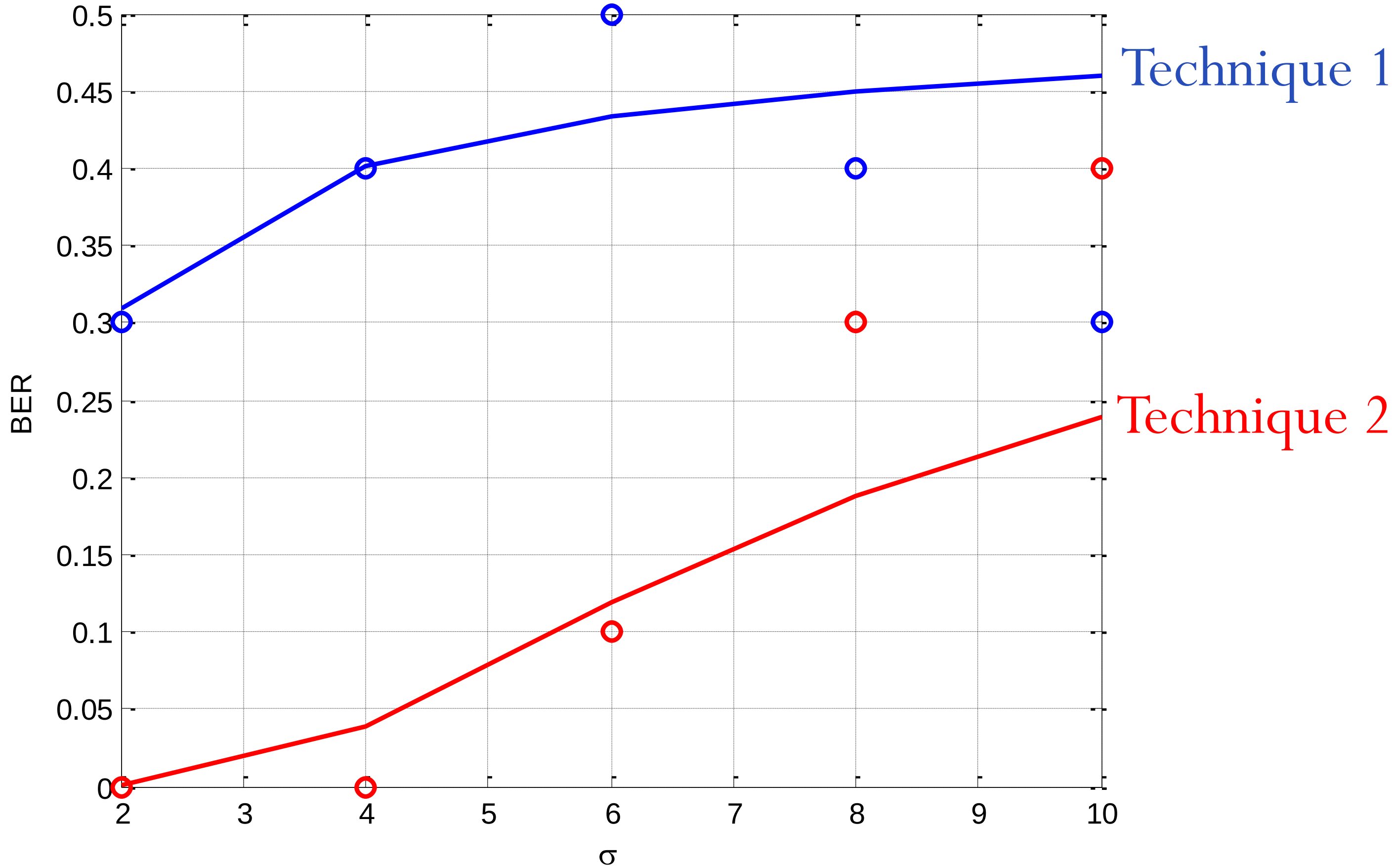
- Technique 2

n	0	1	2	3	4	5	6	7	8	9
$m[n]$	1	-1	-1	1	-1	1	1	1	1	-1
$\hat{m}[n]$	1	-1	-1	1	-1	1	1	1	1	-1

$$BER = \frac{0}{10} = 0$$

BER (Bit Error Rate)

10 bits simulation

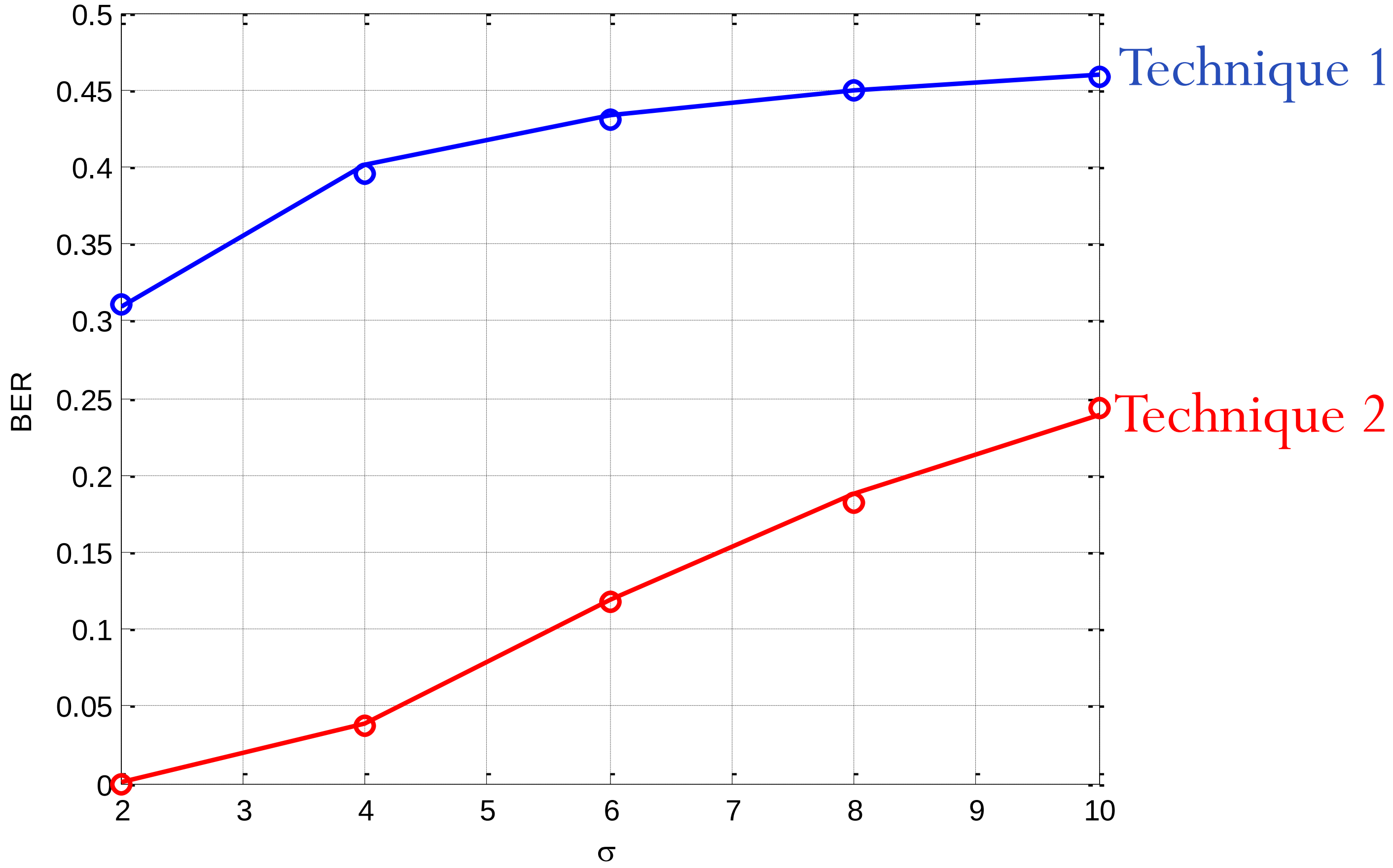


Severity of the noise

[rectPulse_Demo_5_difftPulse_BERCurve.m]

BER (Bit Error Rate)

10,000 bits simulation

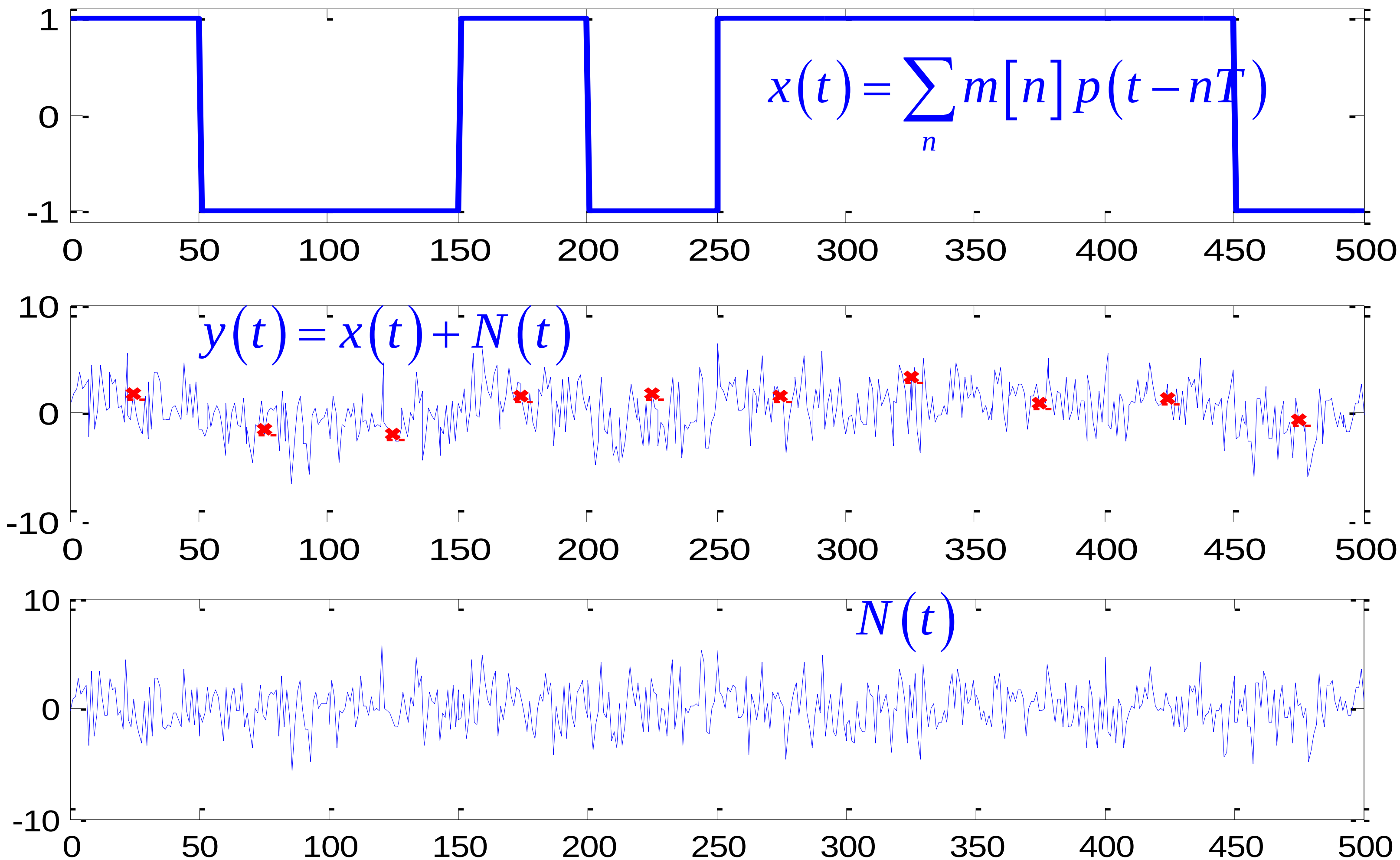


Severity of the noise

[rectPulse_Demo_5_difftPulse_BERCurve.m]

Noise

Note that in MATLAB, the signals are all in discrete-time approximating the actual signals in continuous-time.

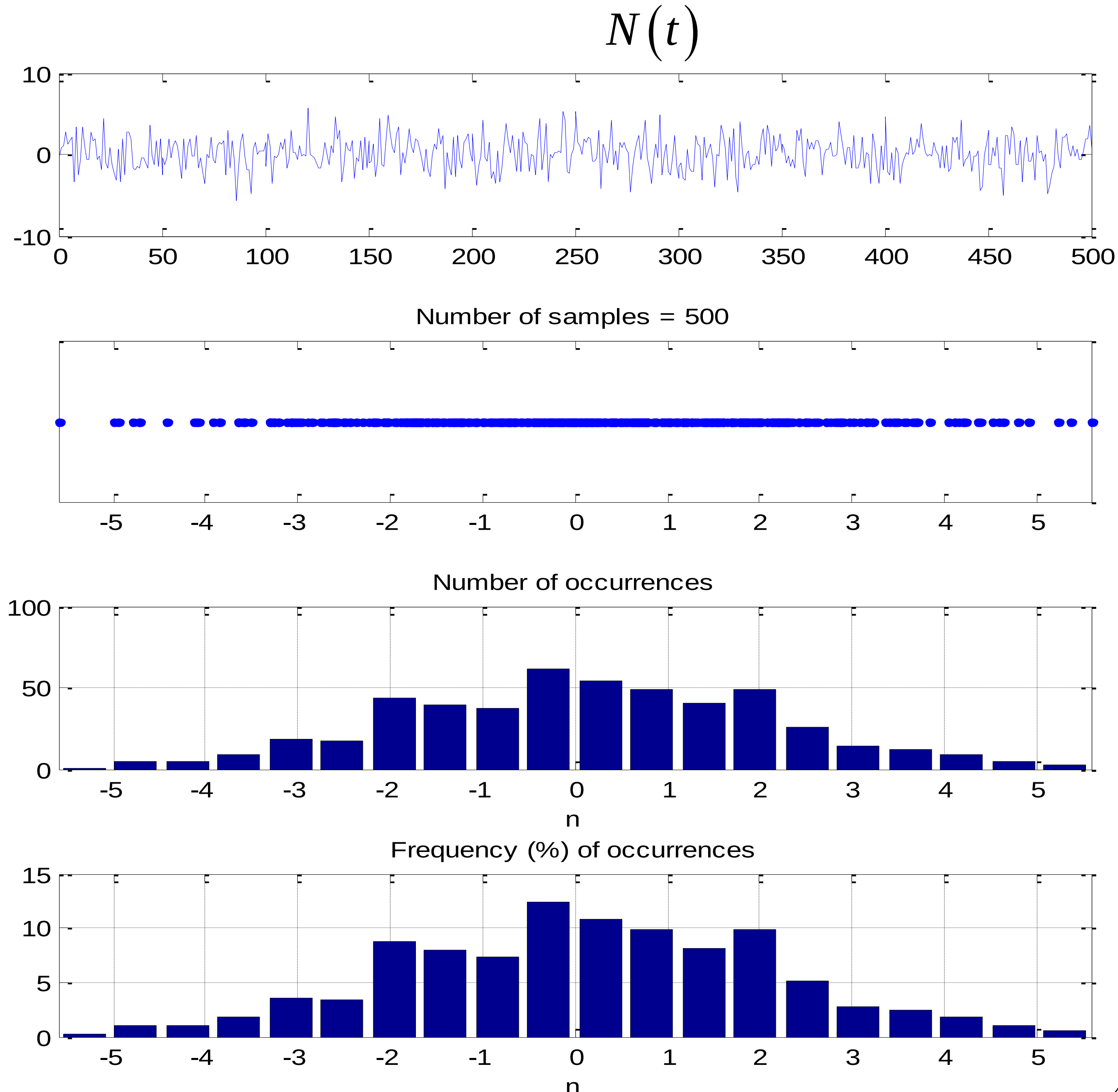


The noise here was generated by running the command `2*randn` 500 times.
(Actually, we use the command `2*randn(1,500)`.

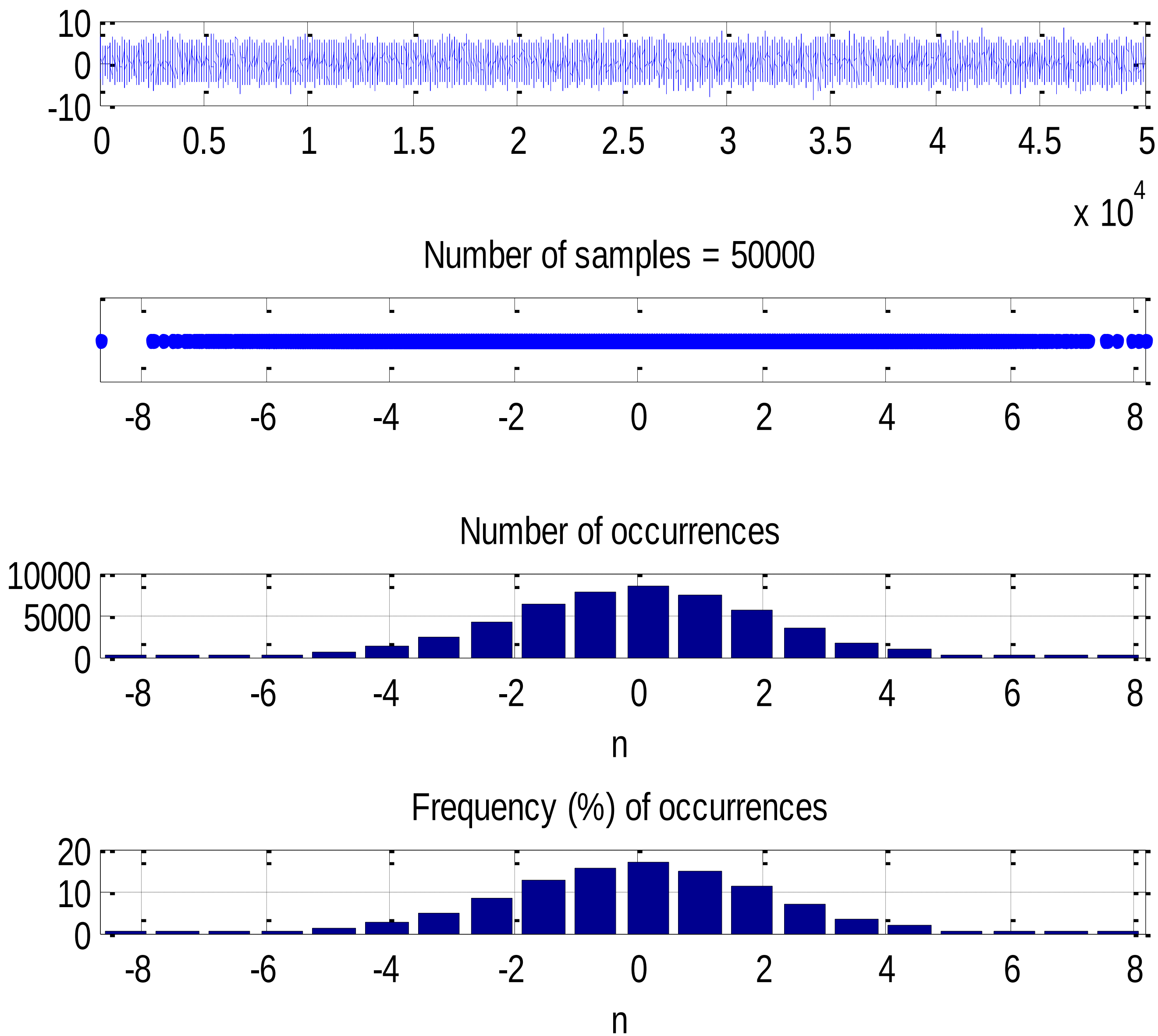
Noise

```
2*randn(1,500)
```

generates 500 i.i.d. Gaussian RVs. These random variables have expected value = 0 and std = 2.



Noise



`2*randn(1,5e4)`

generates 50,000 i.i.d. Gaussian RVs. These random variables have expected value = 0 and std = 2.

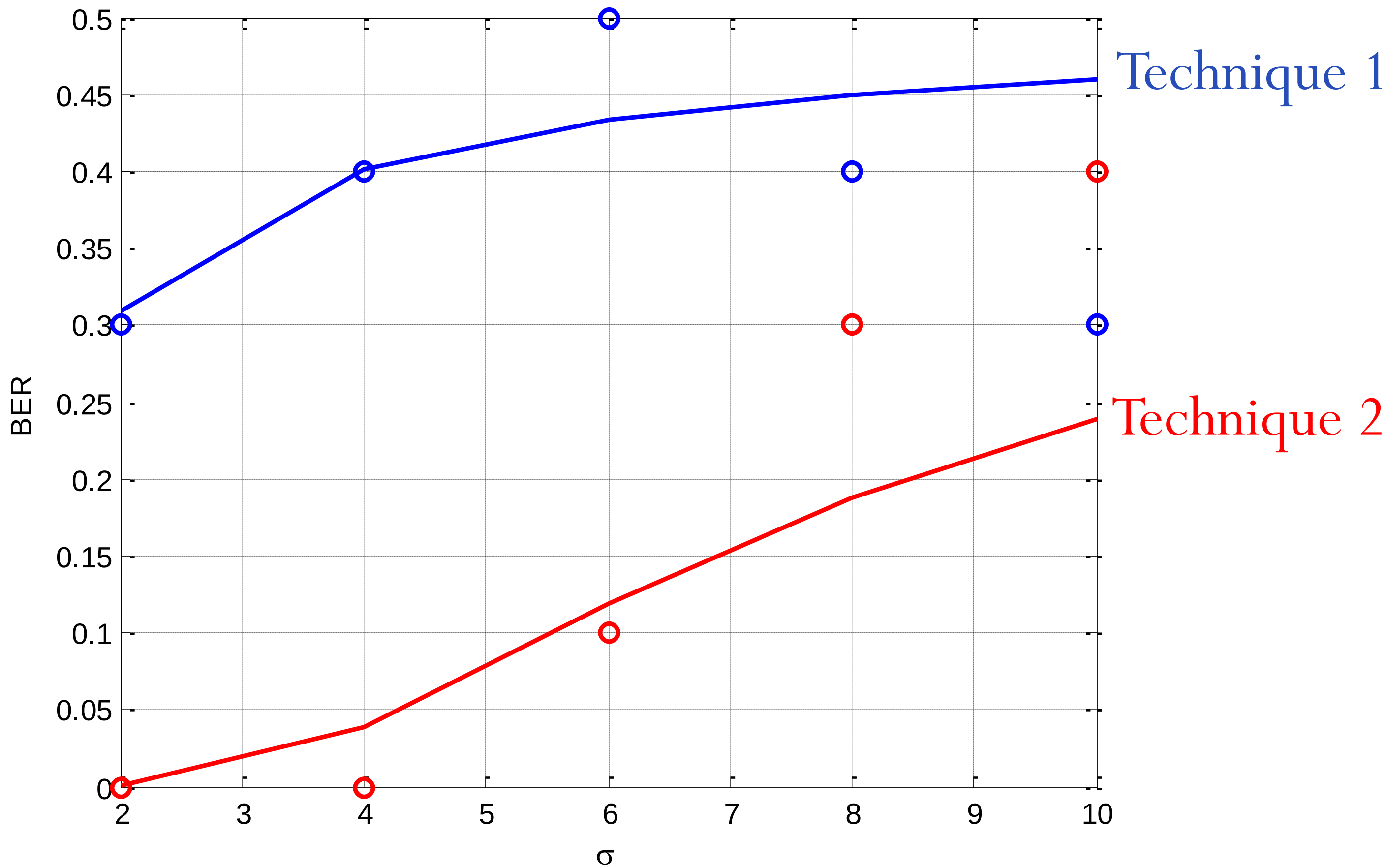
Describing the noise

$$N \sim \mathcal{N}\left(0, \sigma^2\right)$$

$$f_N\left(n\right) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2}$$

BER (Bit Error Rate)

10 bits simulation

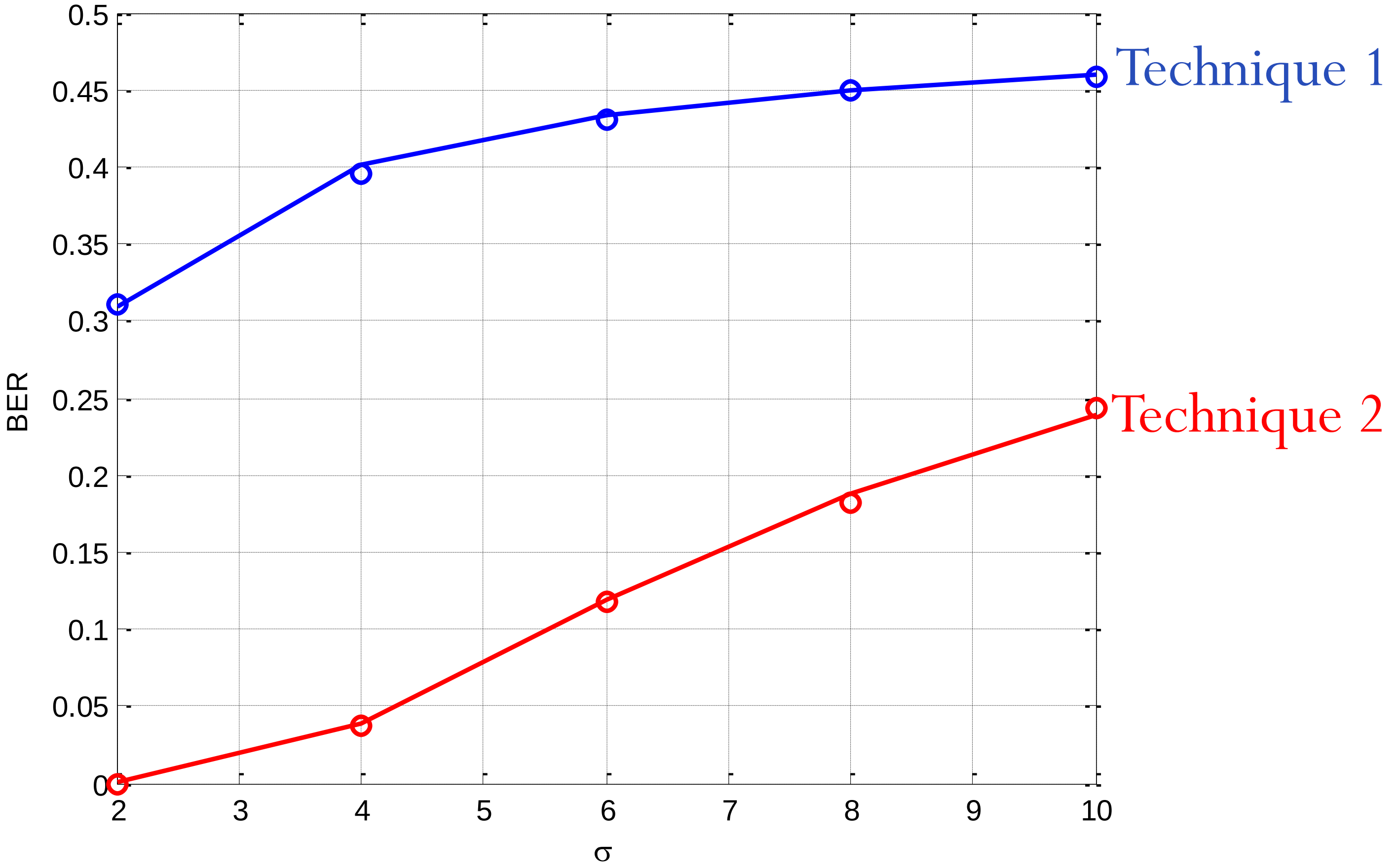


Severity of the noise

[rectPulse_Demo_5_difftPulse_BERCurve.m]

BER (Bit Error Rate)

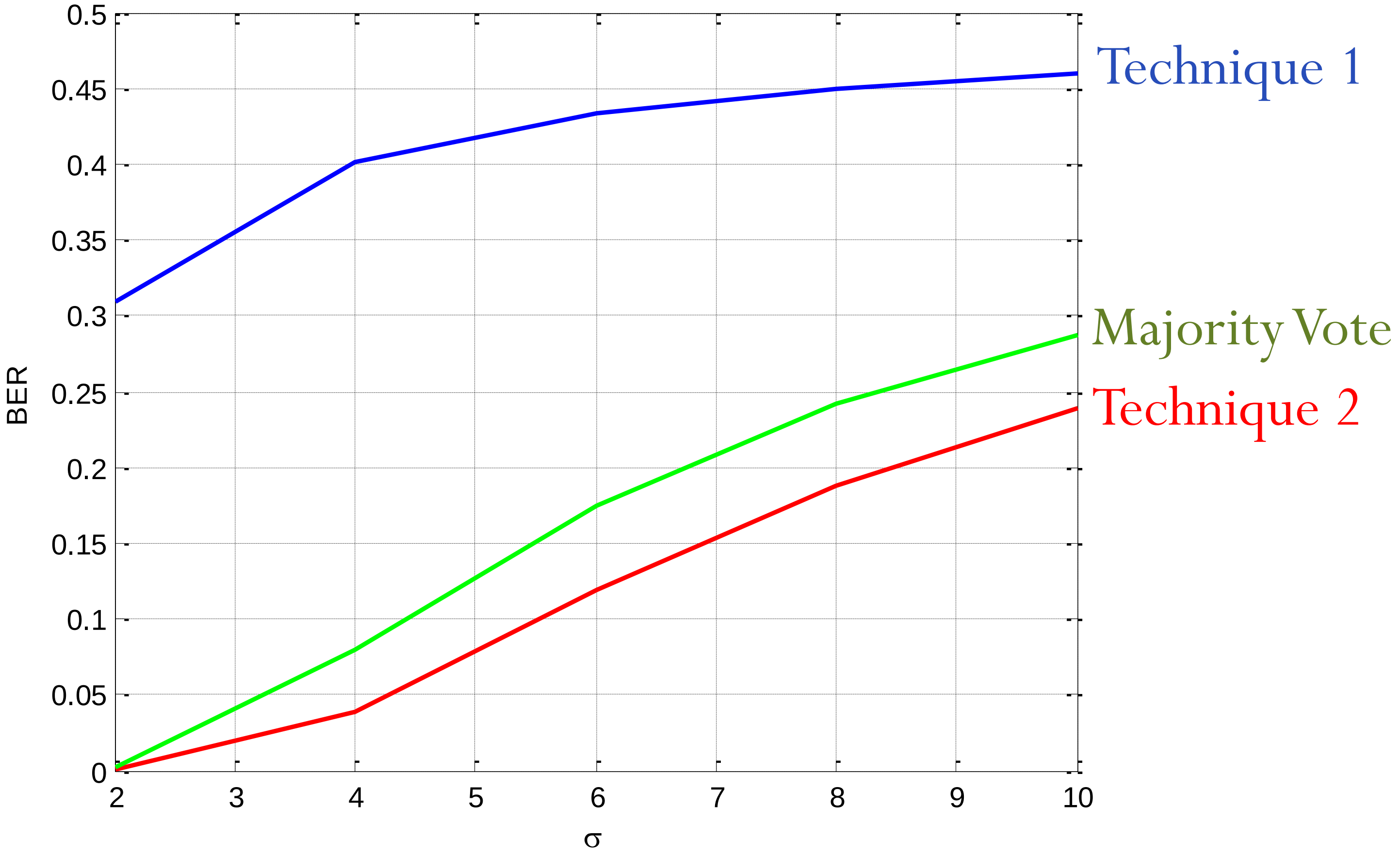
10,000 bits simulation



Severity of the noise

[rectPulse_Demo_5_difftPulse_BERCurve.m]

BER (Bit Error Rate)



[rectPulse_Demo_3_Maj.m]