LAB3-Random-Signals

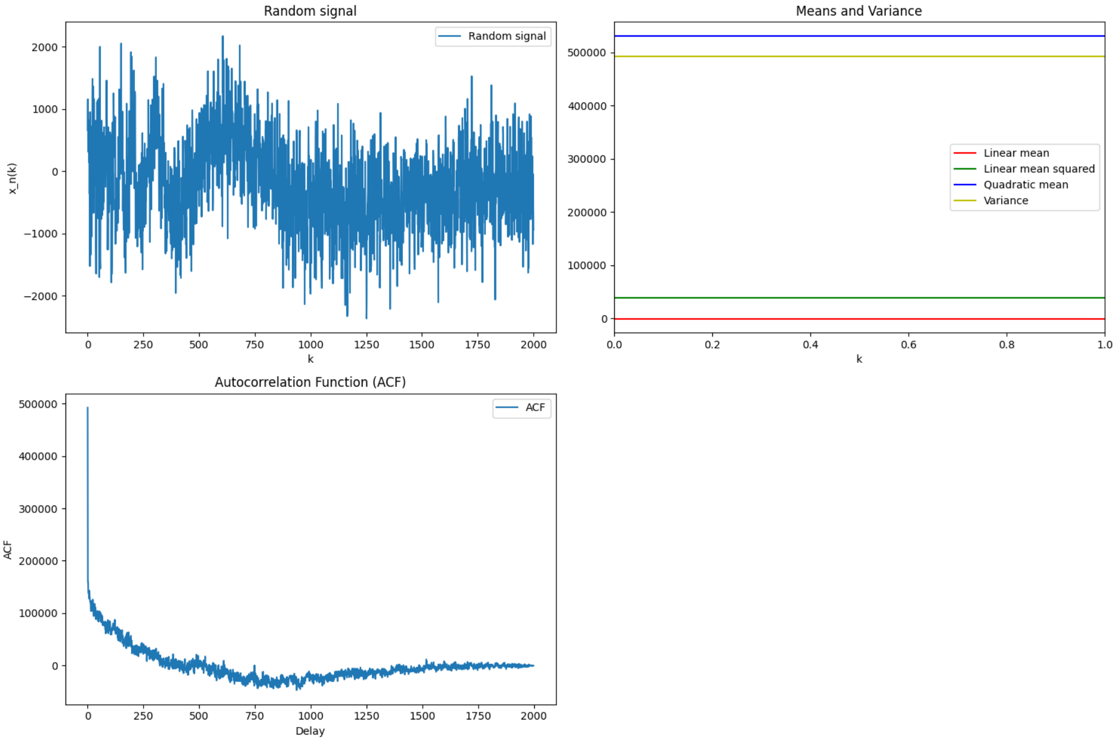
December 26, 2023

[11]:

```
# Rozpoczęcie od k = 1, aby uniknąć dzielenia przez zero
k = np.arange(1, N+1)
ensemble = A * np.cos(2 * np.pi * f / k) + B * np.random.normal(0, 1, N)

# Ponowne obliczenie wartości
linear_mean = np.mean(ensemble)
linear_mean_squared = linear_mean ** 2
quadratic_mean = np.mean(ensemble ** 2)
variance = np.var(ensemble)

print("Linear mean =", linear_mean)
print("Linear mean squared =",linear_mean_squared)
print("Quadratic mean =",quadratic_mean)
print("Variance =",variance)

# Ponowne utworzenie wykresów
plt.figure(figsize=(15, 10))

# Wykres sygnału
plt.subplot(2, 2, 1)
plt.plot(k, ensemble, label="Random signal")
plt.title("Random signal")
plt.xlabel("k")
plt.ylabel("x_n(k)")
plt.legend()

# Wykres średnich i wariancji
plt.subplot(2, 2, 2)
plt.axhline(y=linear_mean, color='r', linestyle='-', label="Linear mean")
plt.axhline(y=linear_mean_squared, color='g', linestyle='-', label="Linear mean␣
  ↪squared")
plt.axhline(y=quadratic_mean, color='b', linestyle='-', label="Quadratic mean")
plt.axhline(y=variance, color='y', linestyle='-', label="Variance")
plt.title("Means and Variance")
plt.xlabel("k")
plt.legend()
```

```
# Funkcja autokorelacji (ACF)
acf = np.correlate(ensemble - linear_mean, ensemble - linear_mean, mode='full') / N
acf = acf[N-1:]  # Biorąc pod uwagę tylko nieujemne opóźnienia

plt.subplot(2, 2, 3)
plt.plot(acf, label="ACF")
plt.title("Autocorrelation Function (ACF)")
plt.xlabel("Delay")
plt.ylabel("ACF")
plt.legend()

plt.tight_layout()
plt.show()
```

```
Linear mean = -195.1477760054931
Linear mean squared = 38082.654479890116
Quadratic mean = 530396.7457995306
Variance = 492314.0913196406
```



[ ]:

```
```