StefinRacho-Homework3

September 26, 2024

0.1 CS4010 Homework 3

0.1.1 Due Friday, 9/27/2024

0.1.2 Create a New iPython (Jupyter) notebook. Name the notebook FirstAndLastName_Homework3 and save it before you start working

0.1.3 To submit, export or print your notebook as a pdf, with all outputs visible. Upload both the pdf and a copy of your notebook (.ipynb) in Canvas.

0.1.4 Ch 2.6-3.3

Data files needed for problems 3.1, 3.2, and 3.3 can be found on Canvas or on the textbook website.

Be sure to include "from matplotlib import pyplot as plt" at the beginning of your code to use for plotting.

1. The Choose Function is defined by factorial functions as follows, $\binom{n}{x} = \frac{n!}{k!(n-k)!}$, when k >= 1, or 1 when k = 0.

a) Write a factorial function using a for loop to calculate n! where the function take n as an argument. (See page 76 for an example.)

[59]:
```
# Part 1a
def factorial(n):
    product = 1
    for i in range(1, n + 1):
        product *= i

    return product
```

b) Check that your factorial function works with n = 4, 5 and 6.

[62]:
```
# Part 1b
print(f"4 factorial = {factorial(4)}")
print(f"5 factorial = {factorial(5)}")
print(f"6 factorial = {factorial(6)}")
```

```
4 factorial = 24
5 factorial = 120
6 factorial = 720
```

c) Write a choose function that takes n and k as arguments. This function should use multiple calls to your factorial function to calculate the Choose Function.

[65]:
```
# Part 1c
def choose(n, k):
    return factorial(n) / (factorial(k) * factorial(n - k))

choose(8, 4)
```

[65]:
```
70.0
```

d) Upgrade your factorial fuction using the recursion method in Exercise 2.13. Make sure to give it a different name than your factorial function from part a).

[68]:
```
# Part 1d
def factorial_upgraded(n):
    if (n == 1):
        return 1

    return n * factorial_upgraded(n - 1)

print(f"4 factorial = {factorial_upgraded(4)}")
print(f"5 factorial = {factorial_upgraded(5)}")
print(f"6 factorial = {factorial_upgraded(6)}")
```

```
4 factorial = 24
5 factorial = 120
6 factorial = 720
```

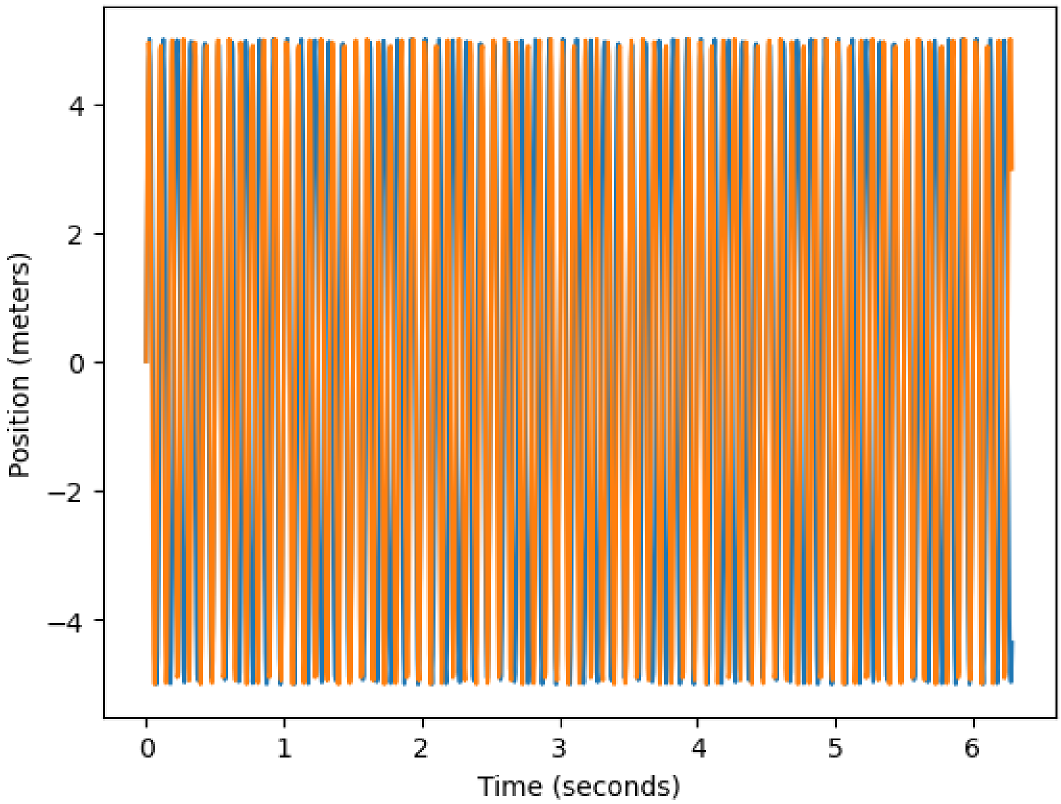
2. Plotting:

a) Create plots of $y_1(t) = Asin(2\pi f_1 t)$ and $y_2(t) = Asin(2\pi f_2 t)$ on the same axis with where t is time with units of seconds, y is position with units of meters. A = 5.0 and $f_1 = 10Hz, f_2 = 12Hz$. Make the range using linspace from 0 to 2*pi with a 1000 points. Be sure to label your axes.

[3]:
```
import numpy as np
from matplotlib import pyplot as plt
```

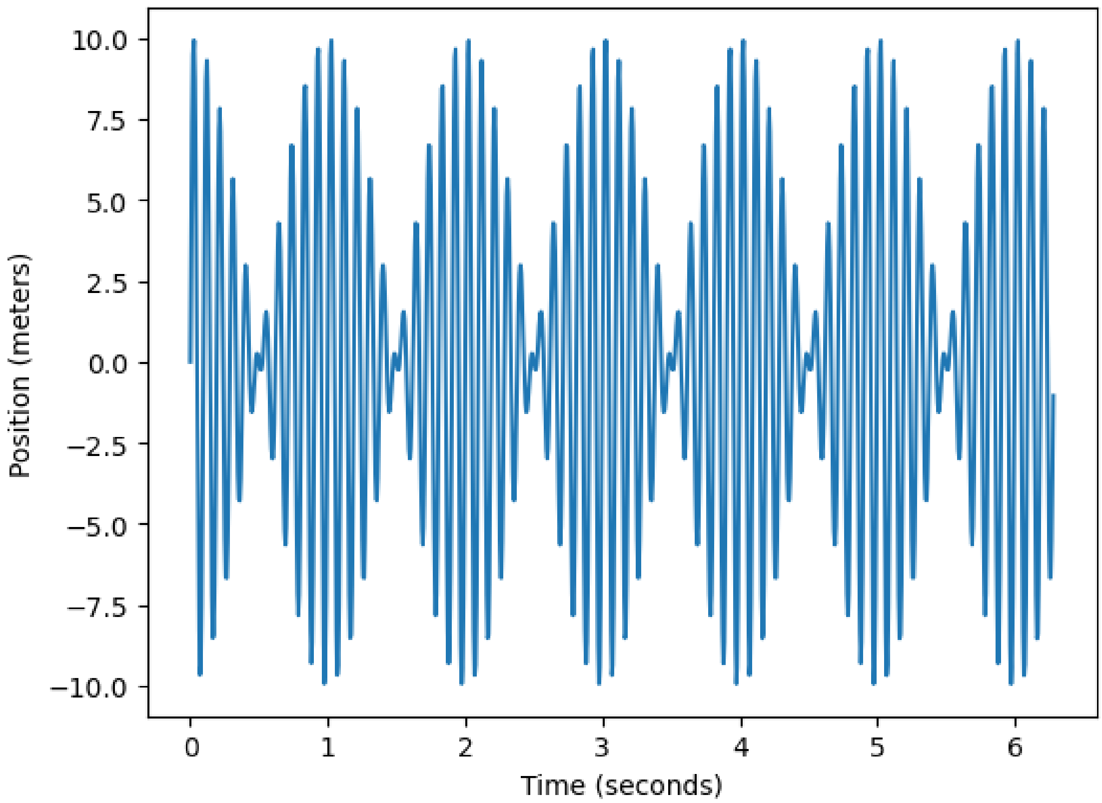
[119]:
```
A = 5.0
f1 = 10
f2 = 12
t = np.linspace(0, 2 * np.pi, 1000)

y1 = A * np.sin(2 * np.pi * f1 * t)
y2 = A * np.sin(2 * np.pi * f2 * t)

plt.plot(t, y1)
plt.plot(t, y2)
```

```
plt.xlabel("Time (seconds)")
plt.ylabel("Position (meters)")

plt.show()
```



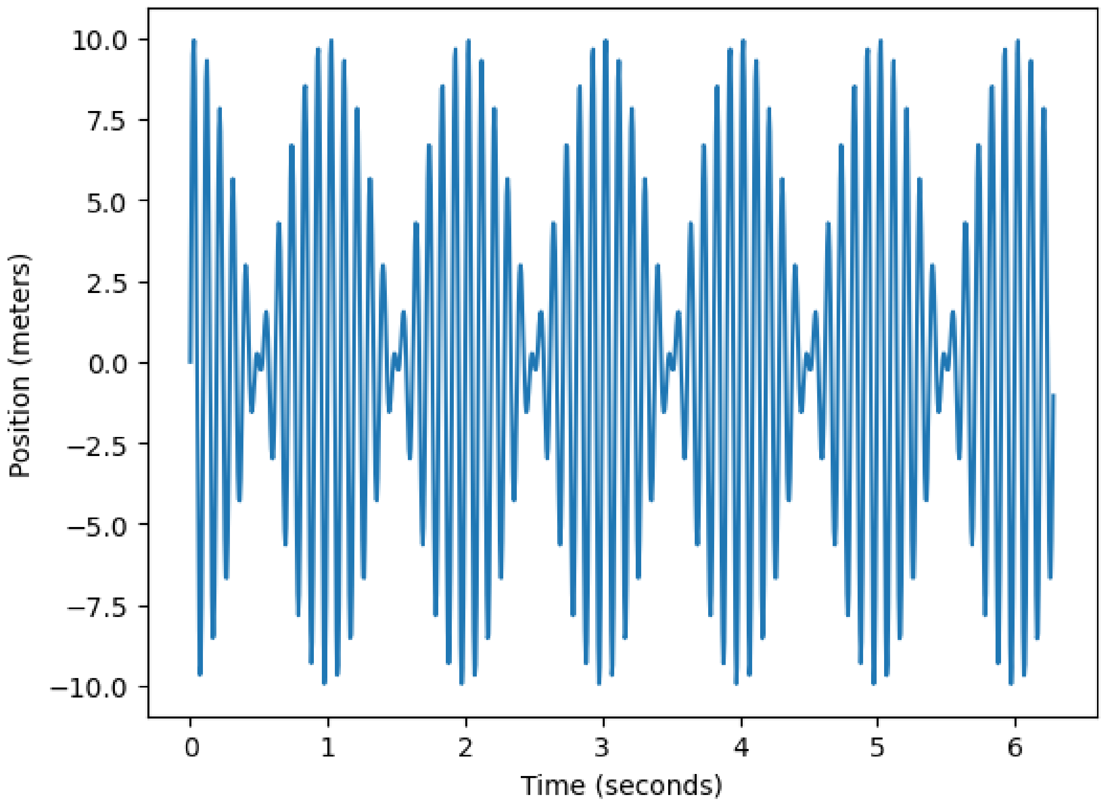
b) Create a plot of y1+y2 with the same parameters above.

[123]:
```
y3 = y1 + y2

plt.plot(t, y3)

plt.xlabel("Time (seconds)")
plt.ylabel("Position (meters)")

plt.show()
```

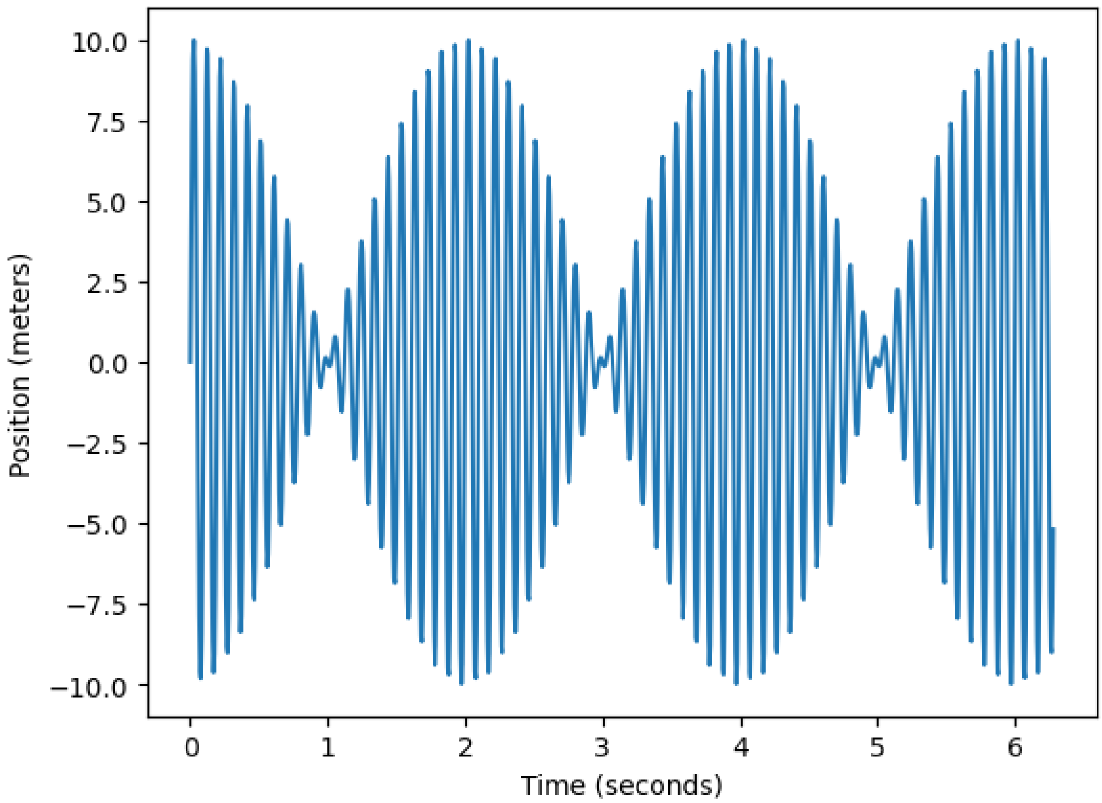


c) Repeat again with a beat frequency of $\Delta f = 1Hz, 0.5Hz, 0.1Hz$. Use multiple code boxes by copying and pasting the codes with different Δf's

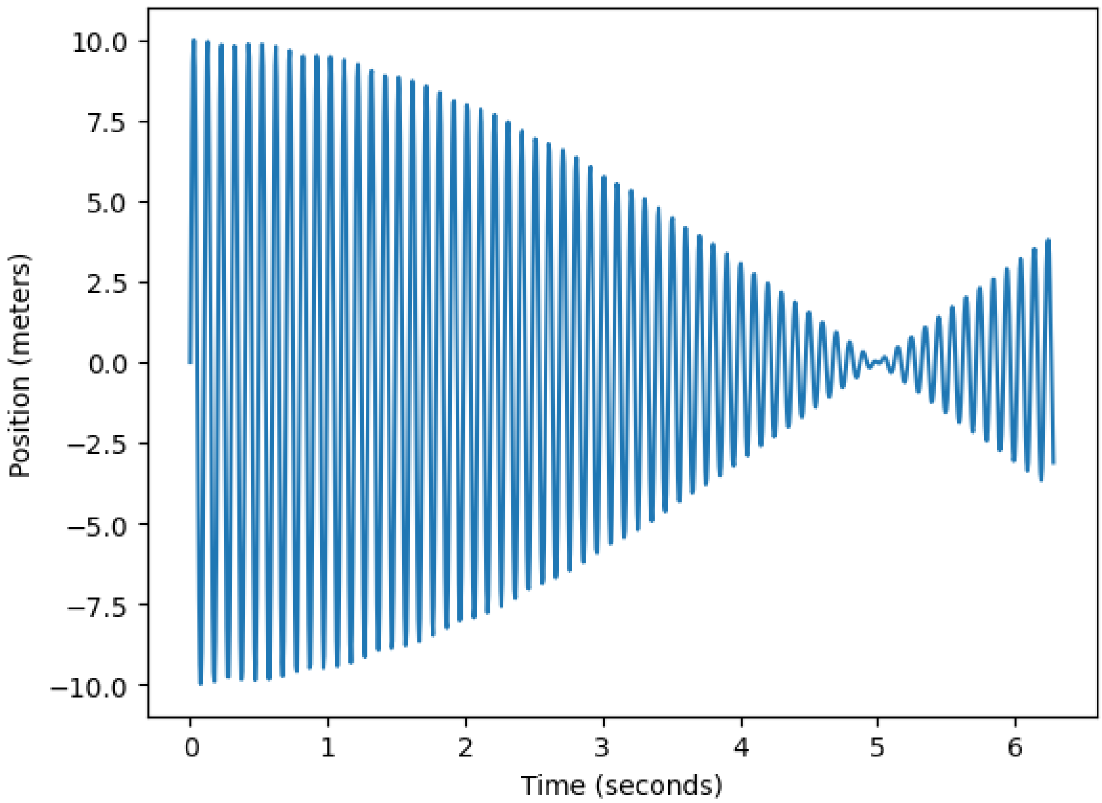
[125]:
```
deltaF = 1
f2 = f1 + deltaF

y1 = A * np.sin(2 * np.pi * f1 * t)
y2 = A * np.sin(2 * np.pi * f2 * t)
y3 = y1 + y2

plt.plot(t, y3)

plt.xlabel("Time (seconds)")
plt.ylabel("Position (meters)")

plt.show()
```



[133]:
```
deltaF = 0.5
f2 = f1 + deltaF

y1 = A * np.sin(2 * np.pi * f1 * t)
y2 = A * np.sin(2 * np.pi * f2 * t)
y3 = y1 + y2

plt.plot(t, y3)

plt.xlabel("Time (seconds)")
plt.ylabel("Position (meters)")

plt.show()
```

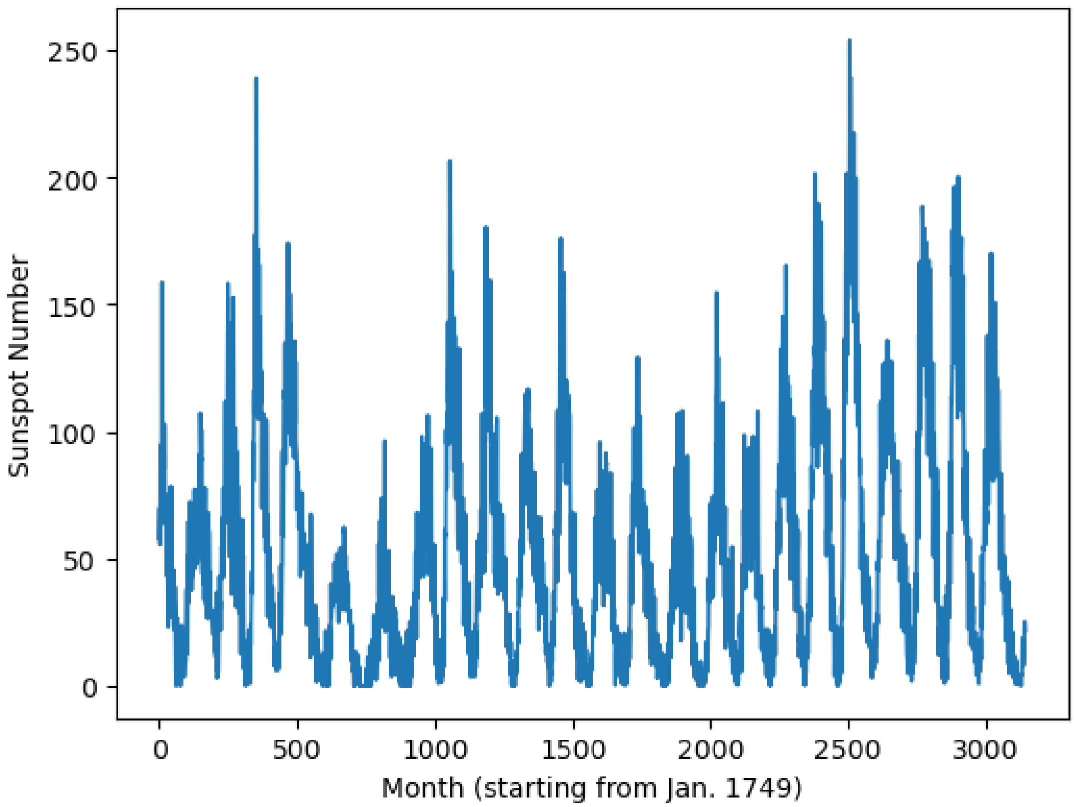


[131]:
```
deltaF = 0.1
f2 = f1 + deltaF

y1 = A * np.sin(2 * np.pi * f1 * t)
y2 = A * np.sin(2 * np.pi * f2 * t)
y3 = y1 + y2

plt.plot(t, y3)

plt.xlabel("Time (seconds)")
plt.ylabel("Position (meters)")

plt.show()
```

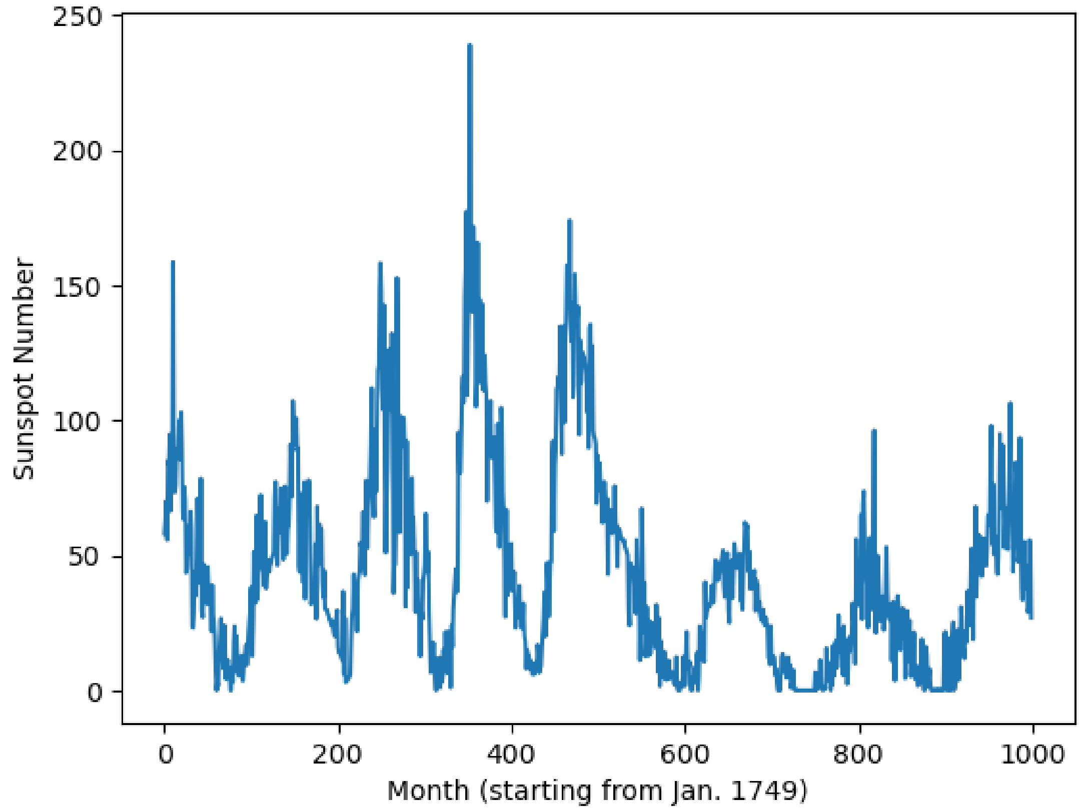


3. Exercise 3.1

[28]:
```
# Part 3a
data = np.loadtxt("sunspots.txt")
months = data[:, 0]
sunspots = data[:, 1]

plt.plot(months, sunspots)

plt.xlabel("Month (starting from Jan. 1749)")
plt.ylabel("Sunspot Number")

plt.show()
```

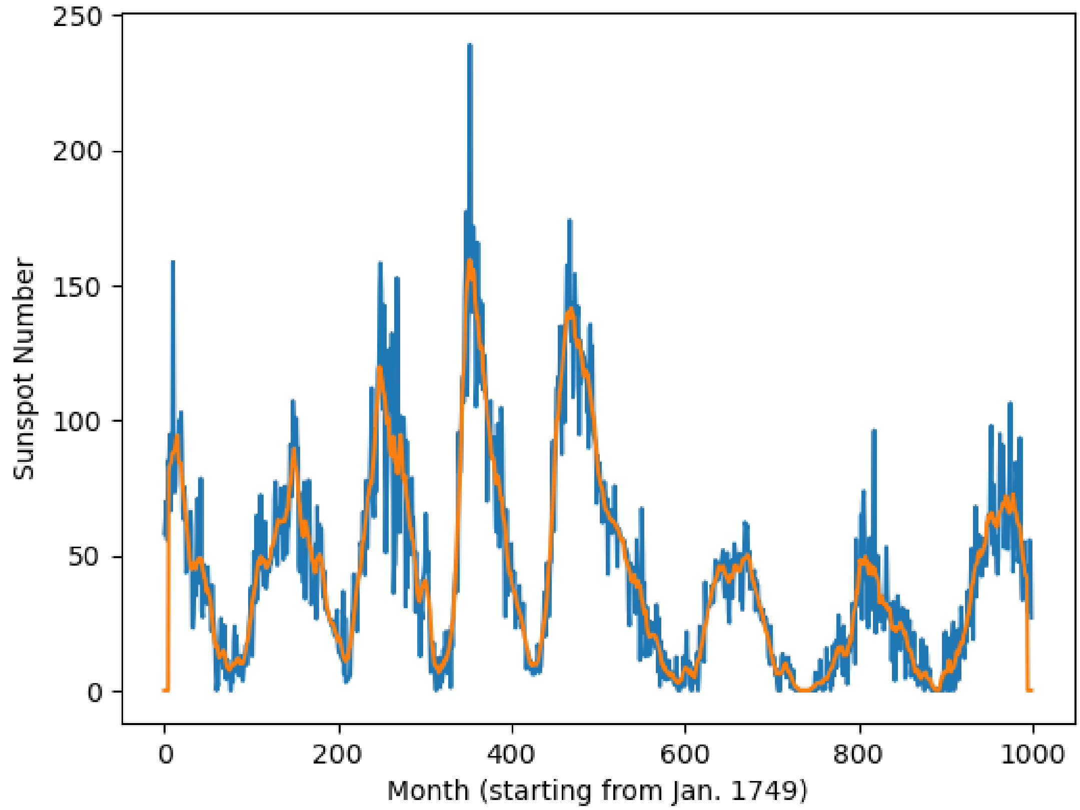


[30]:
```
# Part 3b
months = months[:1000]
sunspots = sunspots[:1000]

plt.plot(months, sunspots)

plt.xlabel("Month (starting from Jan. 1749)")
plt.ylabel("Sunspot Number")

plt.show()
```



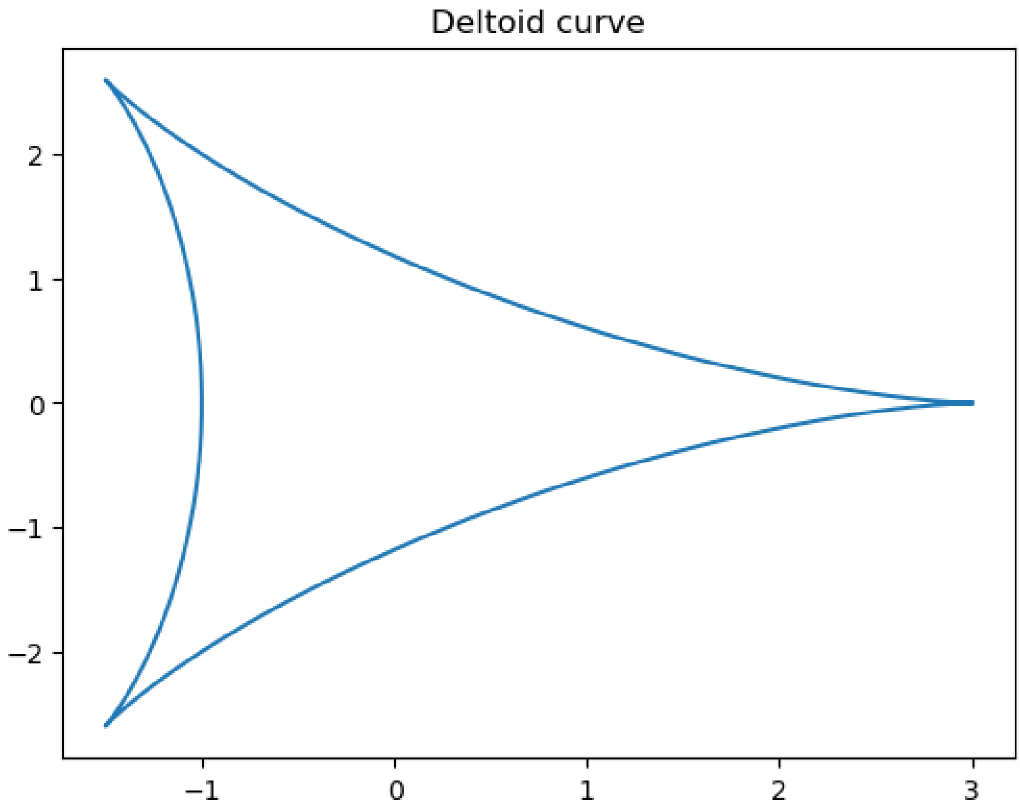
[54]:

```
plt.plot(months, sunspots)

r = 5
running_average = np.zeros(len(sunspots))

for k in range(r, len(sunspots) - r):
    running_average[k] = np.mean(sunspots[k - r:k + r + 1])

plt.xlabel("Month (starting from Jan. 1749)")
plt.ylabel("Sunspot Number")

plt.plot(months, running_average)

plt.show()
```

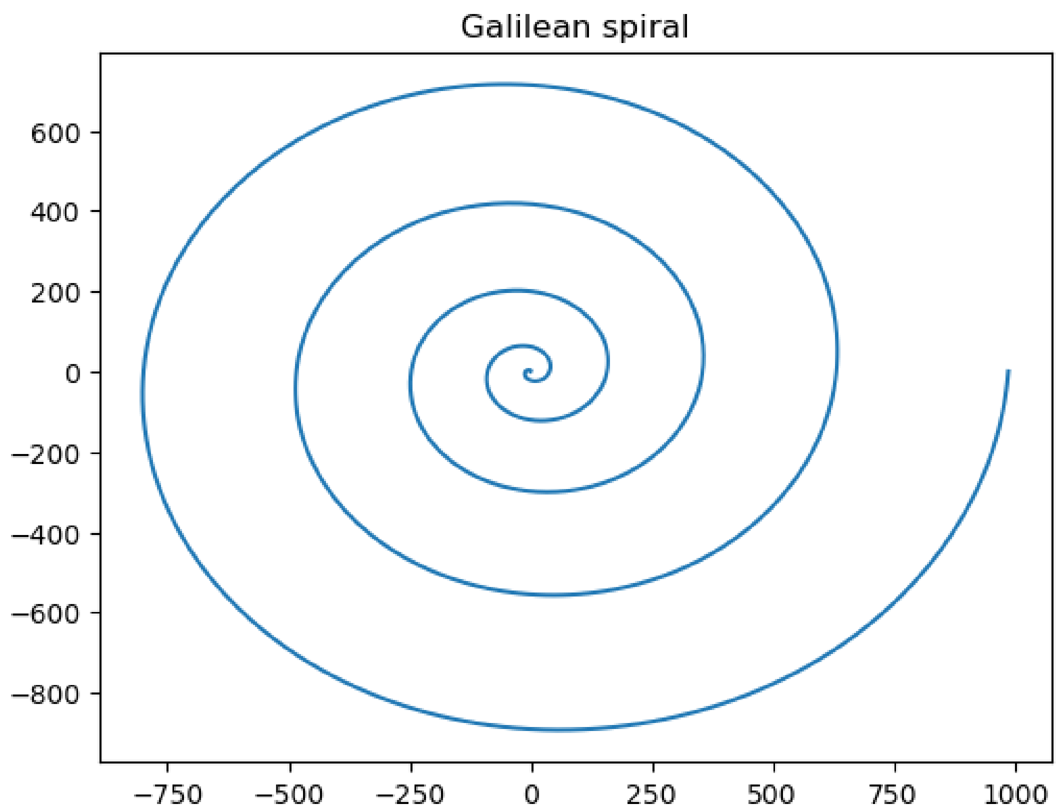


4. Exercise 3.2

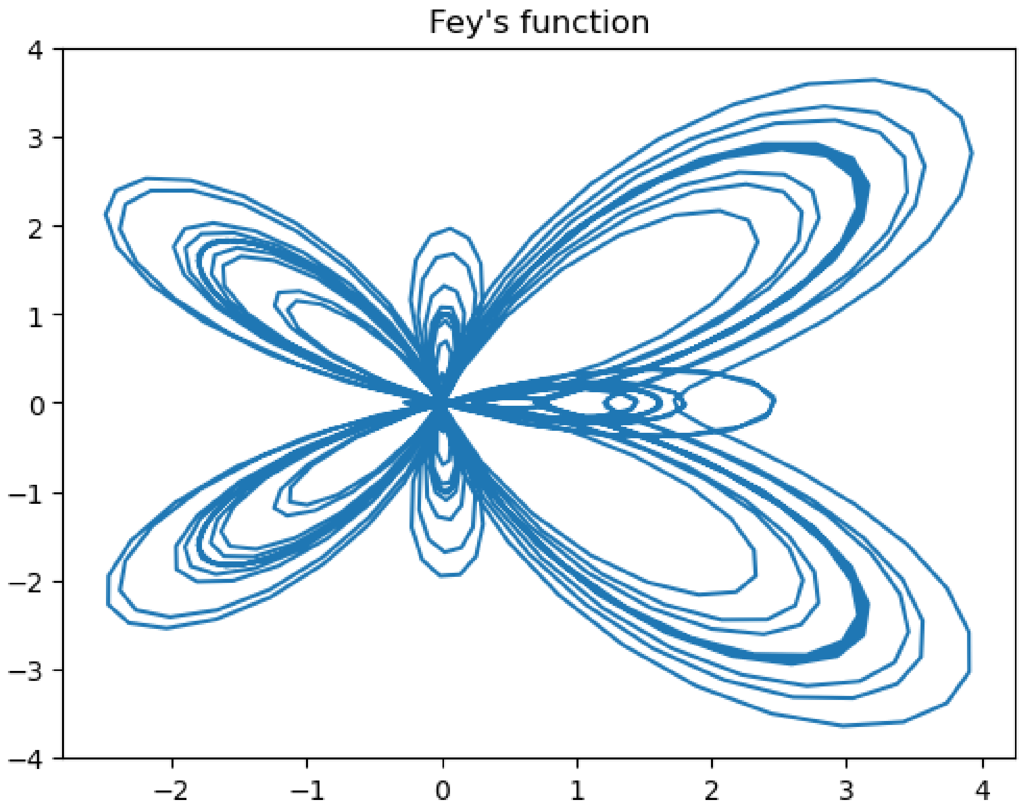
[17]:
```
theta = np.linspace(0, 2 * np.pi, 1000)

x = 2 * np.cos(theta) + np.cos(2 * theta)
y = 2 * np.sin(theta) - np.sin(2 * theta)

plt.plot(x, y)
plt.title("Deltoid curve")
plt.show()
```



[15]:
```
theta = np.linspace(0, 10 * np.pi, 1000)
r = theta**2

x = r * np.cos(theta)
y = r * np.sin(theta)

plt.plot(x, y)
plt.title("Galilean spiral")
plt.show()
```

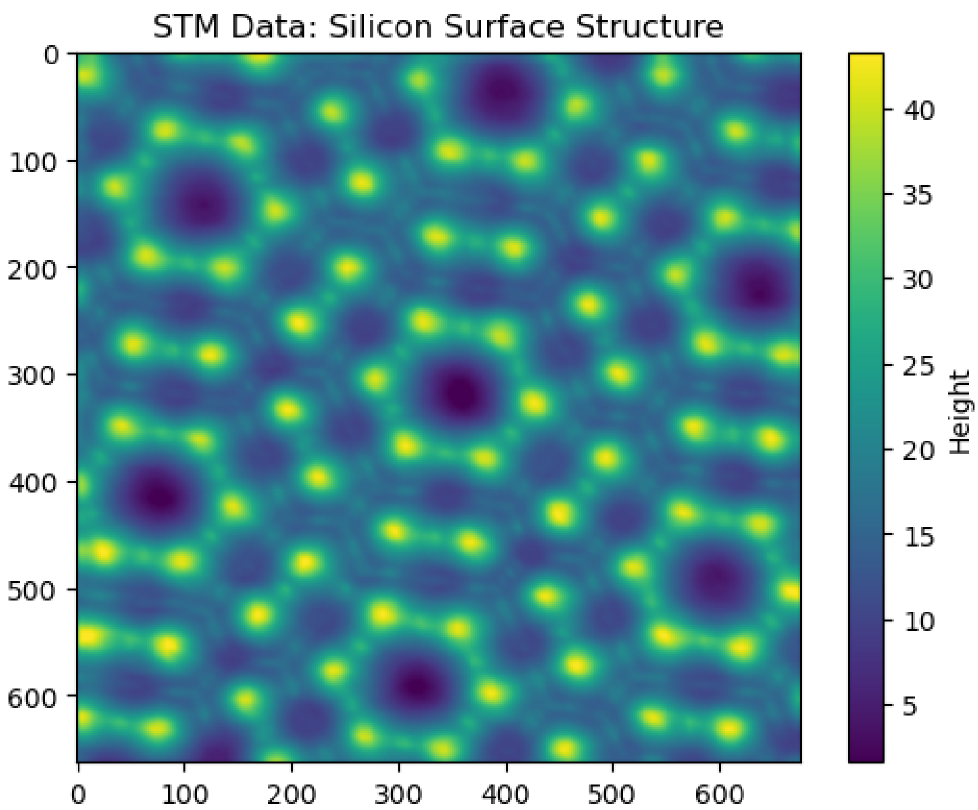


[13]:
```
theta = np.linspace(0, 24 * np.pi, 1000)
r = np.e**(np.cos(theta)) - 2 * np.cos(4 * theta) + np.sin(theta / 12)**5

x = r * np.cos(theta)
y = r * np.sin(theta)

plt.plot(x, y)
plt.title("Fey's function")
plt.show()
```

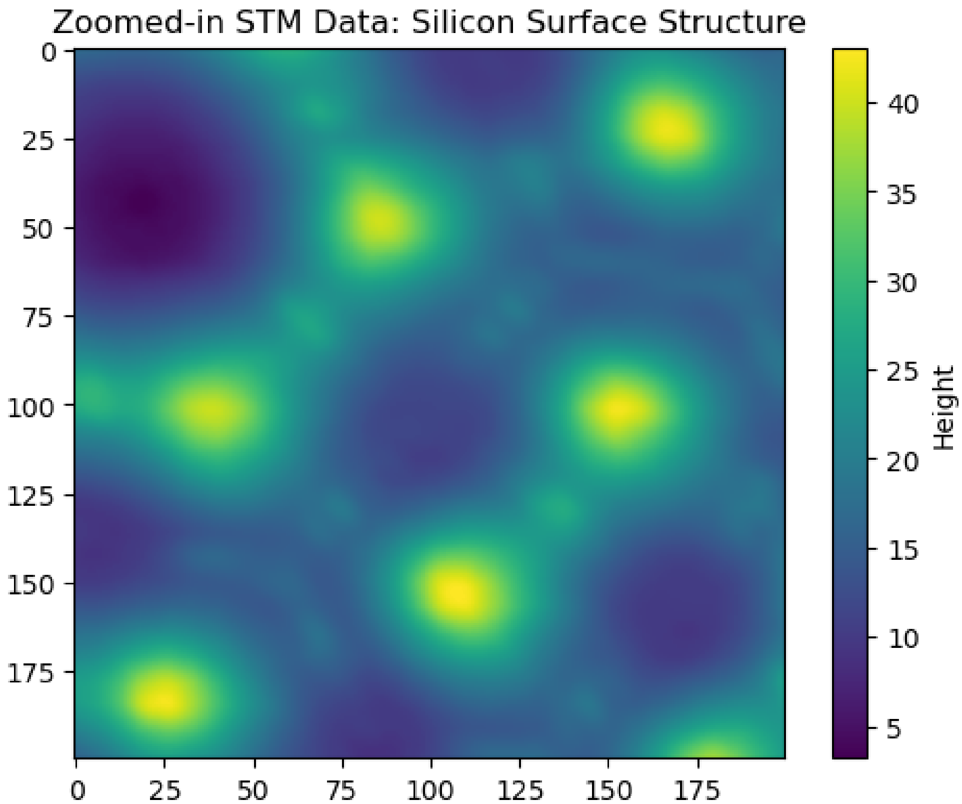


5. Exercise 3.3

[11]:
```
from pylab import imshow,show

data = np.loadtxt("stm.txt")
imshow(data)
plt.colorbar(label='Height')
plt.title('STM Data: Silicon Surface Structure')
show()
```



[9]:
```
y1, y2 = 100, 300
x1, x2 = 100, 300
zoomed_data = data[y1:y2, x1:x2]

imshow(zoomed_data)
plt.colorbar(label='Height')
plt.title('Zoomed-in STM Data: Silicon Surface Structure')
show()
```


Zoomed-in STM Data: Silicon Surface Structure
0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175
40
35
30
25
20
15
10
5
Height

[ ]: