Politechnika Wrocławska
Wydział Informatyki i Telekomunikacji
Kierunek: Informatyczne Systemy Automatyki
Sterowanie adaptacyjne
Projekt 1

STEROWANIE ADAPTACYJNE
PROJEKT 1

Autor: Aliaksandr Kandrat i Mateusz Ambroży
Nr indeksu: 264290 i 264240
Semestr: 5
Grupa: Wtorek, TP, $11^{15} - 13^{00}$
Prowadzący: Dr hab. inż. Grzegorz Mzyk

Październik 2023

Spis treści

1 Wstęp

Głównym celem tego projektu było zbadanie wpływu zakłóceń na jakość estymacji sygnału oraz zbadanie tego, jak błąd średniokwadratowy MSE (od ang. *mean square error*) między oryginalnym sygnałem a jego oszacowaniem zależy od:

- Wariancji zakłóceń pomiarów $Var(Z)$
- Ilości poprzednich pomiarów h, które są używane w filtrze średniej ruchomej do eliminacji zakłóceń

Do przeprowadzenia badań został wykorzystany język programowania **Python** oraz jego narzędzia **NumPy** i **Matplotlib**. W sprawozdaniu zawarty jest kod wykorzystany przy stworzeniu symulacji.

2 Dane wstępne

Jako sygnał wejściowy dla naszego projektu wykorzystaliśmy sygnał sinusoidalny:

$$f(t) = sin(t)$$

```
t = np.linspace(0, 2*np.pi, 1000)
function = np.sin(t)
```

Przyjmijmy, że jego okres (2π) trwa *1000 ms*. Będziemy próbkowali nasz sygnał wejściowy co *4 ms*, otrzymując tym samym *250 próbek*.

```
n_samples = 250
samples_t = np.linspace(0, 2*np.pi, n_samples)
samples_array = np.sin(samples_t)
```

Próbki te są zakłócone szumem wygenerowanym z rozkładu normalnego:

```
std_dev = 0.15
noise = std_dev * np.random.randn(n_samples)
noised_samples = samples_array + noise
```

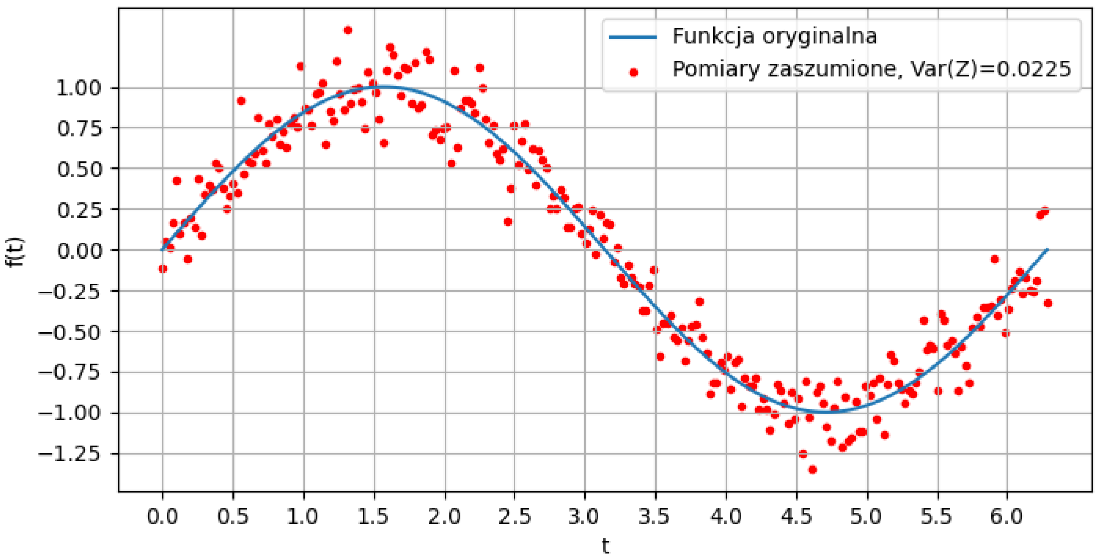
Funkcja *np.random.randn()* zwraca liczby z rozkładu normalnego, czyli o wartości oczekiwanej równej 0 oraz o odchyleniu standardowym równym 1. We fragmencie kodu powyżej zmieniamy wartość odchylenia standardowego na **std_dev**, by zmniejszyć rozrzut otrzymanych zakłóceń.

Zakłócenia pomiarów eliminowaliśmy przy pomocy filtru ruchomej średniej (ang. *moving average*, której głównym parametrem jest wartość h – liczba poprzednich pomiarów branych pod uwagę podczas uśredniania wartości kolejnego pomiaru:

```
def movingAverage(signal: np.array, h: int) -> np.array:
"""
    Simple moving average filter.

    Moving average filter made for smoothing signal's noise.
    !Important!
    This function cuts `h` first samples of the `signal`. For
    input `signal` of size `(1, 628)` output signal will have
    size `(1, 628-h)` and first `h` samples will be lost.

    Arguments:
    signal: np.array - signal to filter
    h: int - number of samples to consider

    Returns:
    np.array - filtered signal of size `(n-h)`
"""

filtered_signal = np.zeros(signal.size - h)
for i in range(filtered_signal.size):
    filtered_signal[i] = 1/h * np.sum(signal[i:h+i])
return filtered_signal
```

3 Przebieg badań

Próbkując sygnał wejściowy w sposób opisany w poprzednim rozdziale, otrzymaliśmy następujący wykres:



Rysunek 3.1: Wykres sygnału wejściowego oraz próbek zakłóconych

Używając używając filtru średniej ruchomej, który działa według wzoru:

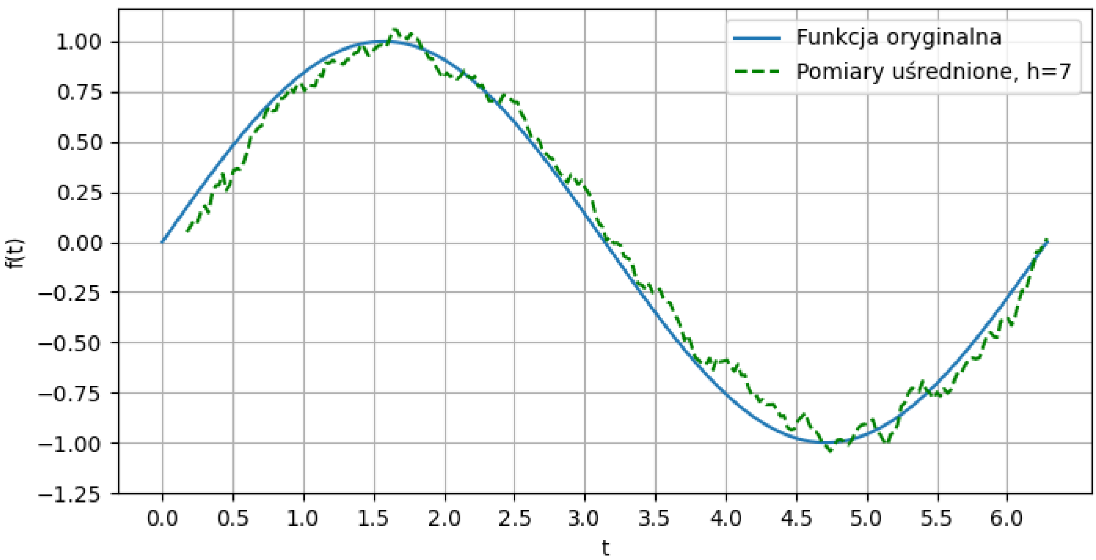
$$\hat{\Theta}_k = \frac{1}{h}\sum_{i=0}^{h-1}\Theta_{k-i} \tag{3.1}$$

gdzie:

- $\hat{\Theta}_k$ - obliczona wartość pomiaru
- Θ_{k-i} - wartości h poprzednich pomiarów

Dla przykładu, spróbujemy wyeliminować zakłócenia zakładając $h = 7$. Wykres :

```
h = 7
filtered = movingAverage(noised_samples, h)
```



Rysunek 3.2: Sygnał otrzymany na wyjściu filtru dla *h=7* oraz *Var(Z)=0.0225*

3.1 Zależność *MSE* od *h*

Aby przeprowadzić skuteczną analizę, potrzebujemy wskaźnika oceniającego dokładność uzyskanych rezultatów w porównaniu do wartości oczekiwanej. W tym celu zastosujemy błąd średniokwadratowy (MSE), opisany wzorem:

$$MSE = \frac{1}{h}\sum_{k}(\hat{\Theta}_k - \Theta_k^*)^2 \tag{3.2}$$

gdzie:

- **MSE** - wartość błędu średniokwadratowego
- $\hat{\Theta}_k$ - wartość pomiaru w chwili k po użyciu filtru średniej ruchomej
- Θ_k^* - prawdziwa wartość w chwili k

Implementacja tej funkcji w *Python'ie* wydląda następująco:

```python
def MSE(first: np.array, second: np.array) -> float:
    """"
        Calculates the Mean Squared Error of two passed arrays.

        Arguments:
        first: np.array - first array
        second: np.array - second array

        Returns:
        float - MSE of two passed arrays
    """
    if first.size != second.size:
        raise ValueError(f"Number of samples in first and second \
            arrays must match! (first={(first.size)}, second={(second.size)})")
    return np.mean(np.square(first-second))
```

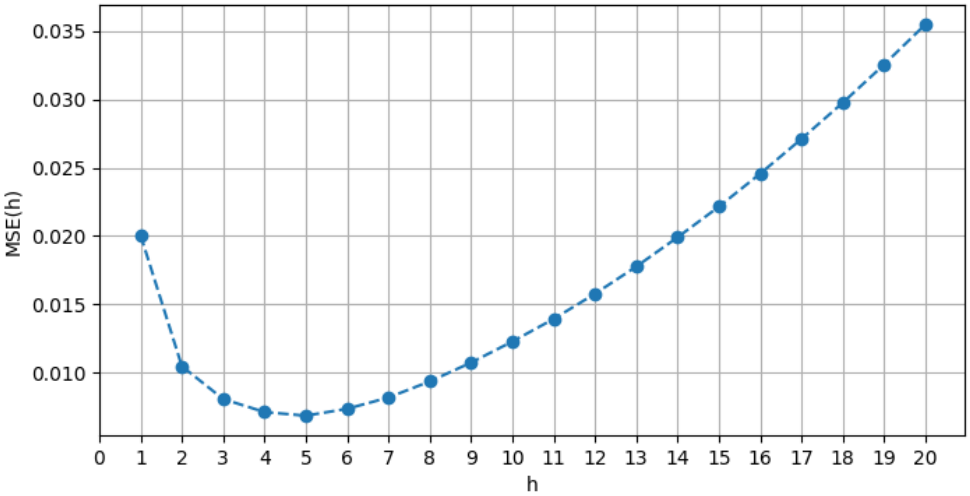
W celu znalezienia optymalnej wartości h filtru, porównamy wartości *MSE* dla różnych wartości h. Stworzymy tablicę wartości h:

```python
h_min, h_max = 1, 21
h_array = np.arange(h_min, h_max, 1)
```

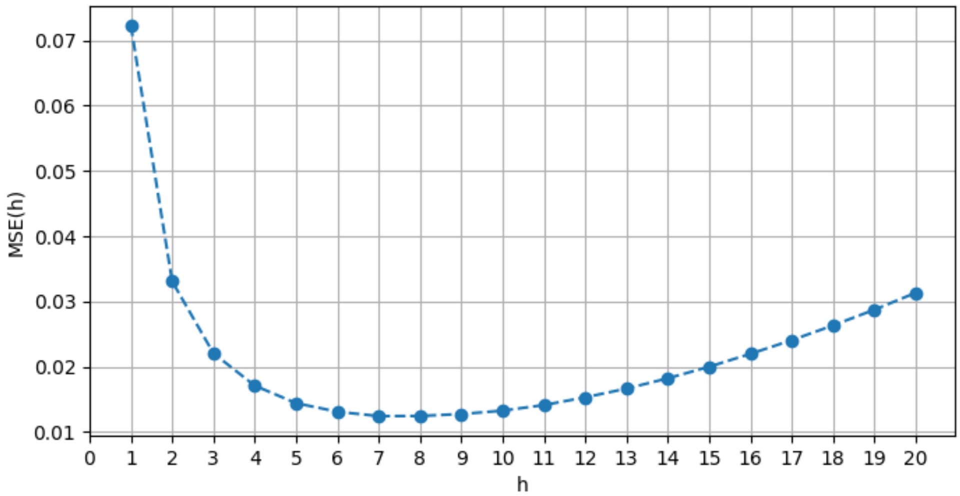
Następnie dla każdej wartości z **h_array** będziemy obliczali wartość *MSE* oraz będziemy dodawali ją do tablicy **mse_array**:

```python
# MSE values for each 'h' in 'h_array'
mse_array = np.zeros(h_array.size)

for h in h_array:
    filtered = movingAverage(noised_samples, h)
    mse_array[h-1] = MSE(filtered, samples_array[h:])
```

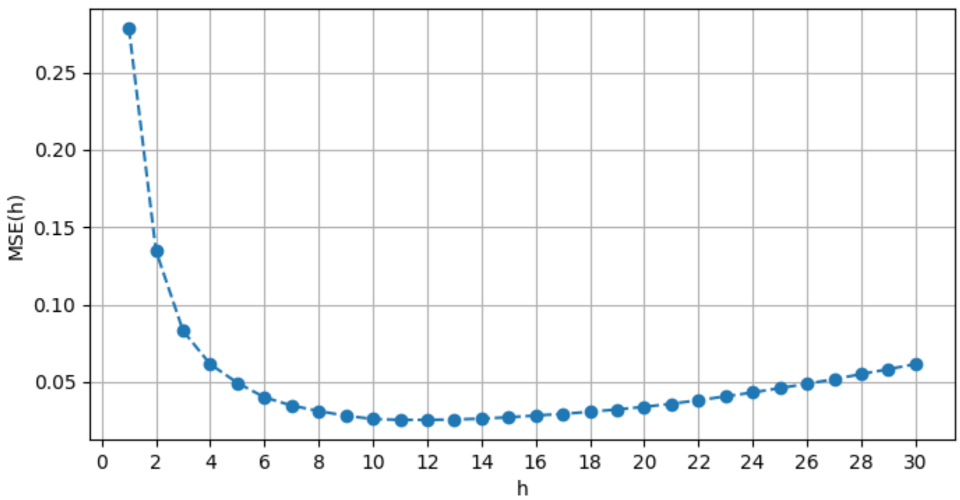
Zależność wartości *MSE* od wartości h możemy zaobserwować na wykresach poniżej:



Rysunek 3.3: $Var(Z) = 0.0225$, optymalna wartość $h = 5$



Rysunek 3.4: $Var(Z) = 0.09$, optymalna wartość $h = 7$



Rysunek 3.5: $Var(Z) = 0.25$, optymalna wartość $h = 11$

Z powyższych wykresów możemy zrobić dwa główne wnioski:

1. Wartość h rośnie wraz ze wzrostem wariancji zakłóceń.
2. Zbyt małe wartości h nie są wstanie wyeliminować dostateczną ilość błędów. Zbyt duże wartości wprowadzają opóźnienie, które nie pozwala układowi reagować na szybkie zmiany sygnału wejściowego.

3.2 Zależność *MSE* od *Var(Z)*

W celu zbadania wpływu wariancji zakłóceń $Var(Z)$ na wartość *MSE* stworzymy tablicę, przechowującą wartości wariancji:

```
variance_min, variance_max = 0, 2.1
variance = np.arange(variance_min, variance_max, 0.1)
```

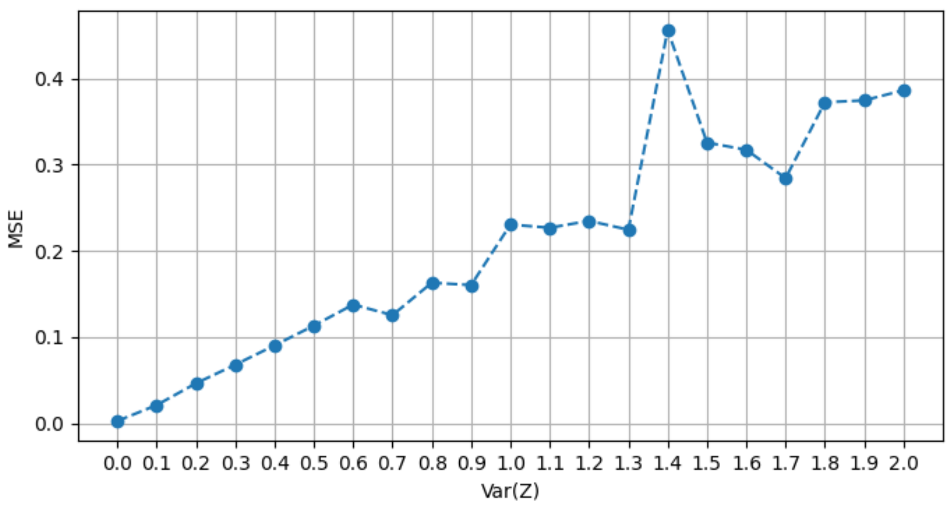
Przyjmijmy wartość $h = 5$ oraz stworzymy tablicę **mse_array** do przechowywania wartości *MSE*:

```
h=5
mse_array = np.zeros(variance.size)
```

Następnie dla każdej wartości wariancji generujemy zakłócenia dla pomiarów. Stosujemy filtr ruchomej średniej oraz zapisujemy wynik wartości *MSE* do tablicy:

```
for i, v in enumerate(variance):
    noise =  np.sqrt(v) * np.random.randn(n_samples)
    noised_samples = samples_array + noise
    filtered = movingAverage(noised_samples, h)
    mse_array[i] = MSE(filtered, samples_array[h:])
```

W wyniku otrzymujemy następujący wykres:



Rysunek 3.6: Zależność *MSE* od $Var(Z)$. $h = 5$

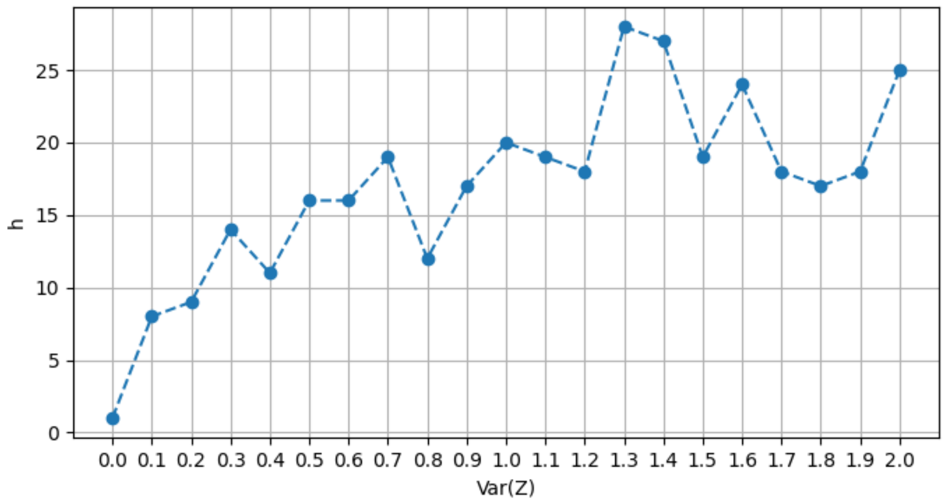
Z tego wykresu jednoznacznie widać, że wzrost wariancji zakłóceń wpływa negatywnie na jakość odfiltrowywania. Kiedy wariancja zakłóceń wzrasta, sygnał staje się bardziej nieregularny i zawiera więcej nieprzewidywalnych wahań, zwiększa się jego rozrzut wokół wartości oczekiwanej. Powoduje to wzrost błędu *MSE* a w praktyce oznacza, że im większa jest wariancja zakłóceń, tym trudniej jest uzyskać dokładną estymację oryginalnego sygnału.

3.3 Zależność *h* od *Var(Z)*

W celu zbadania wpływu wariancji zakłóceń $Var(Z)$ na wartość h będziemy używali tablicę wartości wariancji (**variance**) oraz tablicę wartości h (**h_array**). Dla każdej wartości wariancji generujemy zakłócenia dla pomiarów. Następnie, dla wyliczonych zakłóceń obliczamy optymalną wartość h. Zapisujemy tą wartość w tablicy **h_for_variance** i powtarzamy obliczenia dla następnej wartości wariancji:

```
h_for_variance = np.zeros(variance.size)

for i, v in enumerate(variance):
    noise =  np.sqrt(v) * np.random.randn(n_samples)
    noised_samples = samples_array + noise

    for h in h_array:
        filtered = movingAverage(noised_samples, h)
        mse_array[h-1] = MSE(filtered, samples_array[h:])

    min_mse = np.argmin(mse_array)
    h_opt = h_array[min_mse]
    mse = mse_array[min_mse]
    h_for_variance[i] = h_opt
```

W wyniku otrzymujemy następujący wykres:



Rysunek 3.7: Zależność optymalnych wartości *h* od *Var(Z)*

4 Wnioski

- Jak wartość h wpływa na błąd średniokwadratowy $MSE(\hat{\Theta}_k)$?

  Kiedy zwiększamy zakres okna filtracji, czyli h, zdolność do eliminowania zakłóceń sygnału rośnie. Oznacza to, że możemy zmniejszyć wpływ krótkotrwałych zakłóceń, co przekłada się na niższy błąd średniokwadratowy (*MSE*). Niemniej jednak, zbyt duży zakres h może sprawić, że sygnał odfiltrowany zacznie reagować z opóźnieniem na nagłe zmiany w danych wejściowych, co może nie być pożądane w wielu aplikacjach.

- W jaki sposób wariancja zakłóceń *Var(Z)* wpływa na błąd średniokwadratowy $MSE(\hat{\Theta}_k)$?

  Im większa wariancja zakłóceń, tym trudniej jest oddzielić oryginalny sygnał od interferencji. Wyższa wariancja oznacza bardziej nieregularne i intensywne zakłócenia, bardziej rozrzucone wokół wartości oczekiwanej, co komplikuje filtrację i prowadzi do wzrostu wartości *MSE*.

- Jaka jest zależność między optymalną wartością h a wariancją zakłóceń *Var(Z)*?

  Rozmiar optymalnego okna filtracji, h, może ulec zmianie w zależności od intensywności oraz zakresu wartości zakłóceń w sygnale. W sytuacji, gdy zakłócenia są bardziej intensywne (wyższa wariancja), większe okno filtracji może być potrzebne, aby efektywnie usunąć te zakłócenia i uzyskać niższe wartości *MSE*. Z drugiej strony, chociaż większe okno może poprawić jakość filtracji, może także wprowadzić opóźnienia i zniekształcenia w sygnale. Dlatego ważne jest znalezienie odpowiedniego kompromisu w doborze wartości H w kontekście danej aplikacji i poziomu zakłóceń.