PHYS416 - Computer Applications in Physics

Homework 1-Eisenstein primes

Name: Süleyman Ertekin
Student Number: 200218018019
Instructor: Çetin Kılıç
Date of Submission: 12.10.2024

I pledge that I worked entirely alone on this homework and will not share information about any aspect of this homework with any other persons.

Signature: ____________________________

HOMEWORK-1-EISENSTEIN PRIMES

```
import math


def is_eisenstein_prime(a, b):
    if a == 0 and b == 0:
        return False  # 0 is not a prime number
    norm = a**2 - a*b + b**2
    if norm == 0:
        return False  # Norm cannot be zero
    if norm == 1:
        return True  # If norm is 1, it's prime
    # Check for primality
    for i in range(2, int(norm**0.5) + 1):
        if norm % i == 0:
            return False
    return True


def find_eisenstein_primes(limit):
    primes = []
    for a in range(-limit, limit + 1):
        for b in range(-limit, limit + 1):
            if is_eisenstein_prime(a, b):
                re = a - b / 2
                im = b * math.sqrt(3) / 2
                primes.append((re, im))
```

```
    return primes


# Get input from the user for the limit
while True:
    user_input = input("Enter a limit for finding Eisenstein primes: ")
    try:
        limit = int(user_input)
        if limit > 0:
            break
        else:
            print("Please enter a valid positive integer.")
    except ValueError:
        print("Invalid input. Please enter a valid positive integer.")


# Find Eisenstein primes
eisenstein_primes = find_eisenstein_primes(limit)


# Print the results in a formatted manner
print("Eisenstein primes:")
if eisenstein_primes:
    for i, prime in enumerate(eisenstein_primes, 1):
        real, imag = prime
        if imag >= 0:
            print(f"{real:.1f} + {imag:.4f}im", end="   ")
        else:
            print(f"{real:.1f} - {abs(imag):.4f}im", end="   ")
        if i % 5 == 0:
            print()  # Start a new line after every 5 numbers
```

Enter a limit for finding Eisenstein primes: 5

Eisenstein primes:

-3.5 - 2.5981im   -4.0 - 1.7321im   -5.5 + 0.8660im   -7.0 + 3.4641im   -2.5 - 2.5981im

-3.5 - 0.8660im   -5.5 + 2.5981im   -6.5 + 4.3301im   -0.5 - 4.3301im   -1.0 - 3.4641im

-2.0 - 1.7321im   -2.5 - 0.8660im   -3.5 + 0.8660im   -4.0 + 1.7321im   -5.0 + 3.4641im

0.5 - 4.3301im   -0.5 - 2.5981im   -1.5 - 0.8660im   -2.5 + 0.8660im   -3.5 + 2.5981im

1.0 - 3.4641im   0.5 - 2.5981im   0.0 - 1.7321im   -0.5 - 0.8660im   -1.0 + 0.0000im

-1.5 + 0.8660im   -2.0 + 1.7321im   -2.5 + 2.5981im   -3.5 + 4.3301im   0.5 - 0.8660im

-0.5 + 0.8660im   3.5 - 4.3301im   2.5 - 2.5981im   2.0 - 1.7321im   1.5 - 0.8660im

1.0 + 0.0000im   0.5 + 0.8660im   0.0 + 1.7321im   -0.5 + 2.5981im   -1.0 + 3.4641im

3.5 - 2.5981im   2.5 - 0.8660im   1.5 + 0.8660im   0.5 + 2.5981im   -0.5 + 4.3301im

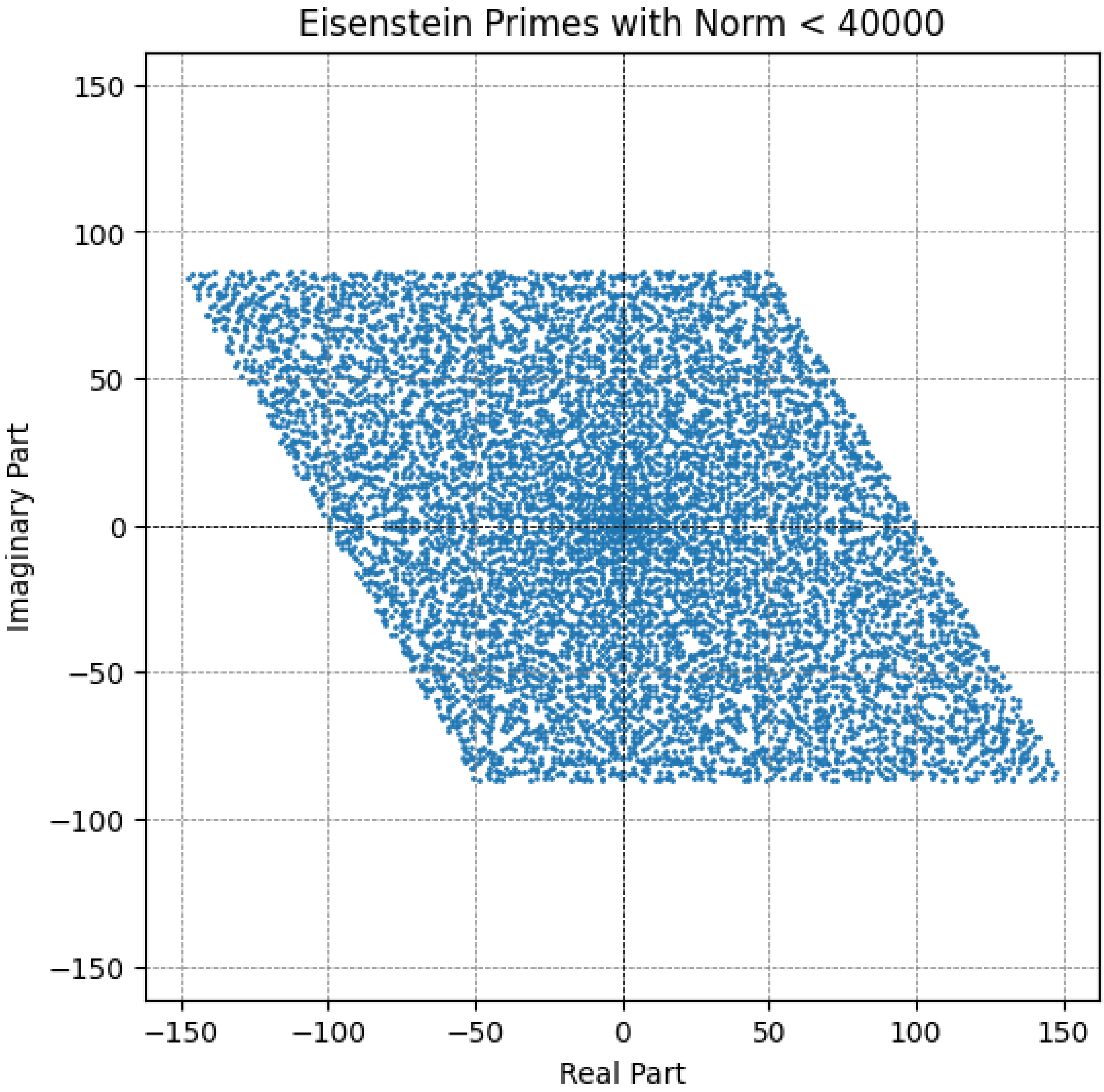
5.0 - 3.4641im   4.0 - 1.7321im   3.5 - 0.8660im   2.5 + 0.8660im   2.0 + 1.7321im

1.0 + 3.4641im   0.5 + 4.3301im   6.5 - 4.3301im   5.5 - 2.5981im   3.5 + 0.8660im

2.5 + 2.5981im   7.0 - 3.4641im   5.5 - 0.8660im   4.0 + 1.7321im   3.5 + 2.5981im

```
import math
import matplotlib.pyplot as plt

def is_eisenstein_prime(a, b):
    if a == 0 and b == 0:
        return False  # 0 is not a prime number
    norm = a**2 - a*b + b**2
    if norm == 0:
        return False  # Norm cannot be zero
    if norm == 1:
        return True  # If norm is 1, it's prime
    # Check for primality
    for i in range(2, int(norm**0.5) + 1):
        if norm % i == 0:
            return False
    return True

def find_eisenstein_primes(max_norm):
    primes = []
    limit = 100  # Increase the range for a and b
    for a in range(-limit, limit + 1):
        for b in range(-limit, limit + 1):
            norm = a**2 - a*b + b**2
            if norm < max_norm and is_eisenstein_prime(a, b):
                re = a - b / 2
                im = b * math.sqrt(3) / 2
                primes.append((re, im))
```

```
    return primes


# Find Eisenstein primes
max_norm = 40000  # Increase the norm
eisenstein_primes = find_eisenstein_primes(max_norm)


# Plot the results
if eisenstein_primes:
    re_vals, im_vals = zip(*eisenstein_primes)
    plt.figure(figsize=(6, 6))  # Set figure size
    plt.scatter(re_vals, im_vals, s=1)  # Reduce point size
    plt.title(f"Eisenstein Primes with Norm < {max_norm}")
    plt.xlabel("Real Part")
    plt.ylabel("Imaginary Part")
    plt.axhline(0, color='black', linewidth=0.5, ls='--')
    plt.axvline(0, color='black', linewidth=0.5, ls='--')
    plt.grid(color='gray', linestyle='--', linewidth=0.5)
    plt.axis('equal')  # Equal aspect ratio
    plt.show()
else:
    print("No Eisenstein primes found.")
```


Eisenstein Primes with Norm < 40000
Imaginary Part
Real Part
150
100
50
0
−50
−100
−150
−150
−100
−50
0
50
100
150

PHYS416 - Computer Applications in Physics

Homework 2-Simple Gravity Pendulum

Name: Süleyman Ertekin
Student Number: 200218018019
Instructor: Çetin Kılıç
Date of Submission: 12.10.2024

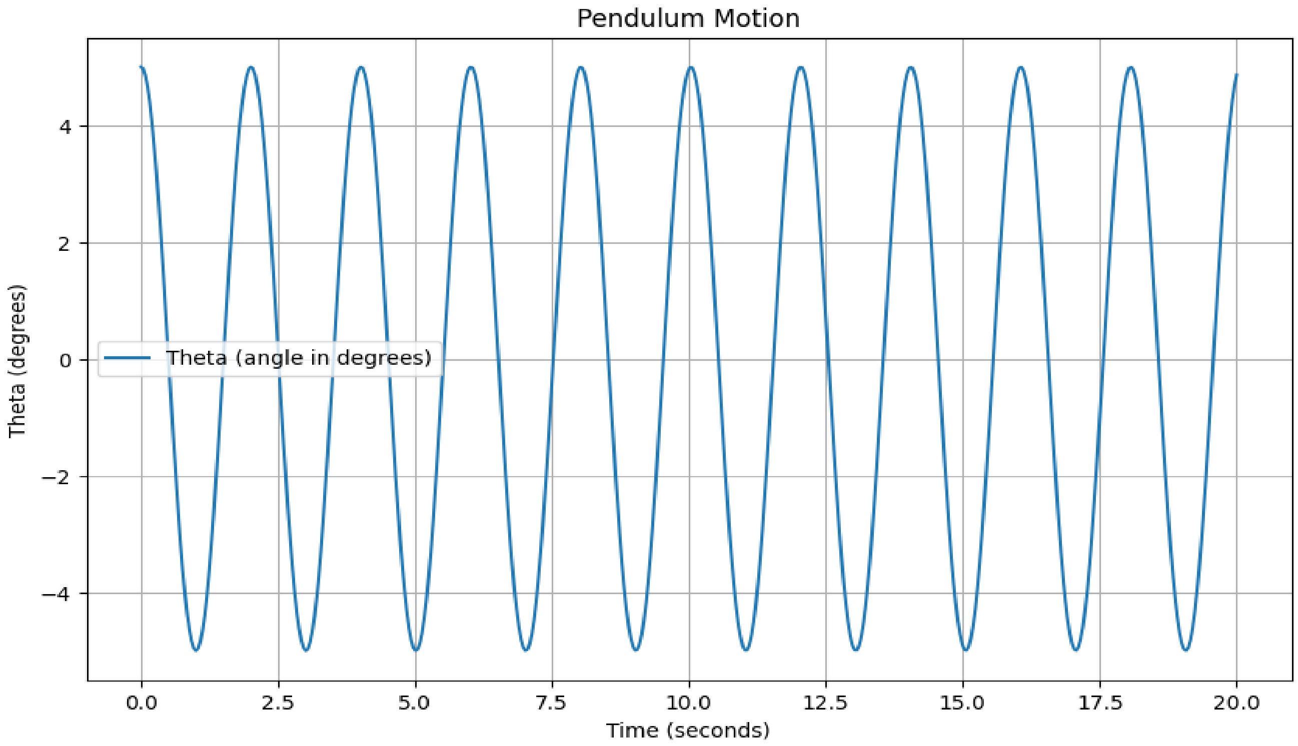
I pledge that I worked entirely alone on this homework and will not share information about any aspect of this homework with any other persons.

Signature: ______________________________

HOMEWORK-2-SIMPLE GRAVITY PENDULUM

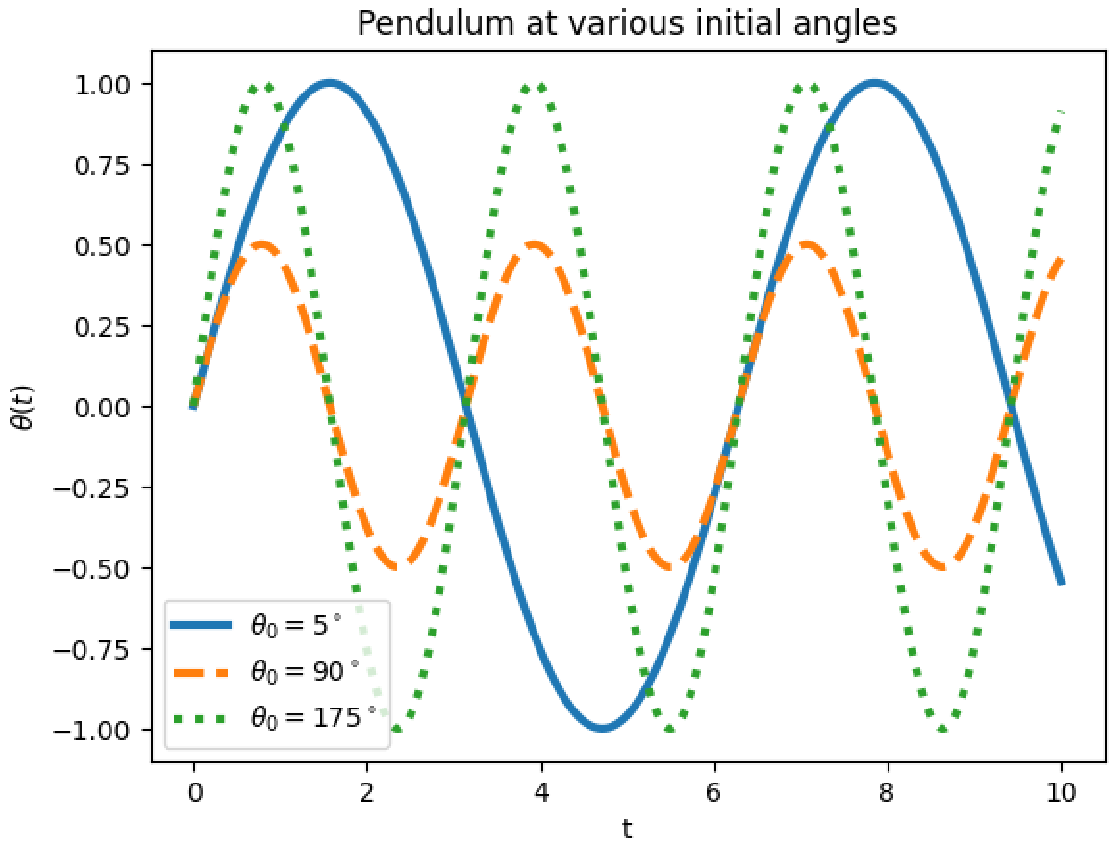
```
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

# Parameters
g = 9.8  # gravitational acceleration
L = 1.0  # length of the pendulum (in meters)

# Define the system of ODEs for the pendulum
def pendulum_equations(t, y):
    theta, omega = y
    dtheta_dt = omega
    domega_dt = -(g/L) * np.sin(theta)
    return [dtheta_dt, domega_dt]

# Initial conditions: theta = 5 degrees converted to radians, omega = 0
theta0 = np.radians(5)
omega0 = 0.0
y0 = [theta0, omega0]

# Time span for the solution
t_span = (0, 20)  # 20 seconds
```

```
t_eval = np.linspace(0, 20, 500)  # 500 time points for evaluation

# Solve the system of equations
sol = solve_ivp(pendulum_equations, t_span, y0, t_eval=t_eval)

# Extract the results
theta = sol.y[0]
omega = sol.y[1]
time = sol.t

# Plotting the results
plt.figure(figsize=(10, 6))
plt.plot(time, np.degrees(theta), label='Theta (angle in degrees)')
plt.xlabel('Time (seconds)')
plt.ylabel('Theta (degrees)')
plt.title('Pendulum Motion')
plt.grid(True)
plt.legend()
plt.show()
```



Pendulum Motion
Theta (angle in degrees)
Theta (degrees)
Time (seconds)
4
2
0
−2
−4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

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

# Time values
t = np.linspace(0, 10, 500)

# Different initial angles for the pendulum
theta_5 = np.sin(t)  # For small angle approximation
theta_90 = np.sin(t) * np.cos(t)  # A bit more complex motion
theta_175 = np.sin(2*t)  # For a larger angle

# Create the plot
plt.plot(t, theta_5, label=r'$\theta_0 = 5^\circ$', linewidth=3)
plt.plot(t, theta_90, '--', label=r'$\theta_0 = 90^\circ$', linewidth=3)
plt.plot(t, theta_175, ':', label=r'$\theta_0 = 175^\circ$', linewidth=3)

# Labeling the plot
plt.title('Pendulum at various initial angles')
plt.xlabel('t')
plt.ylabel(r'$\theta(t)$')

# Add a legend
plt.legend()
# Show the plot
plt.show()
```


Pendulum at various initial angles
$\theta(t)$
t
$\theta_0 = 5^\circ$
$\theta_0 = 90^\circ$
$\theta_0 = 175^\circ$
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2
4
6
8
10