Numerical Methods

Lab 3

Maryna Borovyk

Task 1. Finding machine precision

1st method using sys

In [360…
```
import sys
print(sys.float_info.epsilon)
```

```
2.220446049250313e-16
```

2nd method using numpy

In [361…
```
import numpy as np
print(np.finfo(float).eps)
```

```
2.220446049250313e-16
```

Task 2

In [362…
```
import numpy as np
import math
import matplotlib.pyplot as plt
```

1. Generating 100 logarithmically spaced numbers between 1e-300 and 1e+300

In [363…
```
x_values = np.logspace(-300, 300, num=100)
```

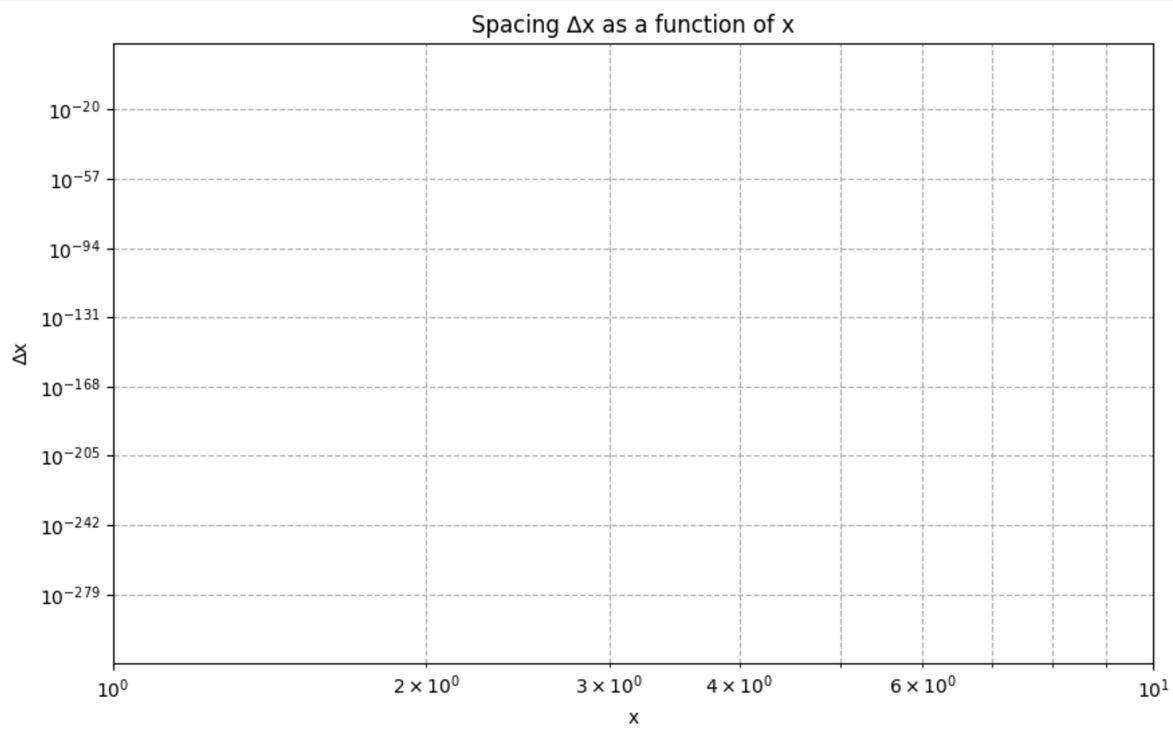
2. Calculating the spacing Δx for each x using math.nextafter

In [364…
```
delta_x = [math.nextafter(x, x + 1) - x for x in x_values]
```

3. Plotting the results

In [365…
```
plt.figure(figsize=(10, 6))
plt.style.use('seaborn-v0_8-deep')
plt.loglog(x_values, delta_x, marker='o', color='g', linestyle='none')
plt.xlabel('x')
plt.ylabel('Δx')
plt.title('Spacing Δx as a function of x')
plt.grid(True, which="both", ls="--")
plt.show()
```

```
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/matplotlib/scale.py:255: RuntimeWarning: overflow encountered in power
  return np.power(self.base, values)
```



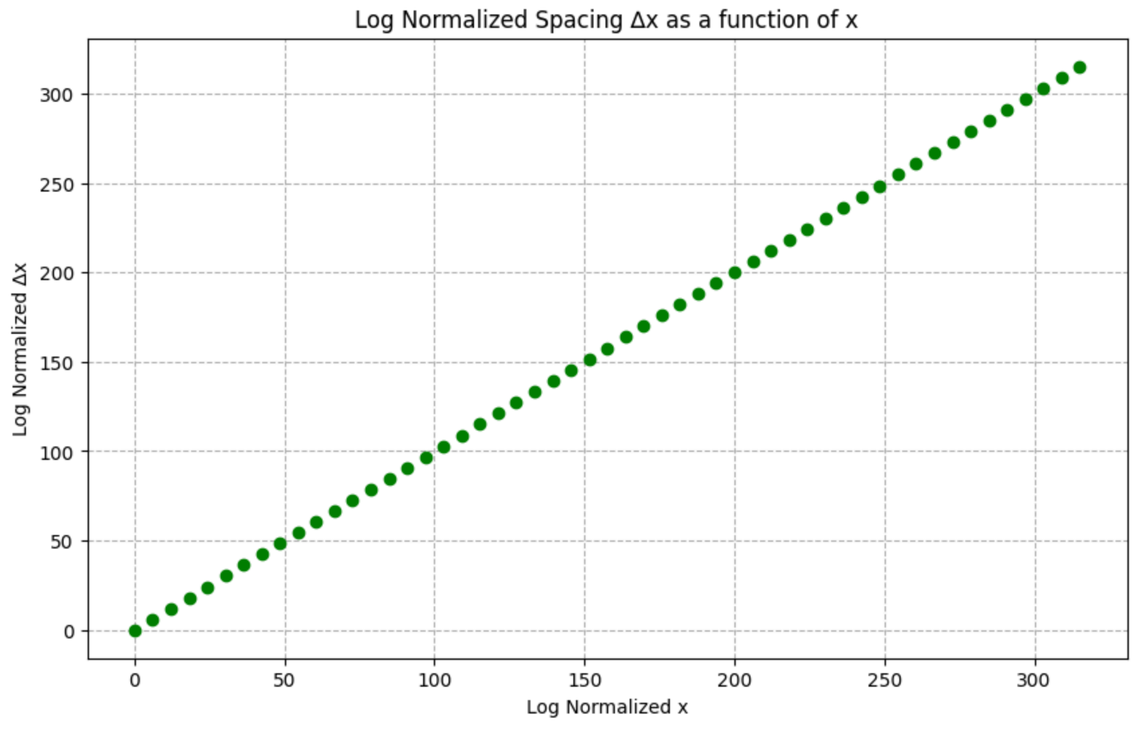
Normalizing values to avoid overflow and plot again

In [366…
```
filtered_delta_x = [dx for dx in delta_x if dx > 0]
filtered_x_values = [x for x, dx in zip(x_values, delta_x) if dx > 0]

log_x_values = np.log10(filtered_x_values)
log_delta_x = np.log10(filtered_delta_x)

log_x_values_normalized = log_x_values - log_x_values[0]
log_delta_x_normalized = log_delta_x - log_delta_x[0]

plt.figure(figsize=(10, 6))
plt.style.use('seaborn-v0_8-deep')
plt.plot(log_x_values_normalized, log_delta_x_normalized, marker='o', color='g', linestyle='none')
plt.xlabel('Log Normalized x')
plt.ylabel('Log Normalized Δx')
plt.title('Log Normalized Spacing Δx as a function of x')
plt.grid(True, which="both", ls="--")
plt.show()
```



Task 3

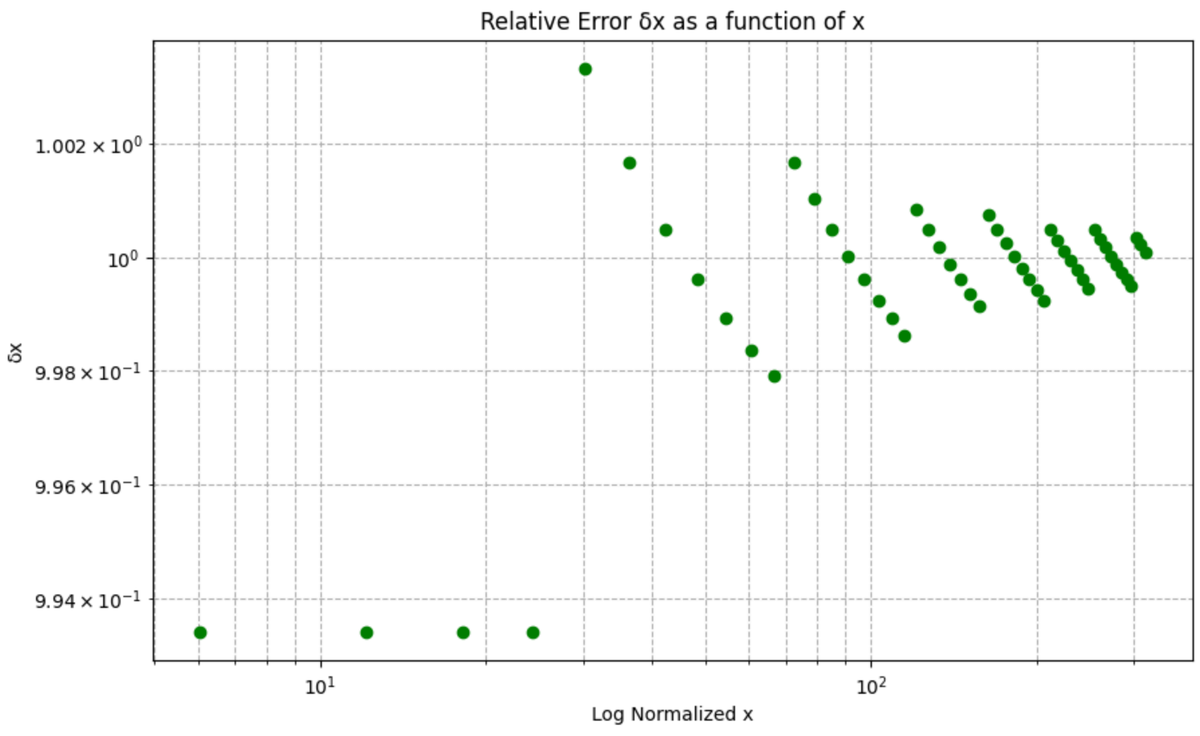
Calculate the relative error δx

In [367…
```
valid_indices = [i for i, x in enumerate(log_x_values_normalized) if x != 0 and not np.isnan(x)]
filtered_log_x_values_normalized = [log_x_values_normalized[i] for i in valid_indices]
filtered_log_delta_x_normalized = [log_delta_x_normalized[i] for i in valid_indices]


relative_error = [dx / x for dx, x in zip(filtered_log_delta_x_normalized, filtered_log_x_values_normalized)]
```

Plotting the results

In [368…
```
plt.figure(figsize=(10, 6))
plt.loglog(filtered_log_x_values_normalized, relative_error, marker='o', color='g', linestyle='none')
plt.xlabel('Log Normalized x')
plt.ylabel('δx')
plt.title('Relative Error δx as a function of x')
plt.grid(True, which="both", ls="--")
plt.show()
```


Relative Error δx as a function of x
1.002×10^0
10^0
9.98×10^{-1}
9.96×10^{-1}
9.94×10^{-1}
δx
10^1
10^2
Log Normalized x