# Questions

Any questions to exercises and homeworks from last time?

# Table of Content

# Correlation

- In signal processing, cross-correlation is a method for searching for a short (known) signal in a longer (measured) signal, by calculating the sliding dot product.
- The correlation is closely related to the convolution (just a sign for mirroring the signal is different) and matched filtering (filter response can be calculated by cross-correlation).
- Auto-correlation (correlate a signal with itself) can be used to assess the “ideal” matched filter response.
- Numpy’s implementation: `numpy.correlate(a, v, mode='valid')` calculates

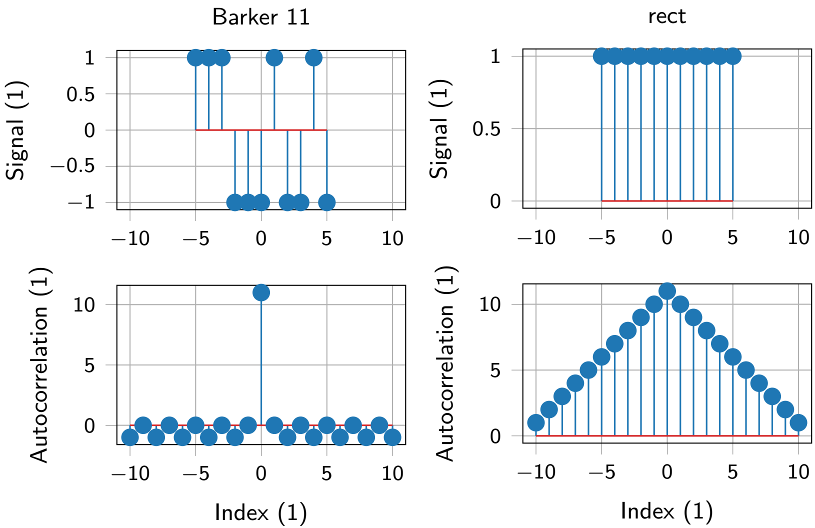
  $$c_{av}[k] = \sum_n a[n+k] \cdot conj(v[n]))$$

  with zero-padding when necessary or requested by optional parameter `mode`. For details on `mode`, see help of `np.convolution`.

## Exercise: Auto-correlation

- Plot the auto-correlation of the Barker 11 sequence
  `barker_11=np.array([+1, +1, +1, -1, -1, -1, +1, -1, -1, +1, -1])`
  and the a rect function of equal length. Both sequences are considered as time series centered at $t = 0$.
- Where do the main peaks in the auto-correlation occur?
- Explain in own words:
  - What are the main differences of the shape of the auto-correlation?
  - How small/high must the rect be to produce a similar main peak as the Barker code?

# Solution: Auto-correlation of Barker and Rect of Length 11



## Homework: Cross-Correlation to Detect a Signal

- Consider a rect of length 11 as test transmit signal.
- Generate a receive signal consisting of: 20 zeros, followed by the test signal, by 39 zeros, by the negative test signal, and by another 20 zeros.
  - Hint: have a look at `concatenate` and `zeros` from the `numpy` module.
- Plot the correlation of the receive signal with the test signal.
  - How are the peaks related to the auto-correlation of the test-signal?
  - Where do the peaks occur?
  - Whats the influence of the negative sign?

# Pseudo-random Numbers

- Large quantities of random numbers are often required in scientific computing, e.g., for Monte-Carlo simulation or generating noise for algorithm testing.
  - True random numbers are hard (and slow) to generate.
  - Pseudo-random numbers are often generated using linear-feedback shift registers.
- Reproducibility
  - Reproducibility is crucial for debugging.
  - Setting the seed means initializing the shift register. Subsequent drawn pseudo-random number will follow a given sequence.
- Often only normalized uniform `rand` and normalized Gaussian `randn` distributed samples are directly available. Numpy provides also `random.normal`, and `random.uniform`.

# Generating Gaussian Distributed Samples with Numpy

- `np.random.randn` generates pseudo-random samples with standard normal distribution $\mathcal{N}(0,1)$
- $x \sim \mathcal{N}(\mu, \sigma^2)$ and $y = a\,x + b$ results in $y \sim \mathcal{N}(a\mu + b, a^2\sigma^2)$
- Example: for generating 4096 samples with $z \sim \mathcal{N}(0,4)$

```
import numpy as np
import matplotlib.pyplot as plt
N=4096    # number of samples
sigma=2.0 # Note: it's the standard deviation here
z=sigma*np.random.randn(N)
plt.hist(z, bins=int(np.sqrt(N)))
plt.xlabel('Values (1)')
plt.ylabel('Frequency (1)')
```

- To get reproducible results, the seed of the random number generator can be reset

```
np.random.seed(0) # any integer number will do
```

# Noise Power vs. Variance

- For Gaussian noise: variance equals average energy *per sample*

$$\sigma^2 = \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} |x[n]|^2 \quad \text{for} \quad x \sim \mathcal{N}\left(0, \sigma^2\right)$$

- Compare with energy of a *signal*

$$E_{\text{signal}} = \sum_{n} |x[n]|^2$$

- More samples (e.g. increase sampling rate) of equal variance will increase signal's energy.
- That is typically not whats happening in a real measurement.
- Be careful when modeling a real noise source, as power of those is given in noise *density*.

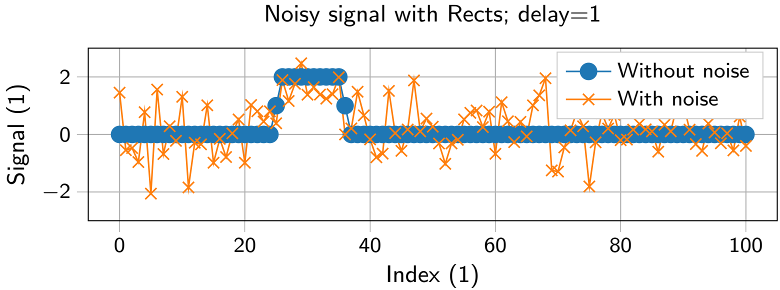
## Homework: Cross-Correlation to Detect a Signal in Noise

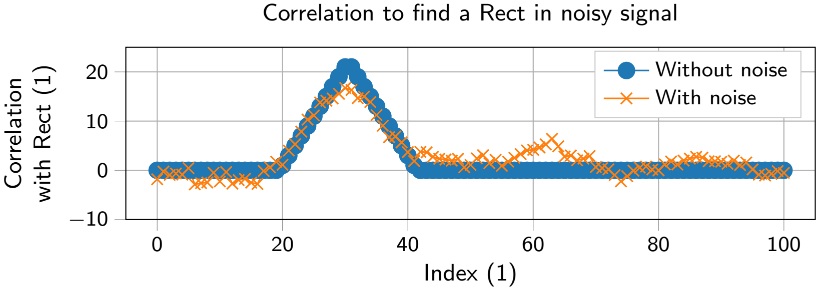
- Add Gaussian noise with zero mean and variance $\sigma^2 = 0.8$ to the receive signal from the previous cross-correlation exercise.
- Plot the receive signal with noise.
  - Is the rect clearly visible in the time domain?
  - How does the signal change compared to the noise free case and over different noise realization (use different seeds)?
- Correlate the noisy receive signal with the transmit signal.
  - Are the correlation peaks clearly visible?

## Homework: Using the Barker 11 Sequence

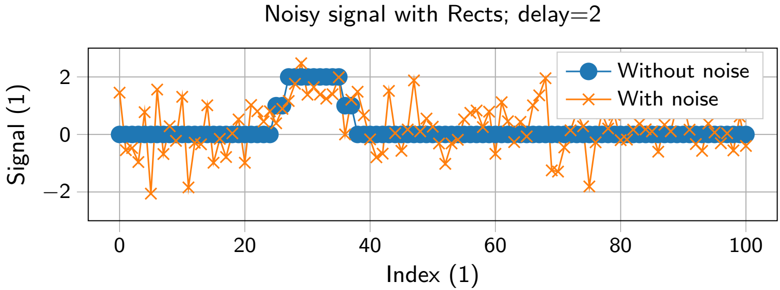
- Redo the previous two correlation exercises using the Barker 11 sequence as test signal instead of the rect.
  - Is the Barker 11 sequence or the rect better suited to locate the test signal in a noisy signal?
  - Is the location of the correlation peak of a the Barker 11 or the rect more effected by noise?

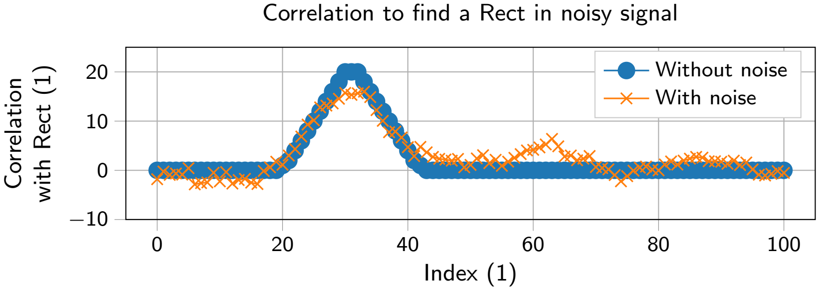
# Partially Overlapping Rects of Length 11



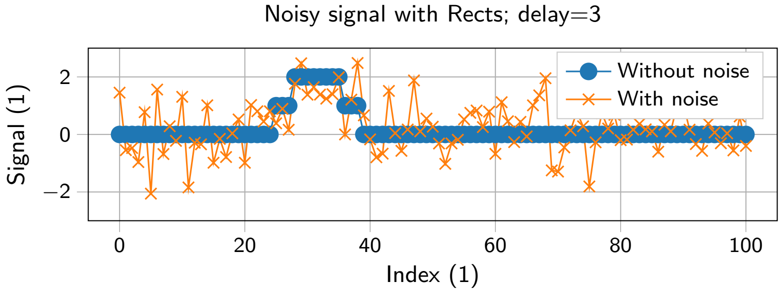



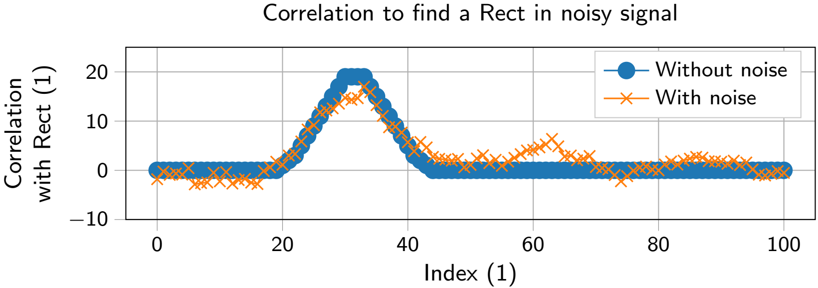
# Partially Overlapping Rects of Length 11



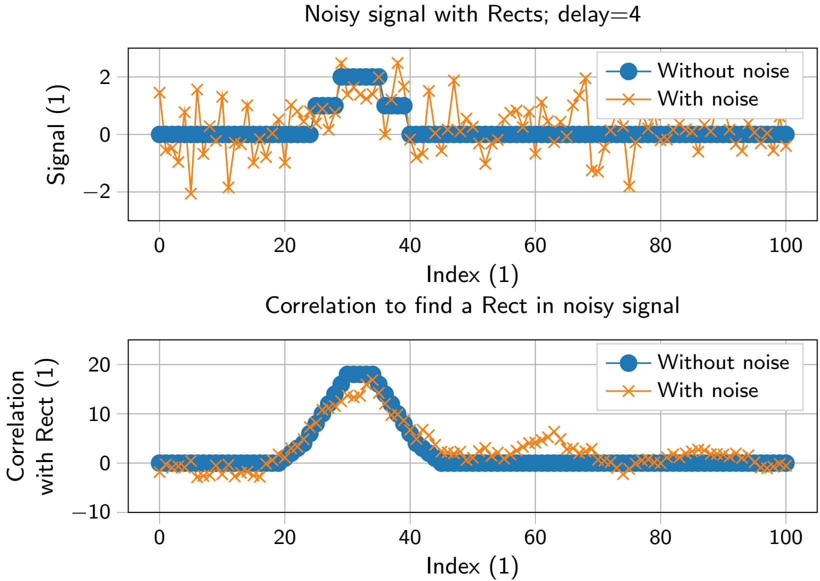



# Partially Overlapping Rects of Length 11

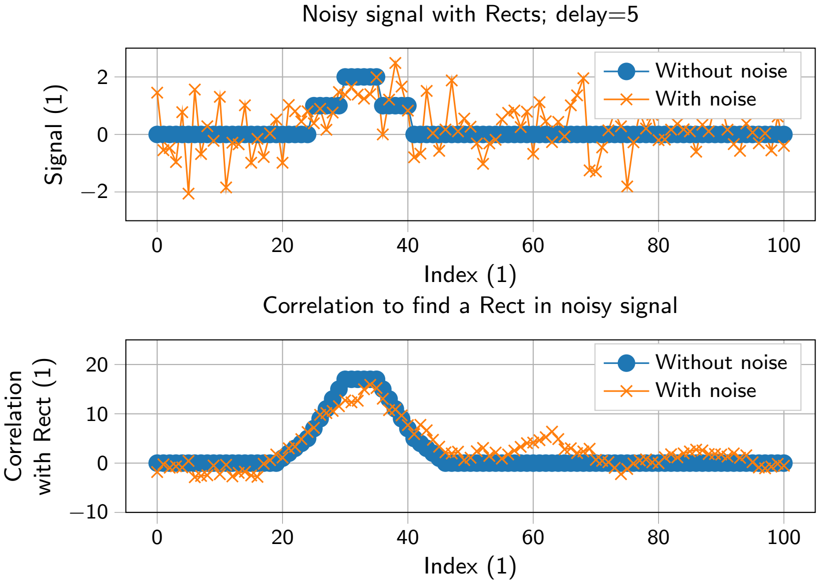


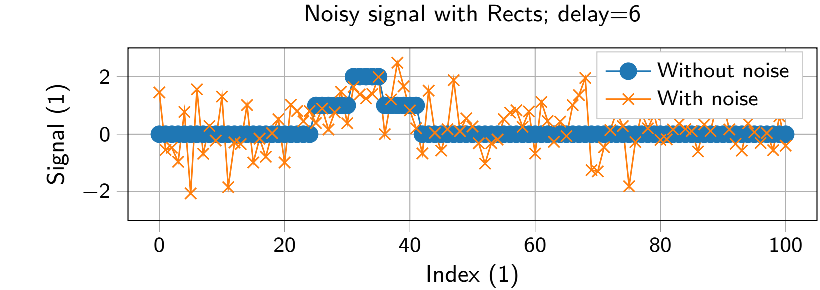
# Partially Overlapping Rects of Length 11

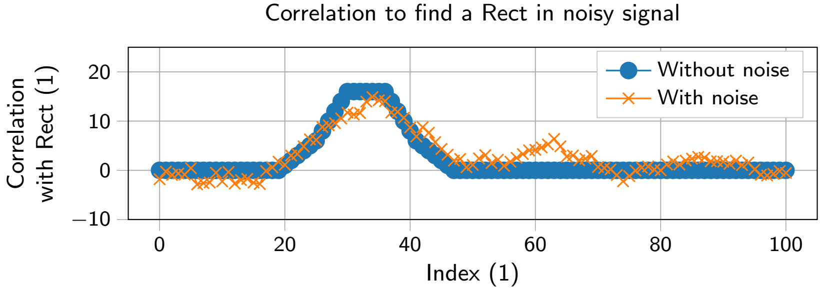


# Partially Overlapping Rects of Length 11

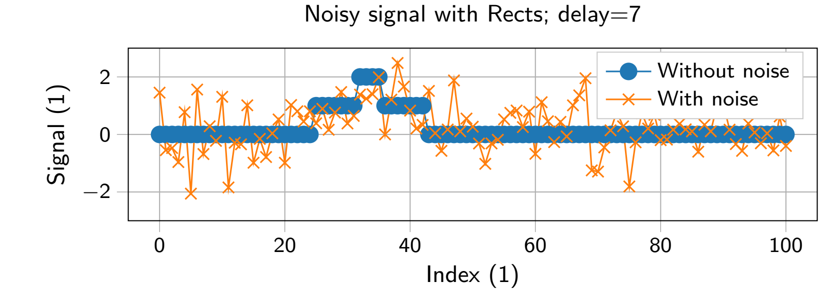


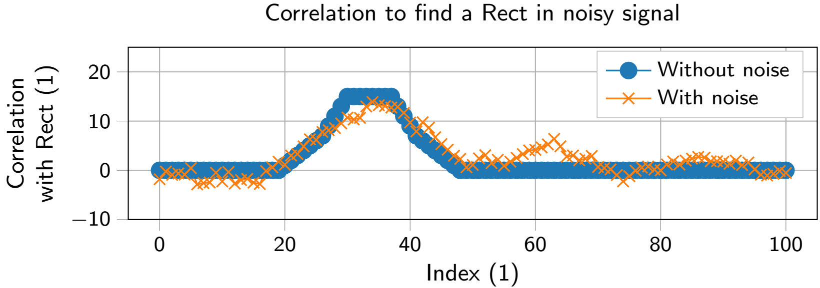
# Partially Overlapping Rects of Length 11



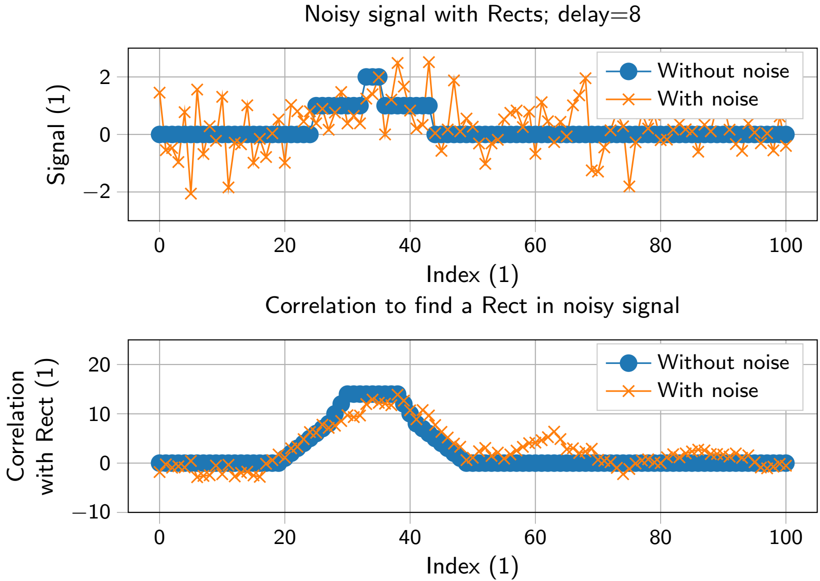



# Partially Overlapping Rects of Length 11

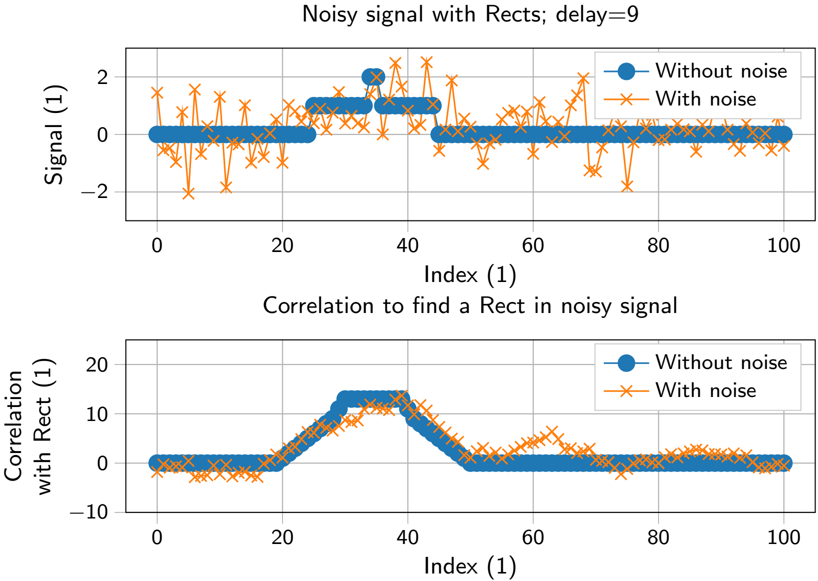


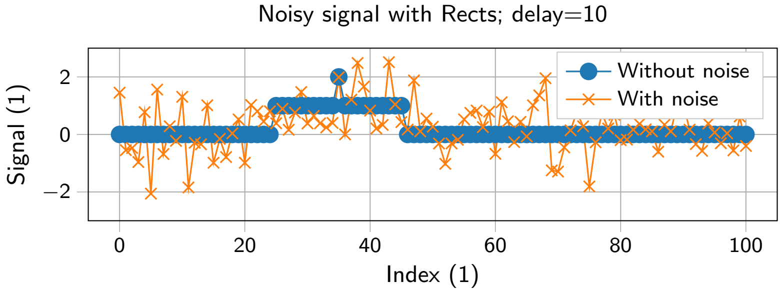
# Partially Overlapping Rects of Length 11

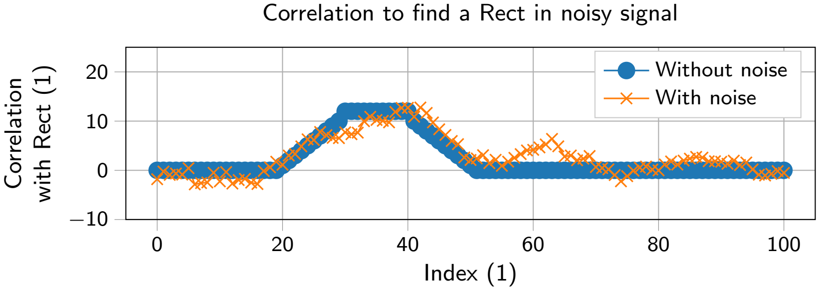


# Partially Overlapping Rects of Length 11

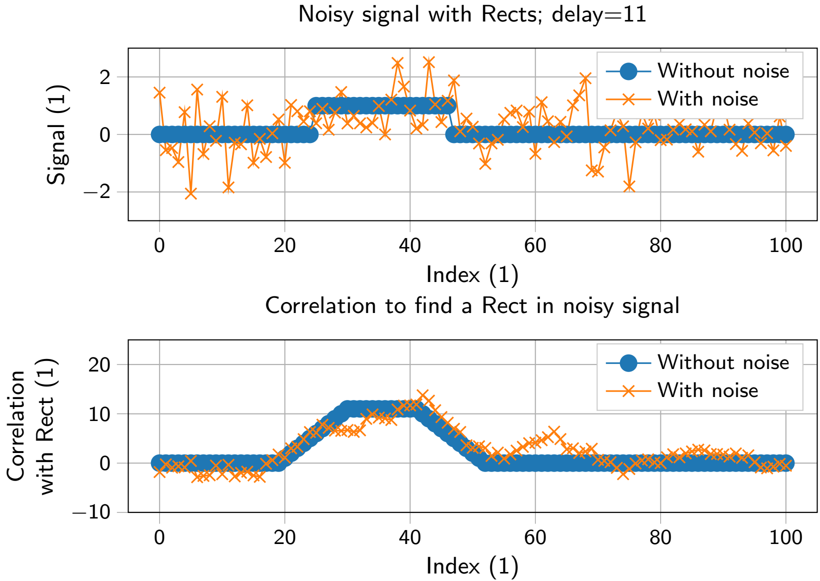


# Partially Overlapping Rects of Length 11

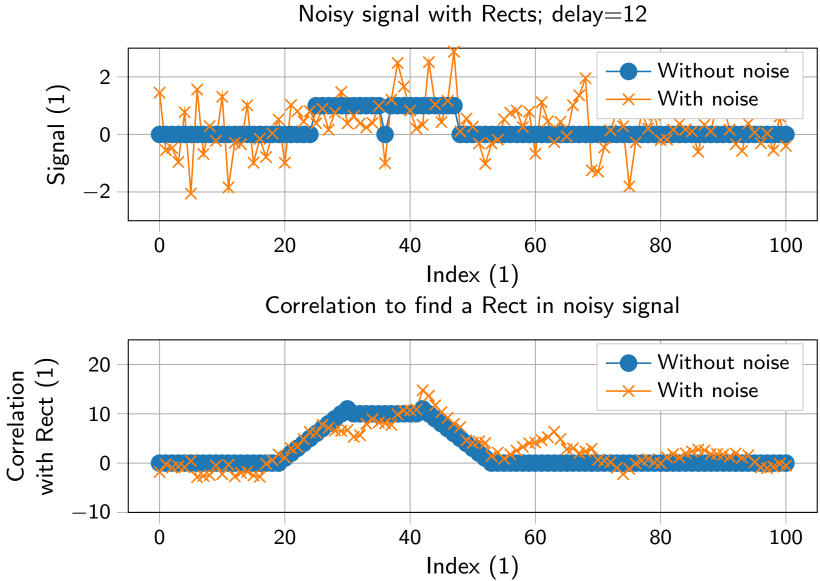


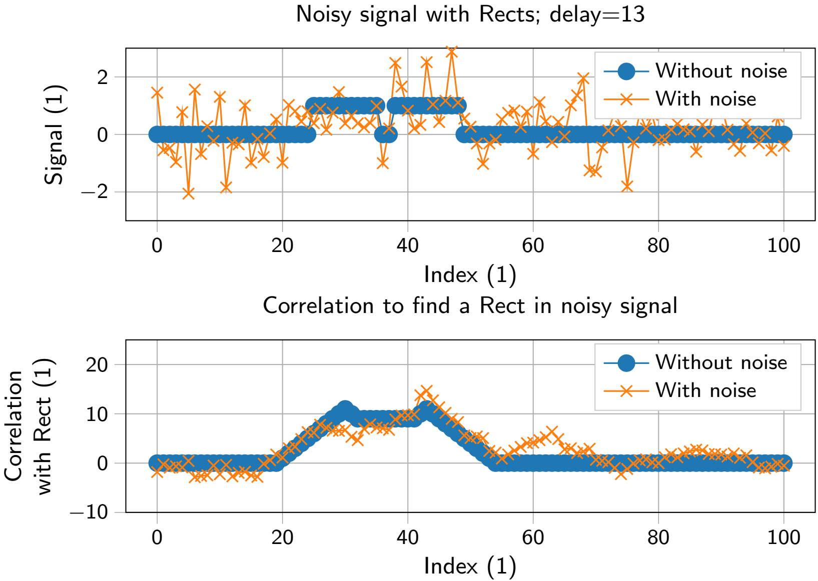
# Partially Overlapping Rects of Length 11



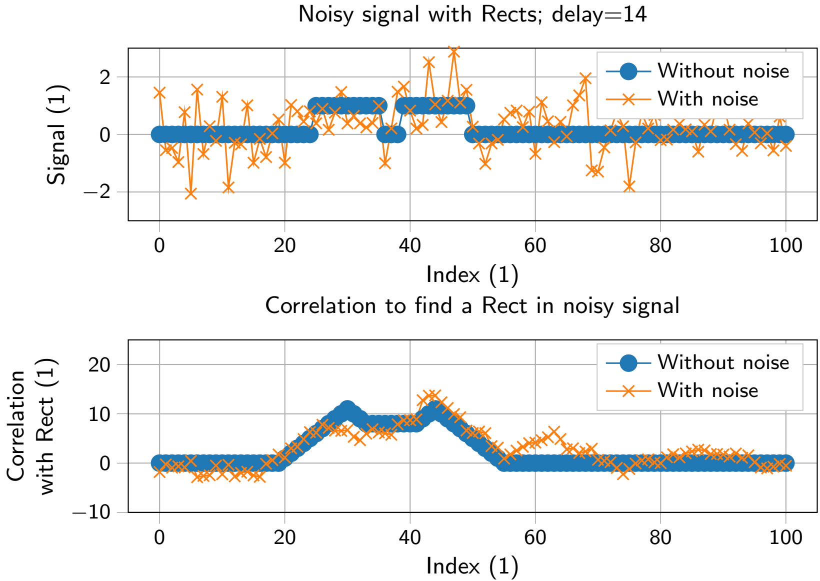
# Partially Overlapping Rects of Length 11



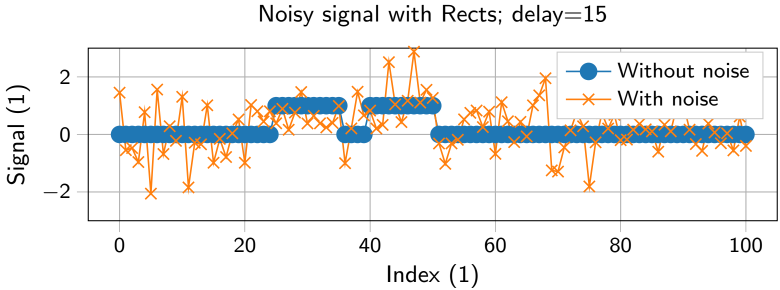
# Partially Overlapping Rects of Length 11

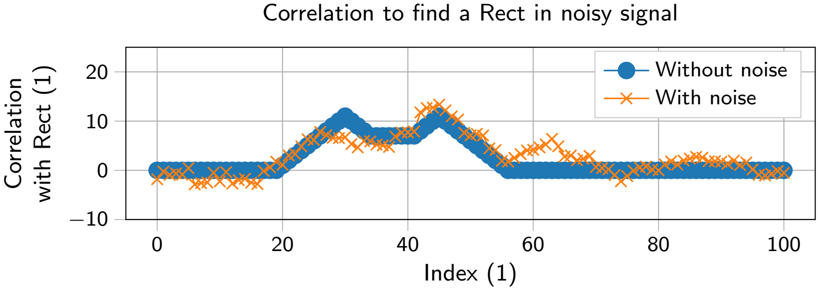


# Partially Overlapping Rects of Length 11

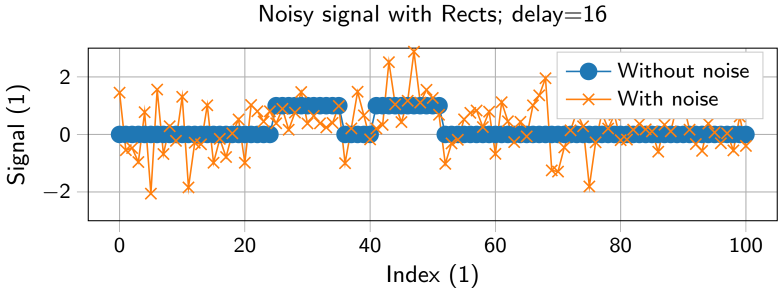


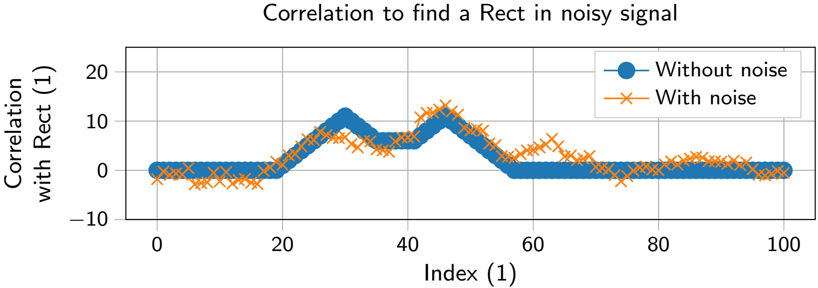
# Partially Overlapping Rects of Length 11



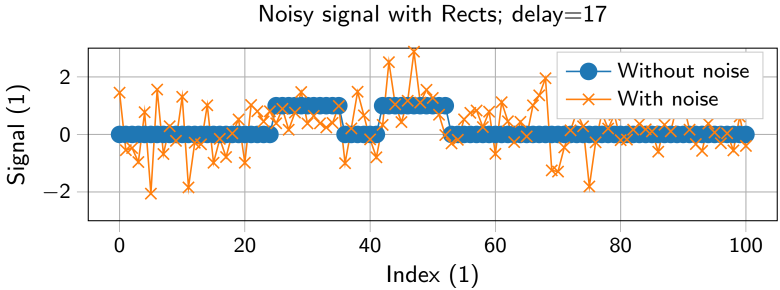



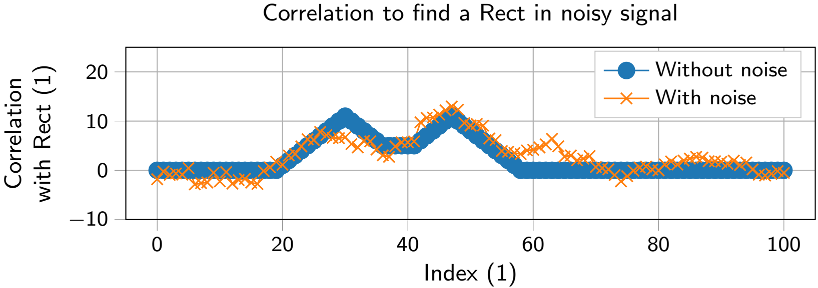
# Partially Overlapping Rects of Length 11



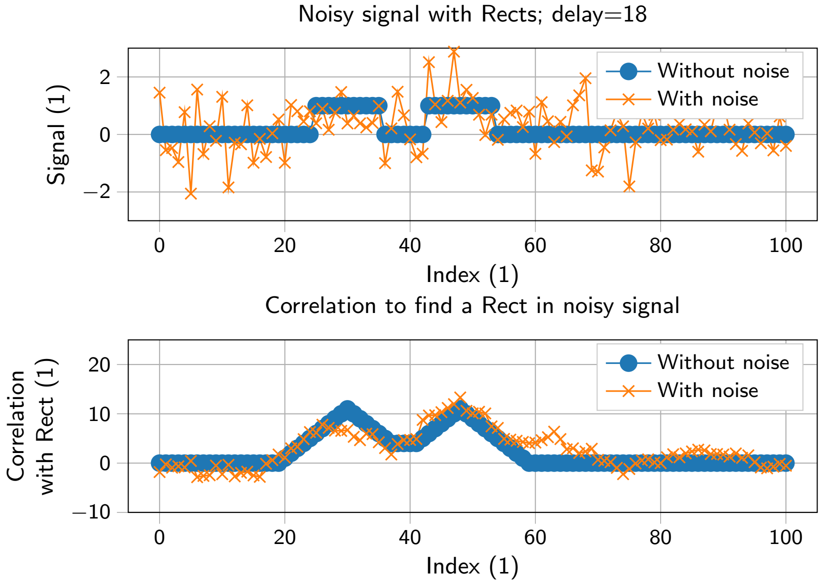



# Partially Overlapping Rects of Length 11

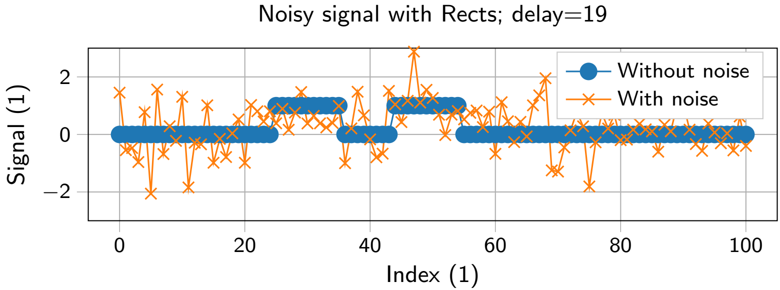


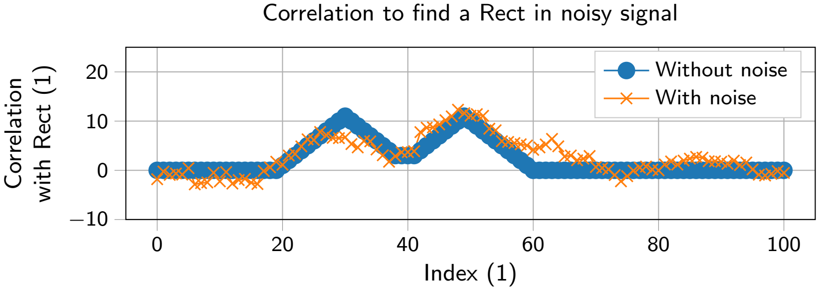



# Partially Overlapping Rects of Length 11

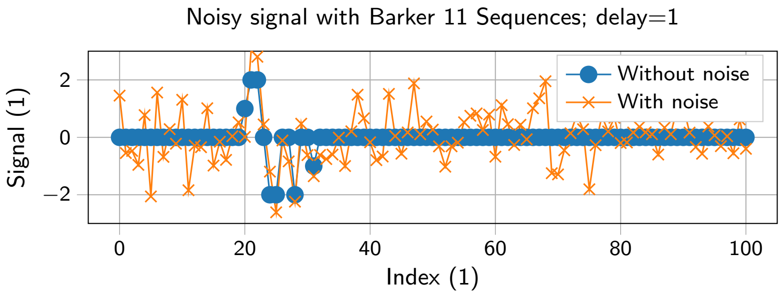


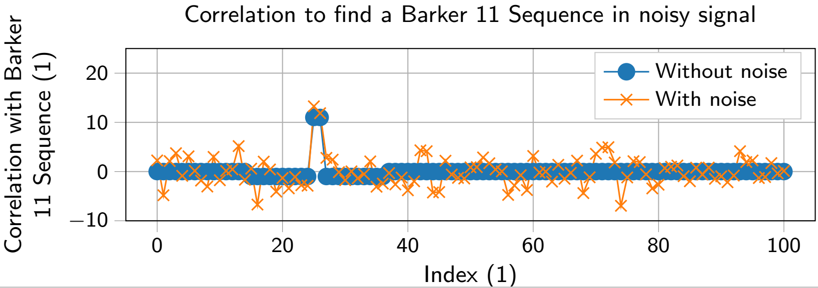
# Partially Overlapping Rects of Length 11



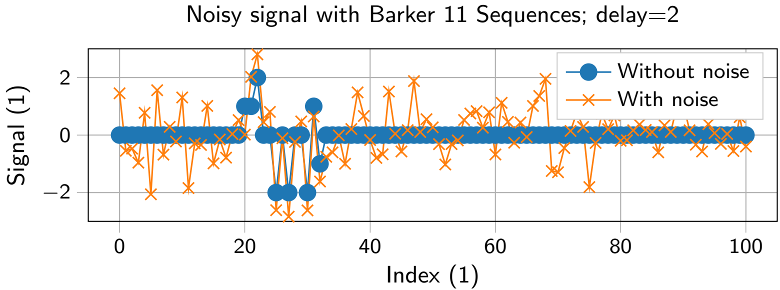



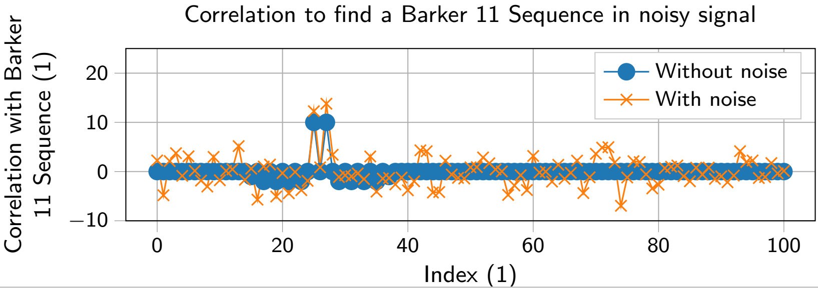
# Partially Overlapping Barker 11 Sequences



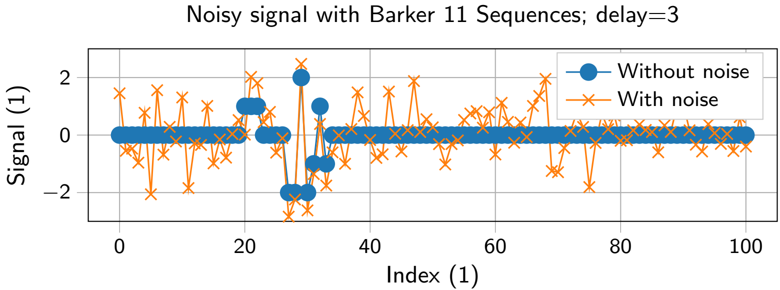



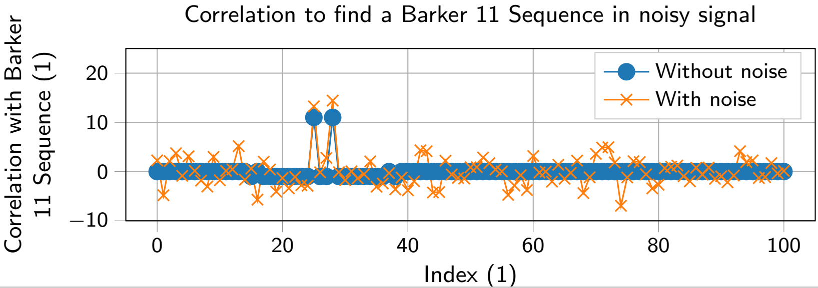
# Partially Overlapping Barker 11 Sequences



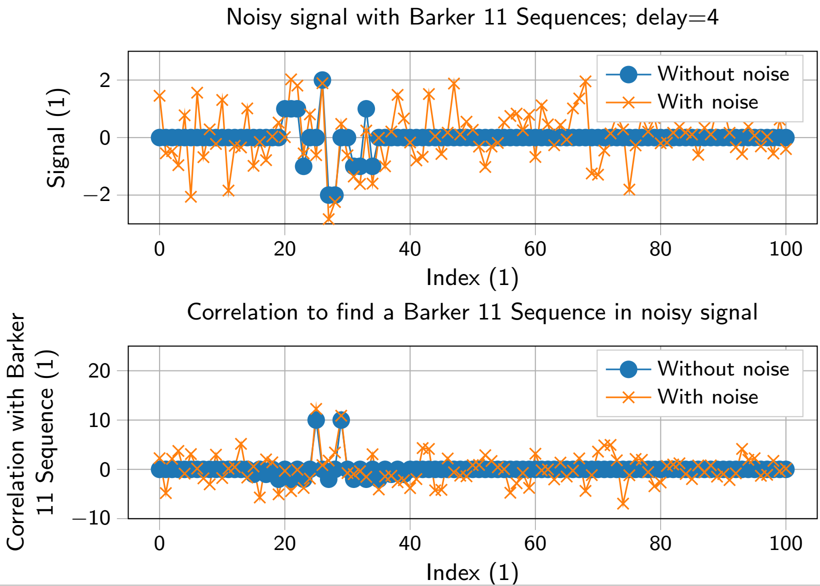



# Partially Overlapping Barker 11 Sequences

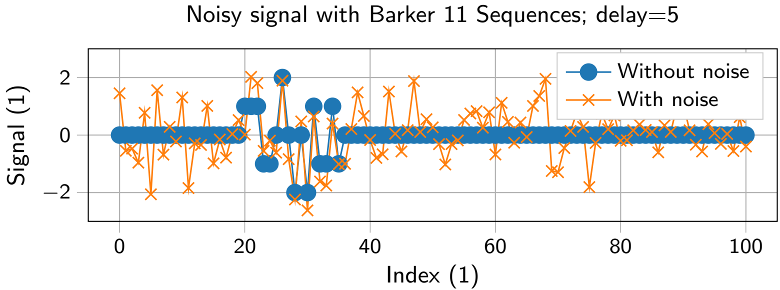


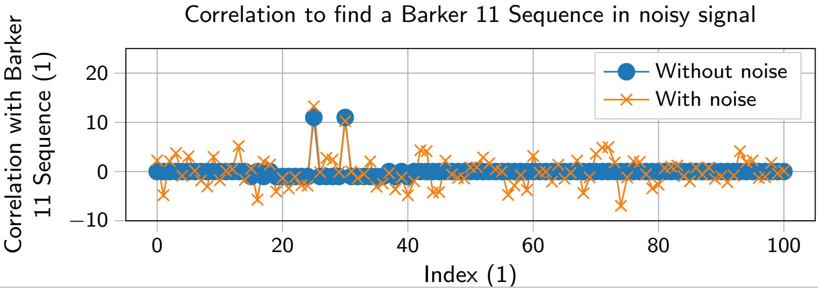



# Partially Overlapping Barker 11 Sequences

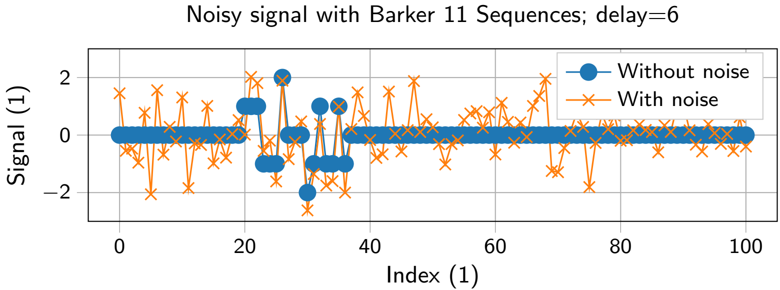


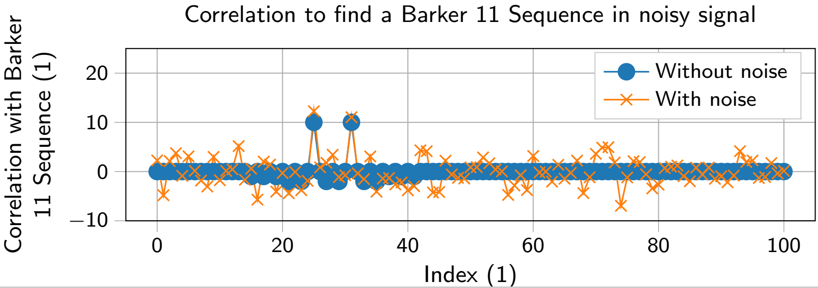
# Partially Overlapping Barker 11 Sequences



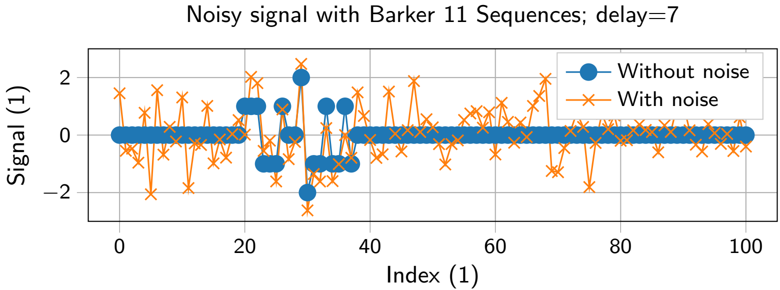



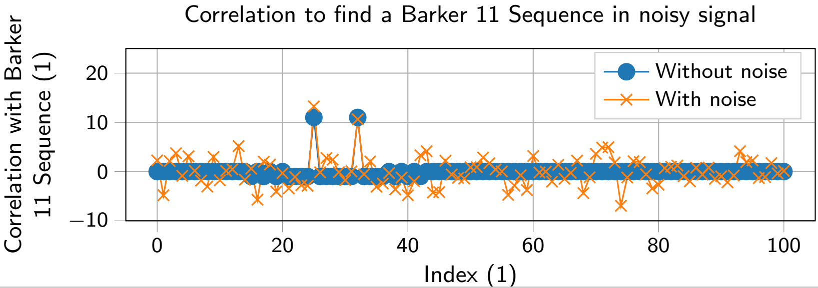
# Partially Overlapping Barker 11 Sequences



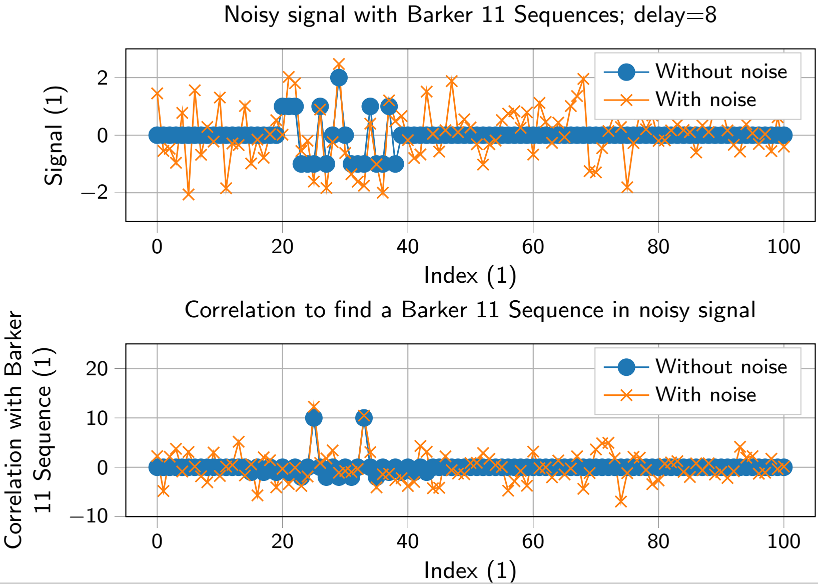



# Partially Overlapping Barker 11 Sequences

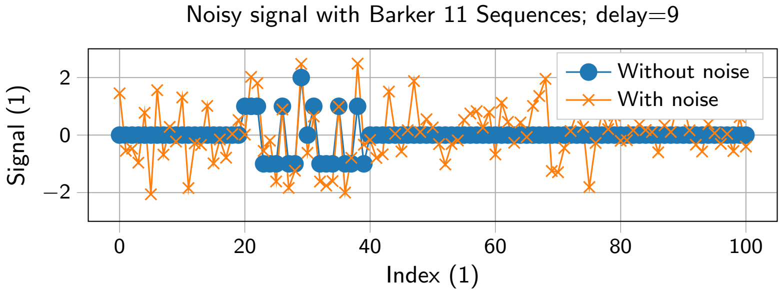


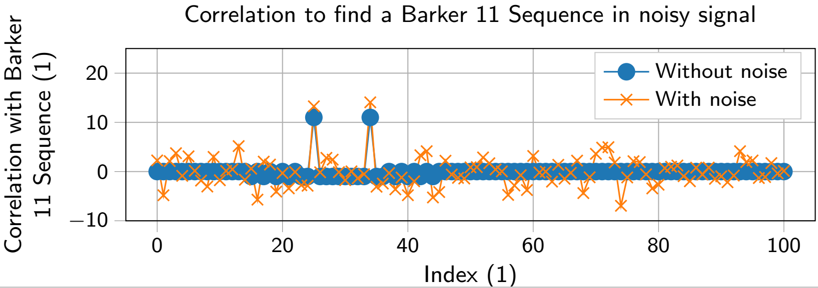



# Partially Overlapping Barker 11 Sequences

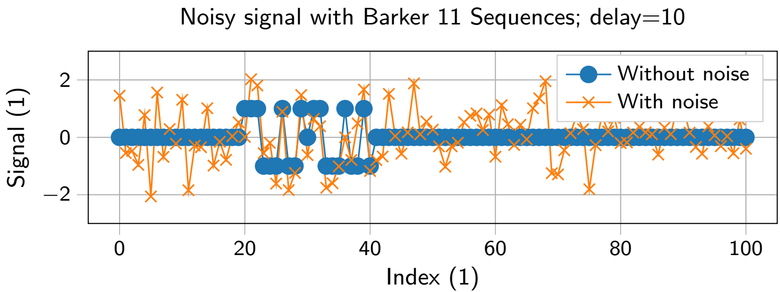


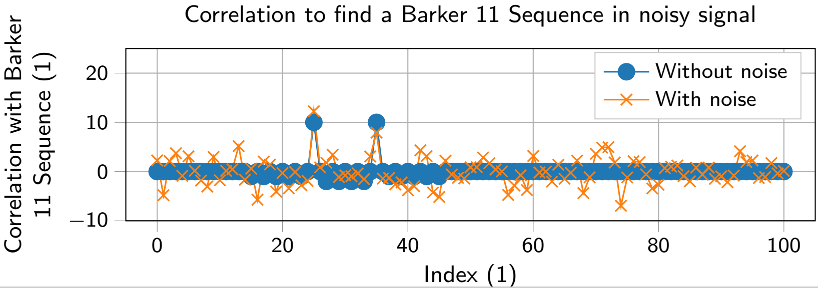
# Partially Overlapping Barker 11 Sequences



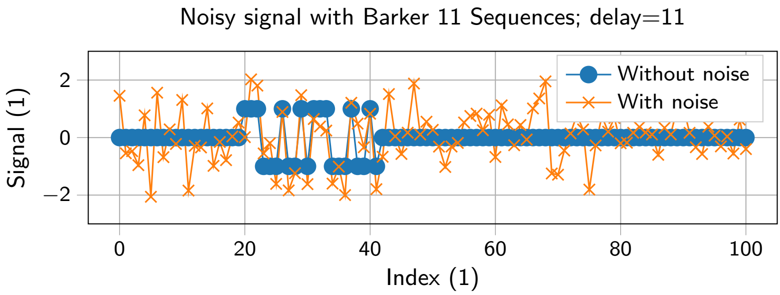



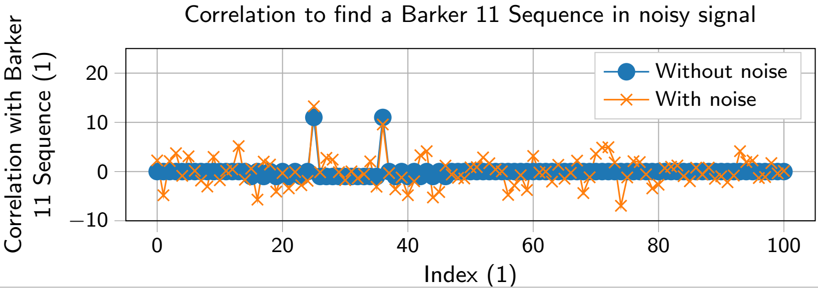
# Partially Overlapping Barker 11 Sequences



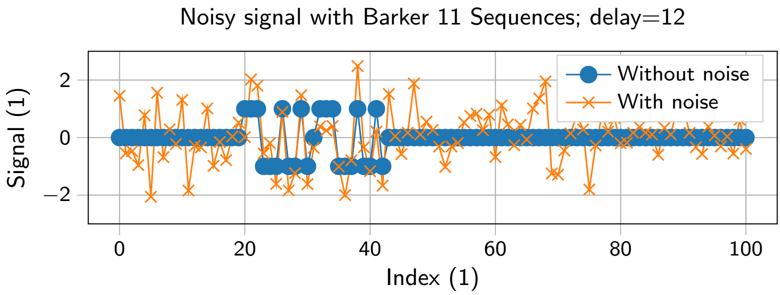



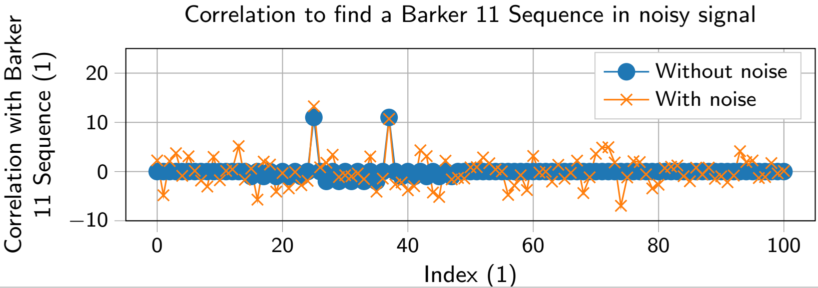
# Partially Overlapping Barker 11 Sequences



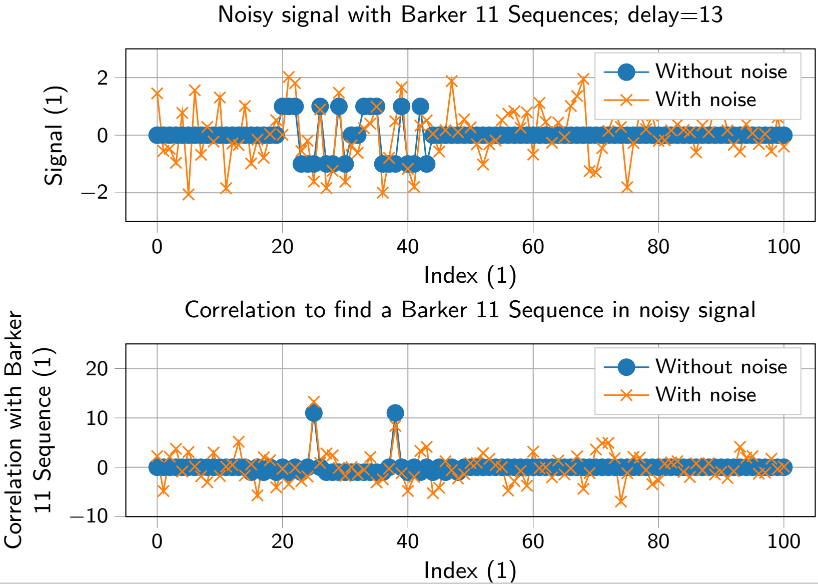



# Partially Overlapping Barker 11 Sequences



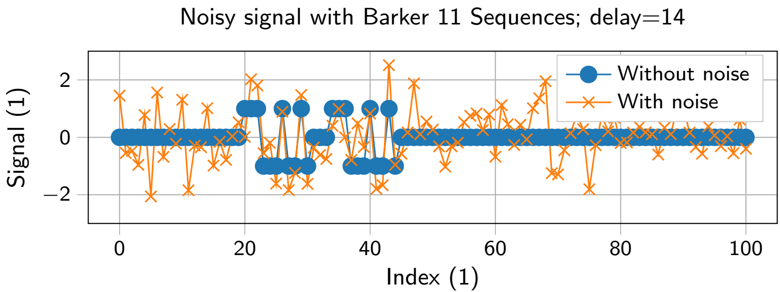



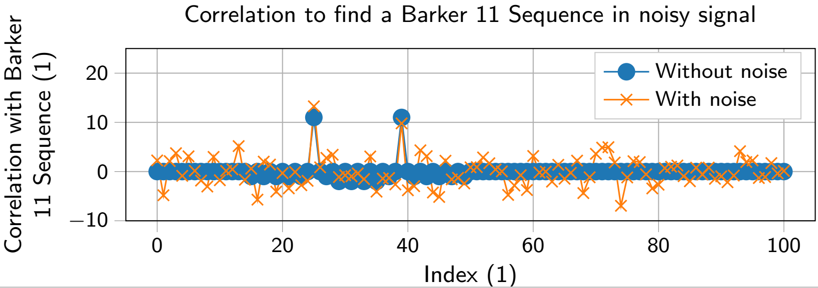
# Partially Overlapping Barker 11 Sequences



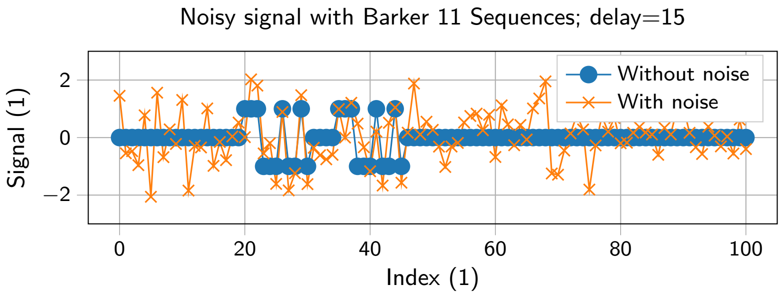

thomas.wagner@jku.at

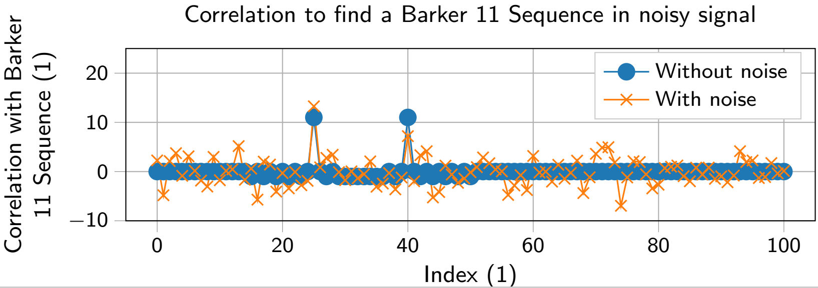
# Partially Overlapping Barker 11 Sequences



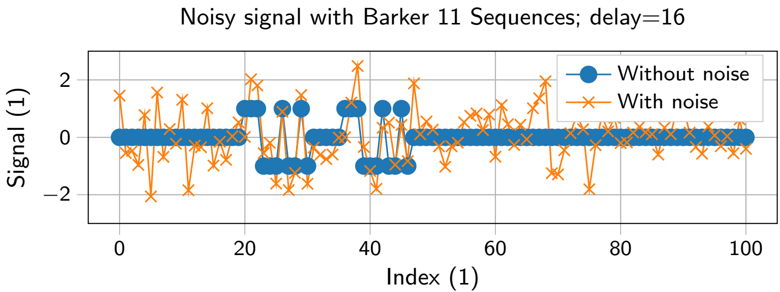



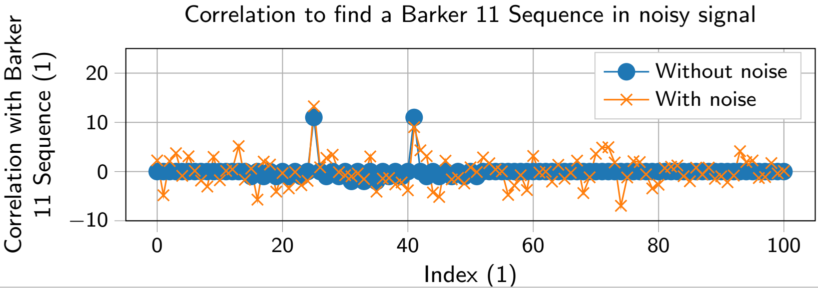
# Partially Overlapping Barker 11 Sequences



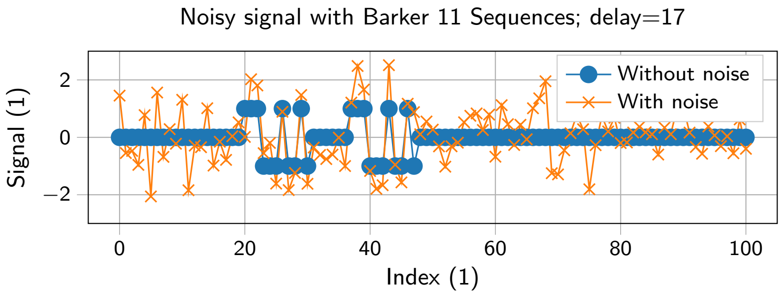



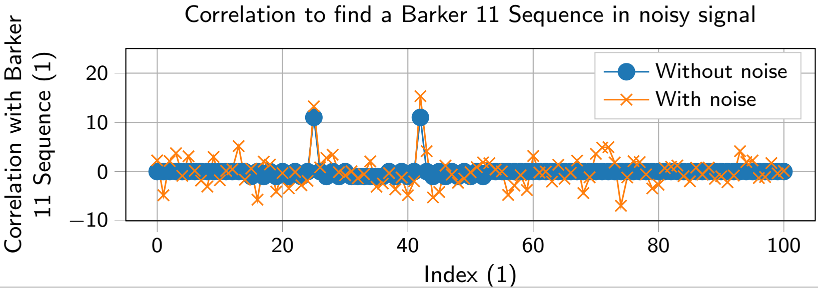
# Partially Overlapping Barker 11 Sequences



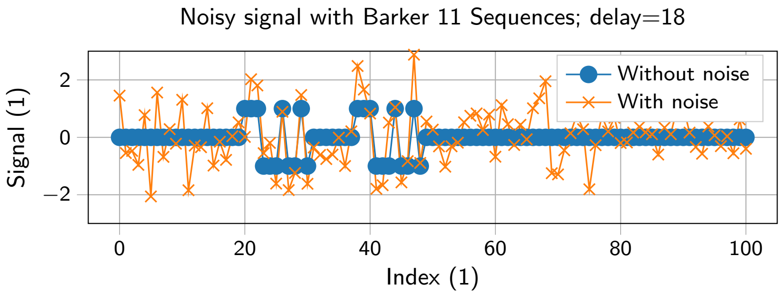



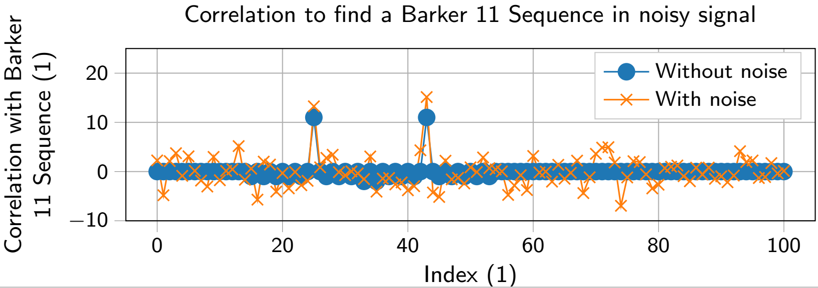
# Partially Overlapping Barker 11 Sequences



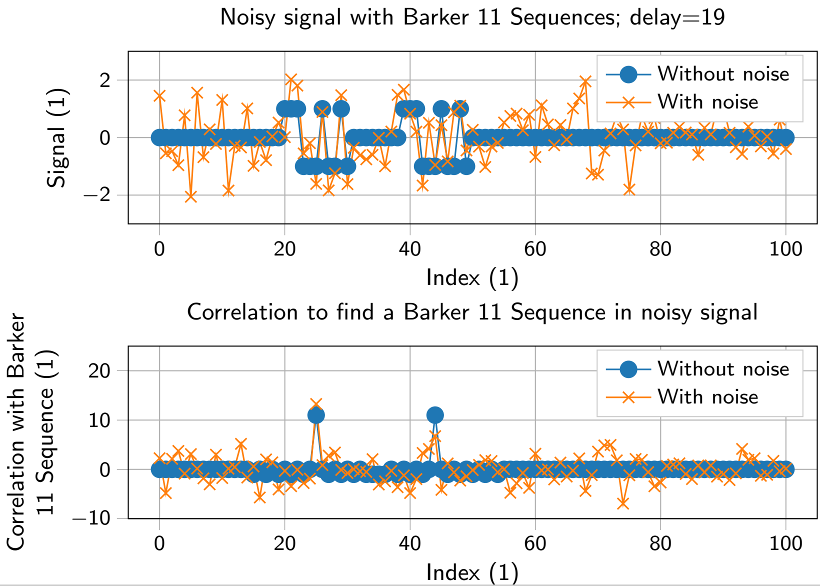



# Partially Overlapping Barker 11 Sequences



# Partially Overlapping Barker 11 Sequences



# Peak Localization and Localization Error

## Peak Localization

- Obtaining the position of a given signal in noise is a reoccurring task, which can be solved by peak search after correlation with the given signal.
- `y_hat=np.argmax(a)` calculates the index of the maximum in `a`
- `y_hat` is called the *estimated* location.

## Estimation Error

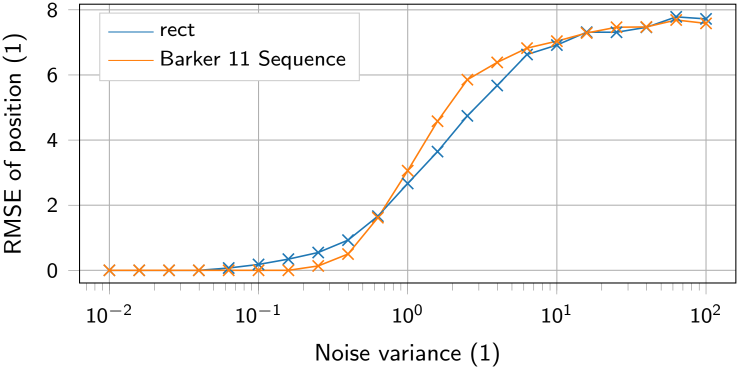
- The root-mean-square error of an estimator can be used as measure of it's quality.

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1}(\hat{y}_n - y)^2}$$

| | |
|---|---|
| $N$ | number of trails |
| $\hat{y}_n$ | estimated peak locations |
| $y$ | true peak locations |

# Monte-Carlo Simulations

- Estimation outcome depends on noise.
- Repeating an experiment based on random samples to derive statistics about the outcome.
- Monte-Carlo simulations are used to assess the quality of an estimator by testing it on different noise realizations.



## Homework: Monte-Carlo Simulation: Peak Localization

- Consider a rect and Barker sequence of length 11 as transmit signal.
- Generate a signal consisting of 10 leading zeros, the transmit signal, and 10 trailing zeros.
- For variances $\sigma^2$ taken from `np.logspace(-1,1,21)`:
  - Add Gaussian noise samples with $\mathcal{N}(0, \sigma^2)$ to the receive signal.
  - Estimate the location of the transmit signal in the receive signal for $N = 1000$ different noise realizations.
  - Calculate the root-mean-square error (RMSE) of the location estimation.
- Plot the RMSE of localization for using the rect and the Barker sequence over the noise variance.