

Sprawozdanie Eksploracja Danych

Laboratorium 2 - 20.11.2024

Zbadanie funkcji estymującej jądra na danych generowanych losowo oraz przedstawienie wyników w formie graficznej.

Szymon Moździerz Krzysztof Żelazny Adrian Sławiński

Cele sprawozdania:

1. Sprawdzenie wpływu ilości wygenerowanych próbek na ich graficzne przedstawienie na wykresie.
2. Zbadanie wpływu ilości przedziałów wykorzystanych do przedstawienia danych na histogramie danego rozkładu, na zgodność wykresu z teorią.
3. Zbadanie wpływu różnych parametrów na wartości funkcji estymatora jądra.

Wykorzystane biblioteki

```
import numpy as np
import matplotlib.pyplot as plt
```

Ziarno

Dla zapewnienia powtarzalności otrzymywanych wyników, wprowadzone zostało stałe ziarno.

```
np.random.seed(4)

# sample_size = 10
# uniform_sample = np.random.uniform(2, 7, sample_size)
# y = [0 for _ in range(sample_size)]

# plt.scatter(uniform_sample, y, marker='|', s=500, c='r')
# plt.title('Wykres rozkładu')
# plt.ylim(0, 0.5)
# plt.xlim(0, 10)
# plt.show()

# plt.hist(uniform_sample, bins=5, edgecolor='black', alpha=0.7)
# plt.title('Histogram rozkładu jednorodnego [2, 7]')
# plt.xlabel('Wartość')
# plt.ylabel('Częstość wystąpień')
# plt.xlim(0, 10)
# plt.grid(axis='y', linestyle='--', alpha=0.7)
# plt.show()
```

Eksperyment 1

Pierwszy eksperyment polegał na sprawdzeniu jak ilość generowanych próbek, w tym przypadku dla rozkładu jednorodnego, wpłynie na wizualne przedstawienie danych. Miało to być organoleptyczne sprawdzenie dla jakich wartości, wizualizacja rozkładu będzie zgodna z teorią. Dla każdej wartości sporządzony został wykres przedstawiający rozłożenie próbek na osi X, gdzie każda próbka została oznaczona poprzez marker '|', a także histogram częstości ich występowania.

```
sample_sizes = [10, 20, 50, 100, 1000, 10000, 100000, 1000000]

fig, axes = plt.subplots(len(sample_sizes), 2, figsize=(12,
len(sample_sizes) * 4))

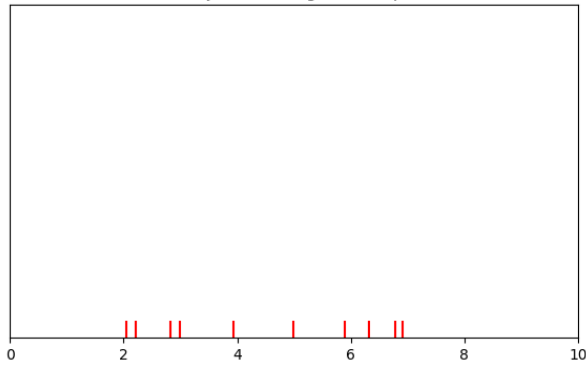
for idx, sample_size in enumerate(sample_sizes):
    uniform_sample = np.random.uniform(2, 7, sample_size)
    y = [0 for _ in range(sample_size)]

    axes[idx, 0].scatter(uniform_sample, y, marker='|', s=500, c='r')
    axes[idx, 0].set_title(f'Rozkład jednorodnego dla {sample_size}
próbek')
    axes[idx, 0].set_xlabel('Wartość')
    axes[idx, 0].set_ylabel('Częstość wystąpień')
    axes[idx, 0].set_ylim(0, 0.5)
    axes[idx, 0].set_xlim(0, 10)
    axes[idx, 0].set_yticks([])

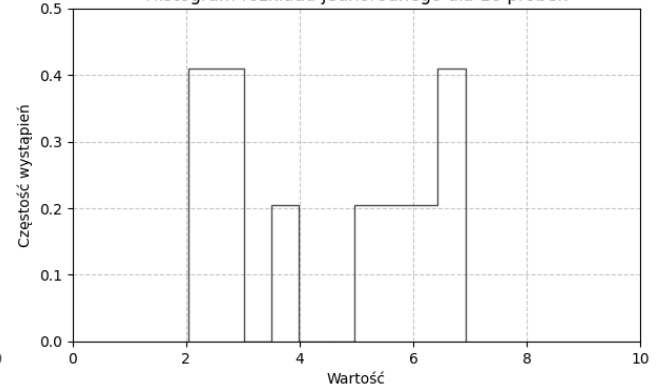
    axes[idx, 1].hist(uniform_sample, bins=10, edgecolor='black',
alpha=0.7, histtype='step', fill=False, density=True)
    axes[idx, 1].set_title(f'Histogram rozkładu jednorodnego dla
{sample_size} próbek')
    axes[idx, 1].set_xlabel('Wartość')
    axes[idx, 1].set_ylabel('Częstość wystąpień')
    axes[idx, 1].set_ylim(0, 0.5)
    axes[idx, 1].set_xlim(0, 10)
    axes[idx, 1].grid(True, linestyle='--', alpha=0.7)

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

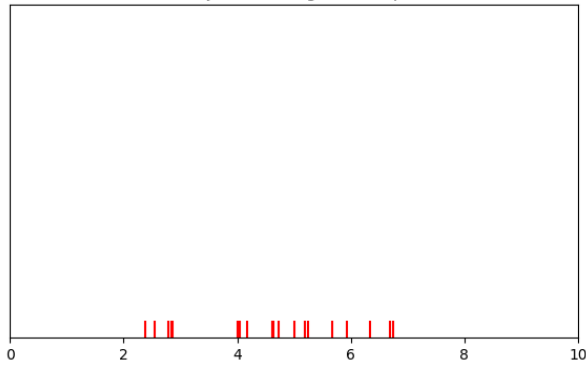
Rozkład jednorodnego dla 10 próbek



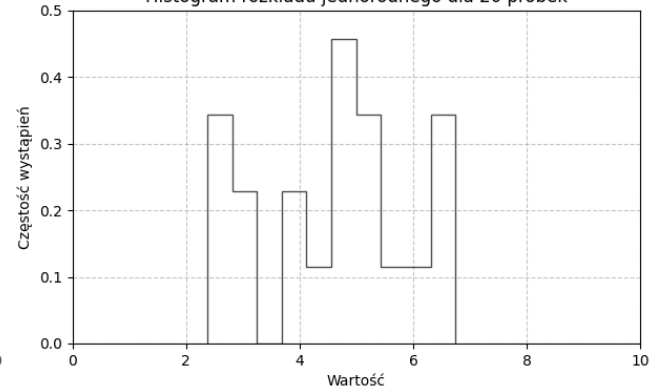
Histogram rozkładu jednorodnego dla 10 próbek



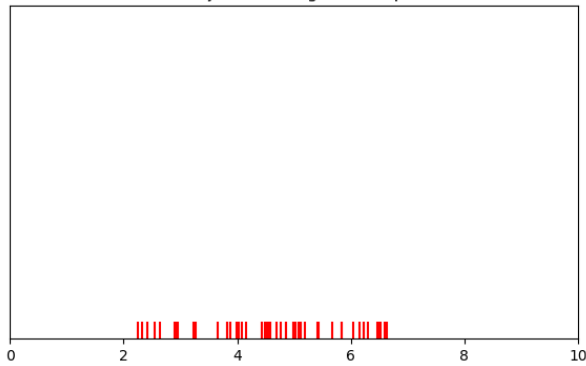
Rozkład jednorodnego dla 20 próbek



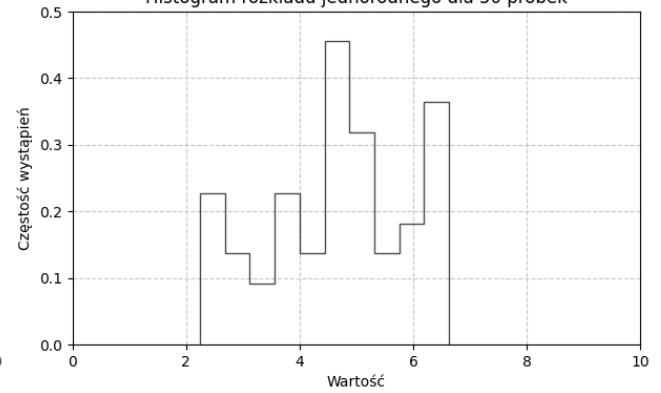
Histogram rozkładu jednorodnego dla 20 próbek



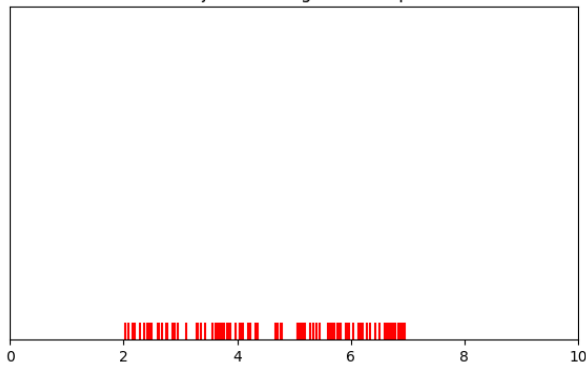
Rozkład jednorodnego dla 50 próbek



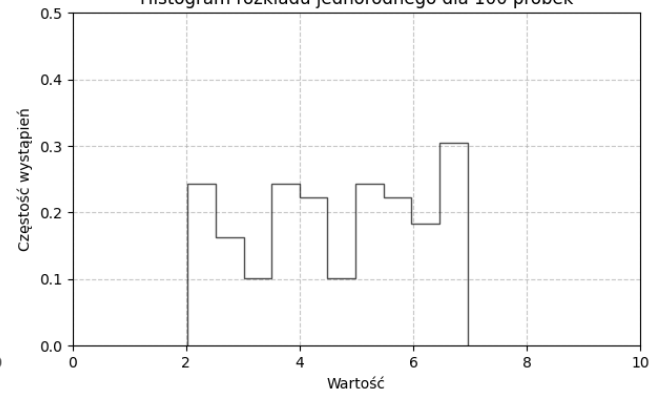
Histogram rozkładu jednorodnego dla 50 próbek



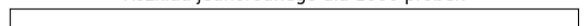
Rozkład jednorodnego dla 100 próbek



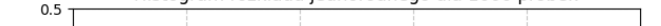
Histogram rozkładu jednorodnego dla 100 próbek



Rozkład jednorodnego dla 1000 próbek



Histogram rozkładu jednorodnego dla 1000 próbek



```

# sample_size = 1000
# bin_counts = [10, 20, 50, 100, 1000, 2000]

# fig, axes = plt.subplots(3, 2, figsize=(12, 12))

# uniform_sample = np.random.uniform(2, 7, sample_size)
# y = [0 for _ in range(sample_size)]

# for i, ax in enumerate(axes.flat):
#     ax.hist(uniform_sample, bins=bin_counts[i], edgecolor='black',
#             alpha=0.7, histtype='step', fill=False, density=True)
#     ax.set_title(f'Histogram (Sample Size = {bin_counts[i]})')
#     ax.set_xlabel('Value')
#     ax.set_ylabel('Frequency')
#     ax.set_ylim(0, 0.5)
#     ax.set_xlim(0, 10)
#     ax.grid(True, linestyle='--', alpha=0.7)

# # fig, axes = plt.subplots(3, 2, figsize=(12, 12))

# # # Loop through the axes and data samples to create plots
# # for i, ax in enumerate(axes.flat):
# #     ax.hist(data_samples[i], bins=5, edgecolor='black', alpha=0.7,
# #             histtype='step', fill=False)
# #     ax.set_title(f'Plot {i + 1}')
# #     ax.set_xlabel('Value')
# #     ax.set_ylabel('Frequency')
# #     ax.grid(axis='y', linestyle='--', alpha=0.7)

# plt.tight_layout()
# plt.show()

sample_size = 1000
bin_counts = [10, 20, 50, 100, 1000, 2000]

fig, axes = plt.subplots(3, 2, figsize=(12, 12))

uniform_sample = np.random.uniform(2, 7, sample_size)

sample_1 = np.random.uniform(2, 7, sample_size // 2)
sample_2 = np.random.uniform(8, 9, sample_size // 2)

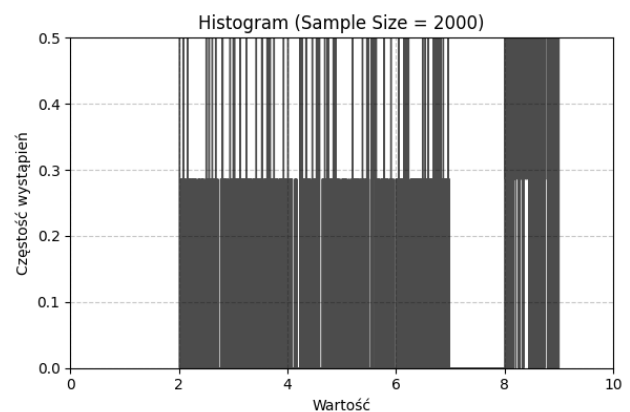
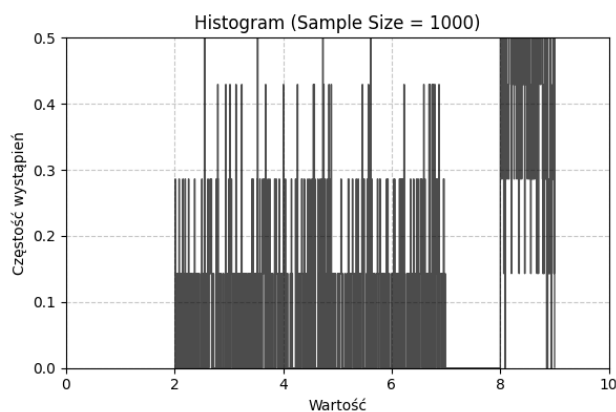
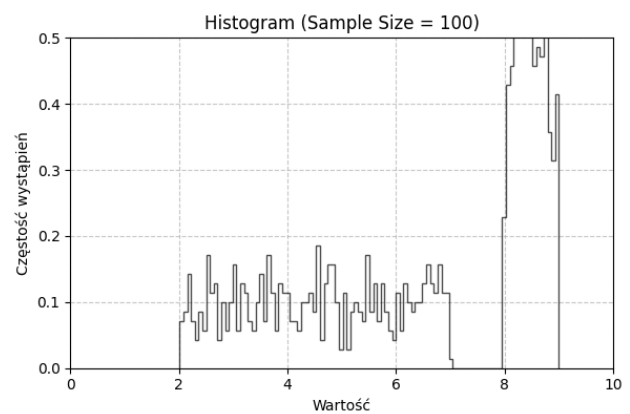
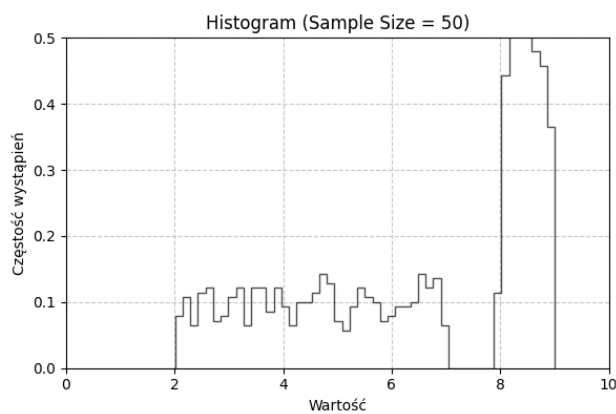
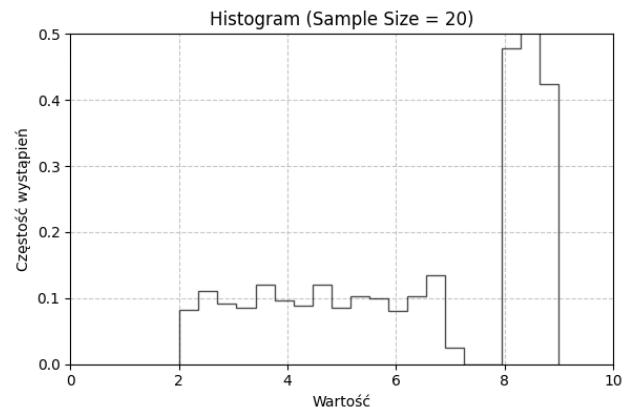
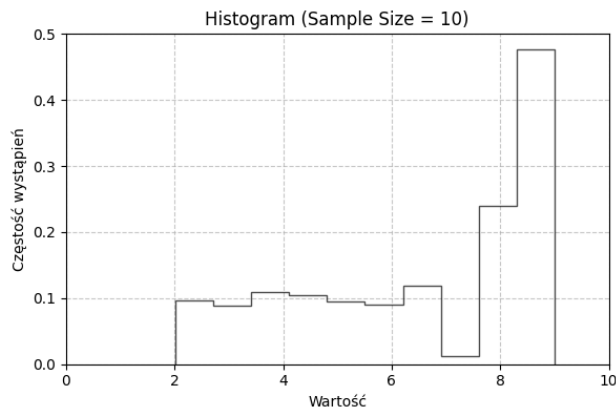
combined_sample = np.concatenate([sample_1, sample_2])

for i, ax in enumerate(axes.flat):
    ax.hist(combined_sample, bins=bin_counts[i], edgecolor='black',
            alpha=0.7, histtype='step', fill=False, density=True)
    ax.set_title(f'Histogram (Sample Size = {bin_counts[i]})')
    ax.set_xlabel('Wartość')
    ax.set_ylabel('Częstość wystąpień')
    ax.set_ylim(0, 0.5)

```

```
ax.set_xlim(0, 10)
ax.grid(True, linestyle='--', alpha=0.7)
```

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

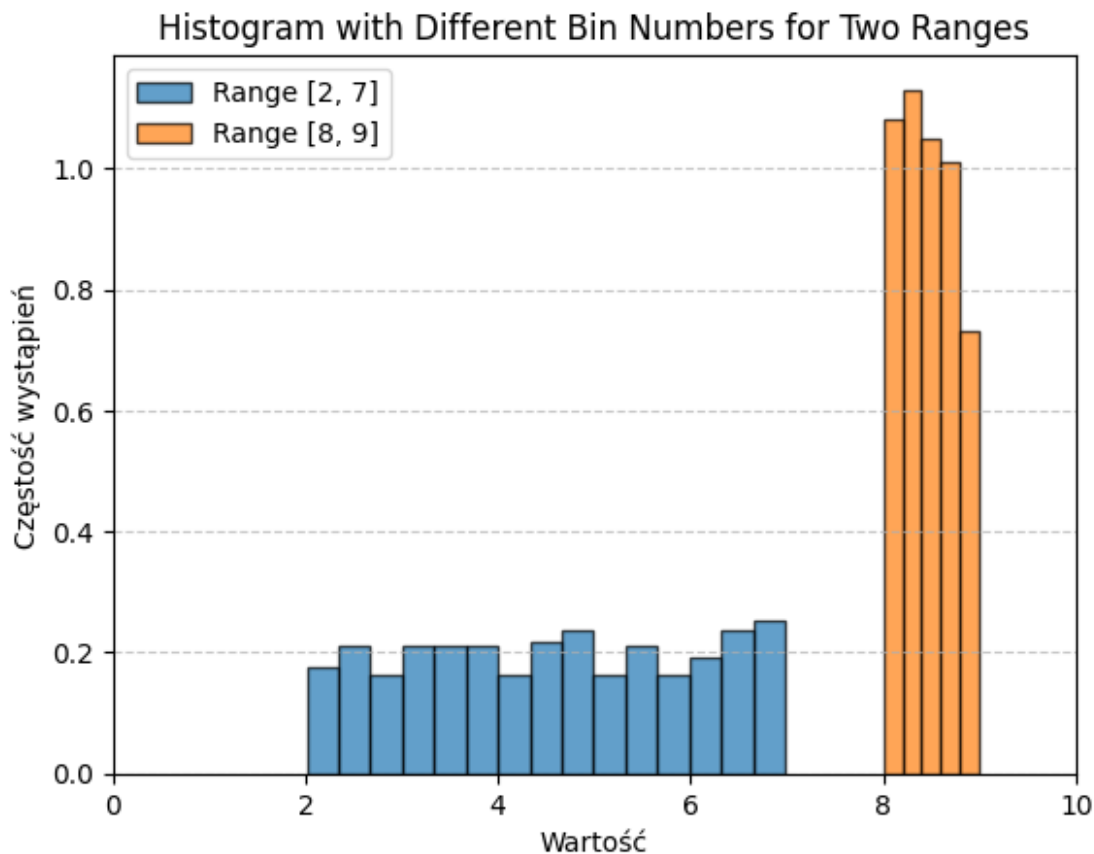


```
bins_range_1 = 15
bins_range_2 = 5

plt.hist(sample_1, bins=bins_range_1, edgecolor='black', alpha=0.7,
label='Range [2, 7]', density=True)
```

```
plt.hist(sample_2, bins=bins_range_2, edgecolor='black', alpha=0.7,
label='Range [8, 9]', density=True)

plt.title('Histogram with Different Bin Numbers for Two Ranges')
plt.xlabel('Wartość')
plt.ylabel('Częstość wystąpień')
plt.legend()
plt.xlim(0, 10)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



```
sample_size = 1000
bin_counts = [10, 20, 50, 100, 1000, 2000]

fig, axes = plt.subplots(2, 3, figsize=(17, 12))

normal_sample = np.random.normal(5, 1, sample_size)

for i, ax in enumerate(axes.flat):
    ax.hist(normal_sample, bins=bin_counts[i], edgecolor='black',
alpha=0.7, histtype='step', fill=False, density=True)
    ax.set_title(f'Histogram rozkładu normalnego dla {bin_counts[i]}
binów')
```

```

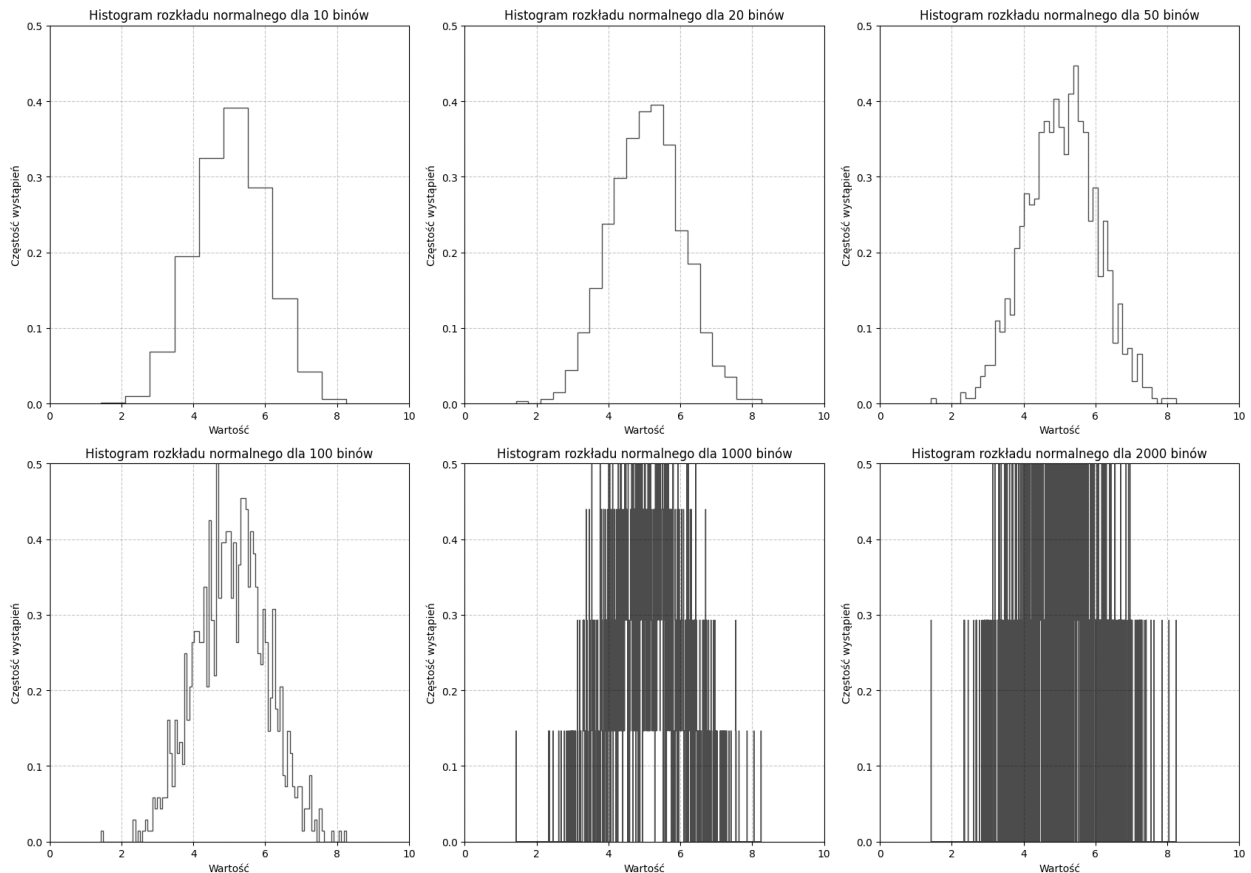
ax.set_xlabel('Wartość')
ax.set_ylabel('Częstość wystąpień')
ax.set_ylim(0, 0.5)
ax.set_xlim(0, 10)
ax.grid(True, linestyle='--', alpha=0.7)

```

```

plt.tight_layout()
plt.show()

```



```

import math

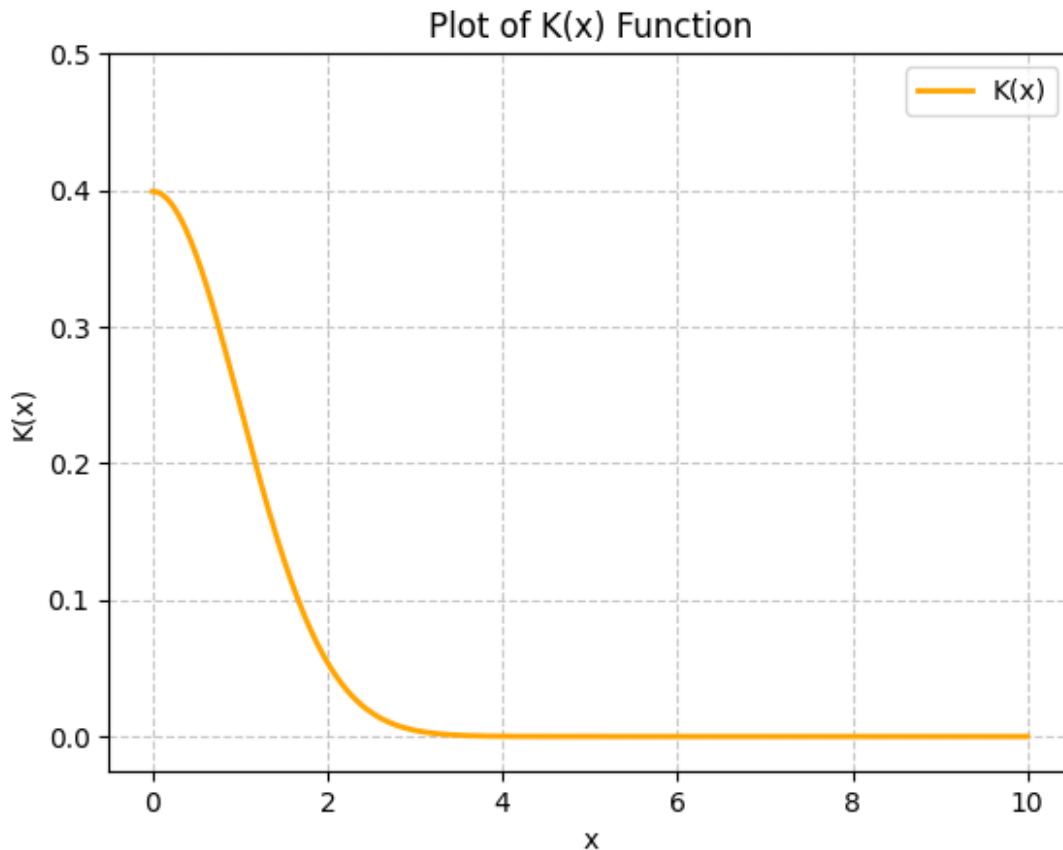
def K(x):
    return 1. / math.sqrt(2 * math.pi) * math.exp(-x ** 2 / 2)

x_values = np.linspace(0, 10, 500)
y_values = [K(x) for x in x_values]

# Plot the function
plt.plot(x_values, y_values, label='K(x)', color='orange',
linewidth=2)
plt.title('Plot of K(x) Function')
plt.xlabel('x')
plt.ylabel('K(x)')

```

```
plt.grid(True, linestyle='--', alpha=0.7)
plt.xlim(- (10 * .05), 10.5)
plt.ylim(- (.5 * .05), 0.5)
plt.legend()
plt.show()
```



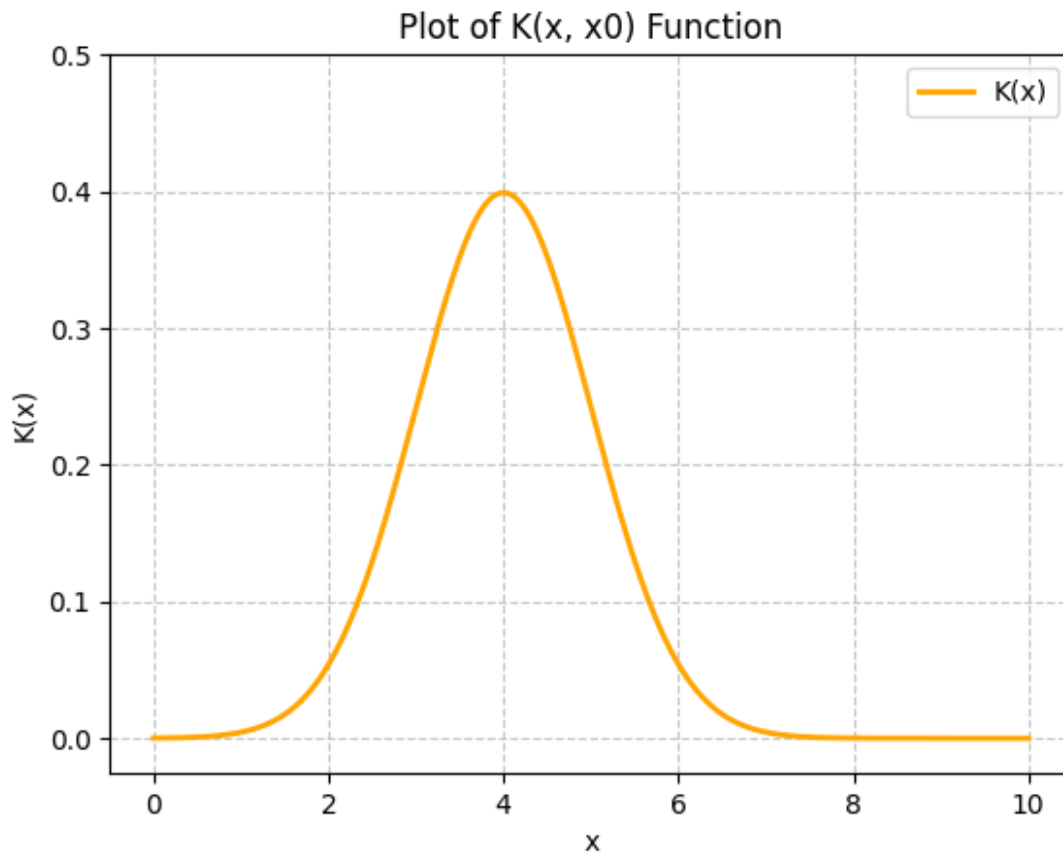
```
def K2(x, x0):
    return 1. / math.sqrt(2 * math.pi) * math.exp(-(x - x0)** 2 / 2)

x_values = np.linspace(0, 10, 500)
y_values = [K2(x, 4) for x in x_values]

# Plot the function
plt.plot(x_values, y_values, label='K(x)', color='orange',
         linewidth=2)
plt.title('Plot of K(x, x0) Function')
plt.xlabel('x')
plt.ylabel('K(x)')
plt.grid(True, linestyle='--', alpha=0.7)
plt.xlim(- (10 * .05), 10.5)
plt.ylim(- (.5 * .05), 0.5)
```

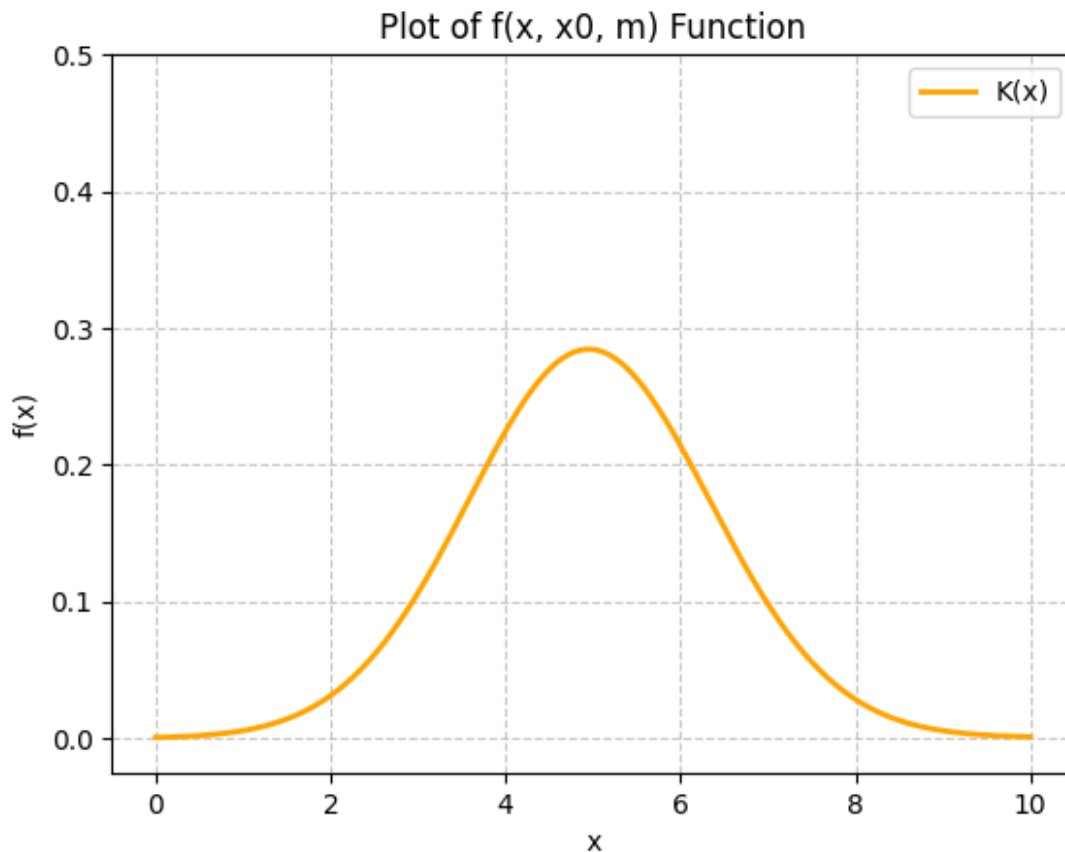


```
plt.legend()  
plt.show()
```



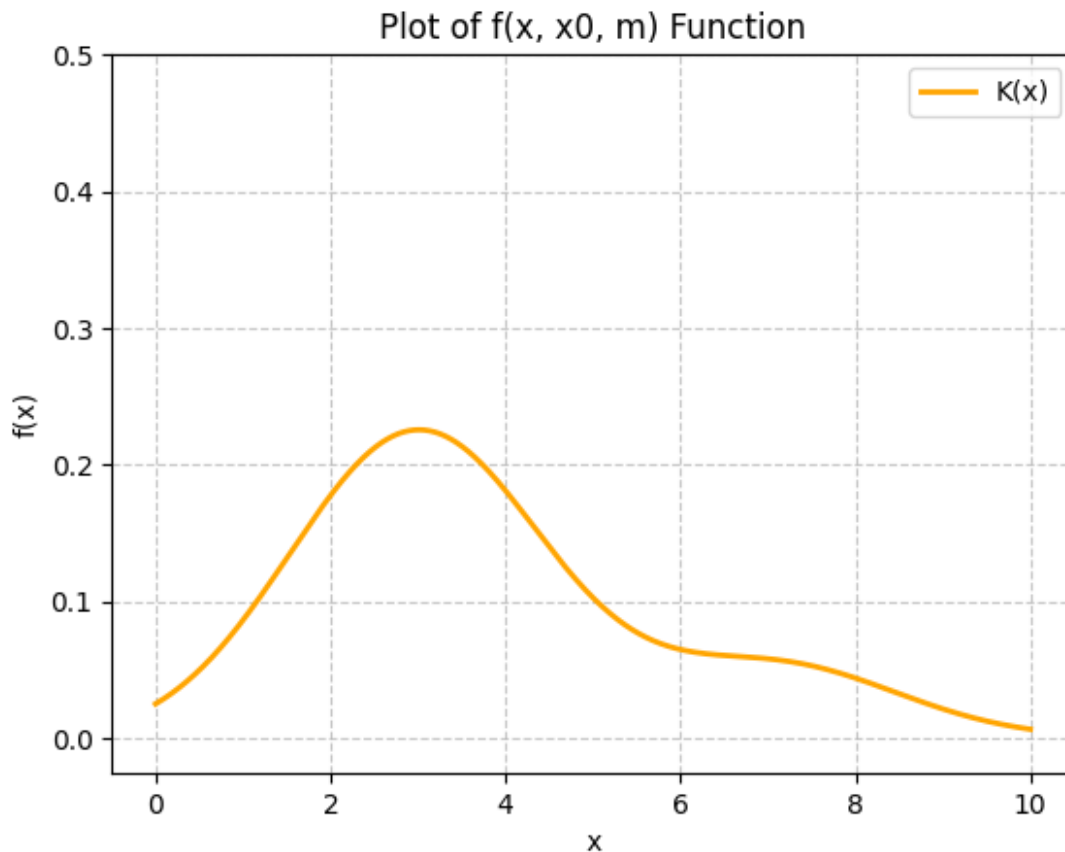
```
def f(x, x0, m=1000):  
    return 1./m * 1./math.sqrt(2 * math.pi) * sum(math.exp(-(x -  
element) ** 2 / 2) for element in x0)  
  
x_values = np.linspace(0, 10, 1000)  
x0 = np.random.normal(5, 1, 1000)  
y_values = [f(x, x0) for x in x_values]  
  
plt.plot(x_values, y_values, label='K(x)', color='orange',  
linewidth=2)  
plt.title('Plot of f(x, x0, m) Function')  
plt.xlabel('x')  
plt.ylabel('f(x)')  
plt.grid(True, linestyle='--', alpha=0.7)  
plt.xlim(- (10 * .05), 10.5)  
plt.ylim(- (.5 * .05), 0.5)
```

```
plt.legend()  
plt.show()
```



```
def fnew(x, x0, m=1000):  
    return 1./m * 1./math.sqrt(2 * math.pi) * sum(math.exp(-(x -  
element) ** 2 / 2) for element in x0)  
  
sample_size = 1000  
x0 = np.concatenate([np.random.normal(3, 1, int(0.8 * sample_size)),  
np.random.normal(7, 1, int(0.2 * sample_size))])  
  
x_values = np.linspace(0, 10, sample_size)  
y_values = [fnew(x, x0) for x in x_values]  
  
# Plot the function  
plt.plot(x_values, y_values, label='K(x)', color='orange',  
linewidth=2)  
plt.title('Plot of f(x, x0, m) Function')  
plt.xlabel('x')  
plt.ylabel('f(x)')  
plt.grid(True, linestyle='--', alpha=0.7)  
plt.xlim(- (10 * .05), 10.5)  
plt.ylim(- (.5 * .05), 0.5)
```

```
plt.legend()
plt.show()
```



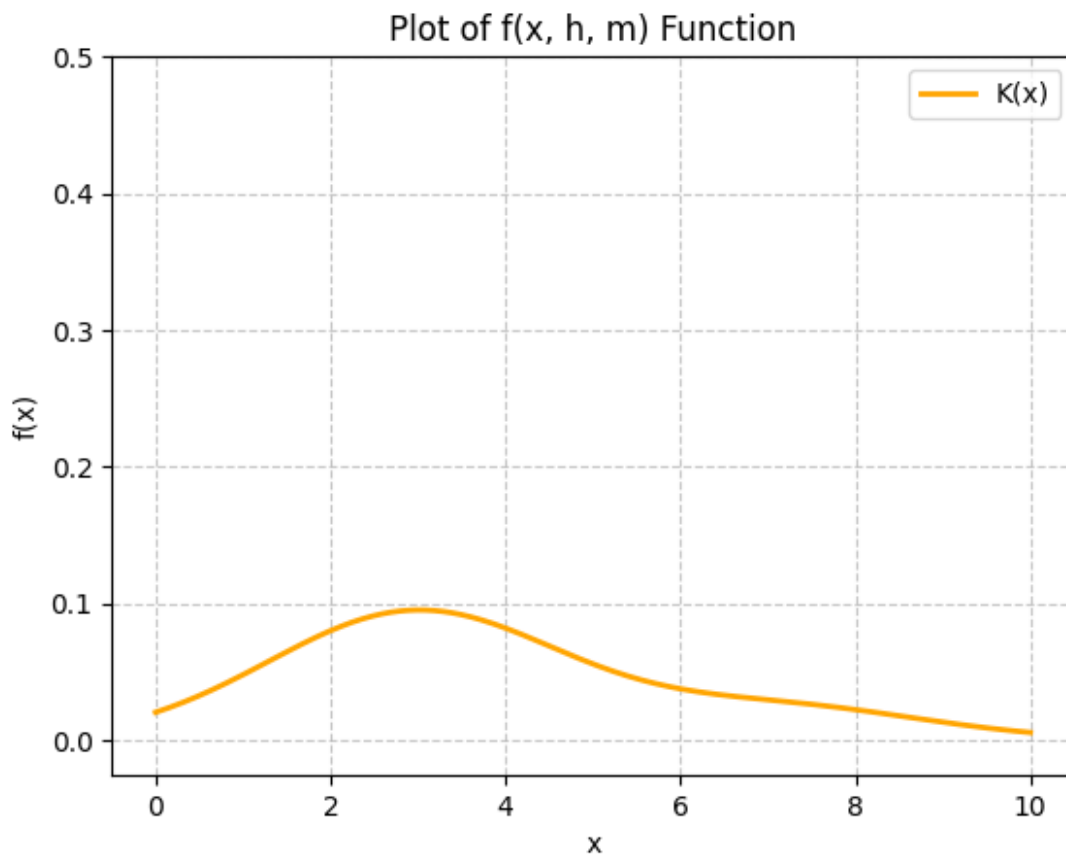
```
def fh(x, x0, h, m=1000):
    return 1./ (m * h) * 1./math.sqrt(2 * math.pi) * sum(math.exp(-((x
- element) / h) ** 2 / 2) for element in x0)

sample_size = 500
x0 = np.concatenate([np.random.normal(3, 1, int(0.8 * sample_size)),
np.random.normal(7, 1, int(0.2 * sample_size))])

x_values = np.linspace(0, 10, sample_size)
y_values = [fh(x, x0, 1.337) for x in x_values]

# Plot the function
plt.plot(x_values, y_values, label='K(x)', color='orange',
linewidth=2)
plt.title('Plot of f(x, h, m) Function')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.grid(True, linestyle='--', alpha=0.7)
plt.xlim(- (10 * .05), 10.5)
plt.ylim(- (.5 * .05), 0.5)
```

```
plt.legend()  
plt.show()
```



jak obliczyc h

jest na mailu w ksiazce i rownania - sigma

Cell In[44], line 1

jak obliczyc h
^

SyntaxError: invalid syntax