ESA614: Computational Astrophysics
Assignment # 1

Name: Abraham Mathews
ID: SC22M077
Program: MS Astronomy and Astrophysics

---

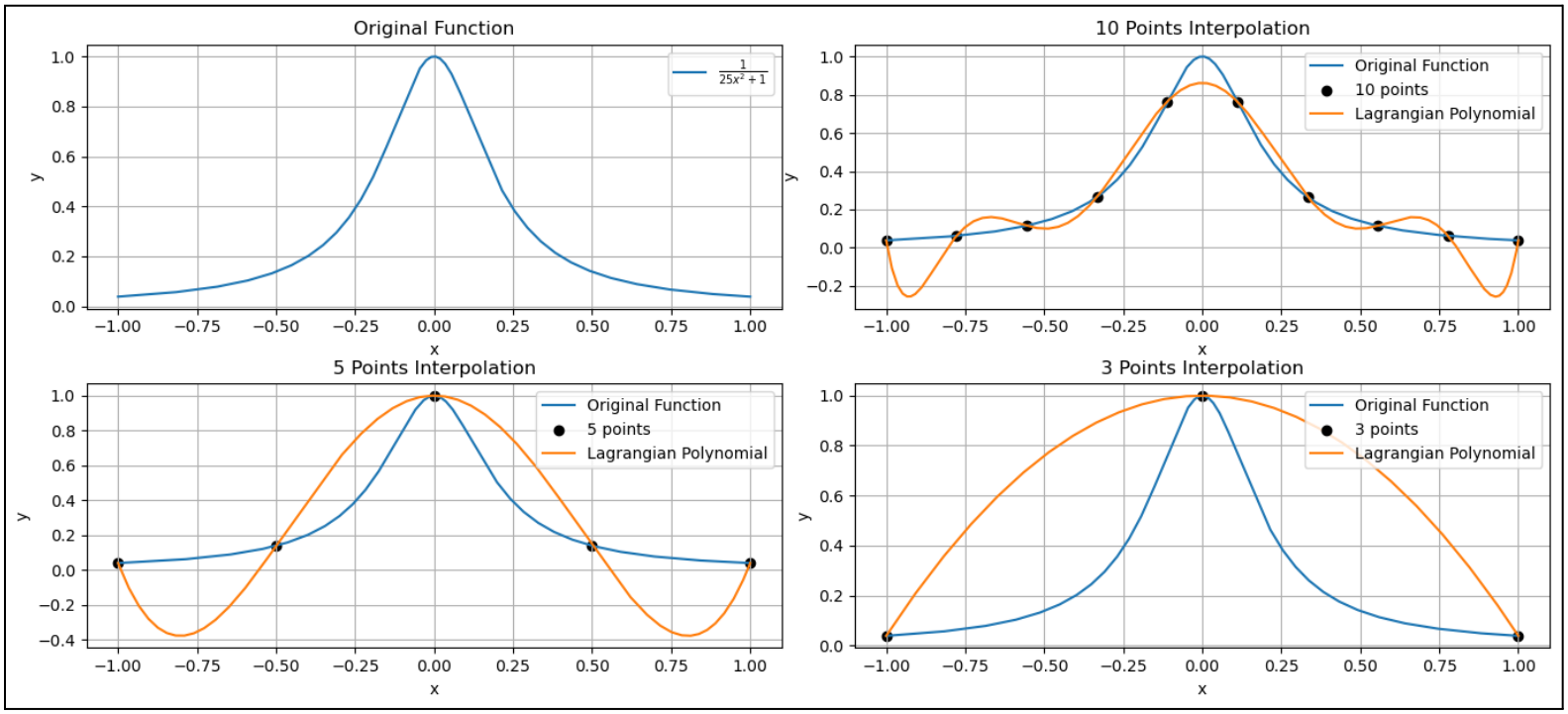
1. Use the Lagrange interpolation formula, and obtain a 10-point interpolation polynomial to approximate the function $f(x) = \frac{1}{25x^2 + 1}$ in the region -1≤ x ≤ 1. Plot the function vs the approximation. Try the approximation for 3-point and 5-point polynomials. Write a short note on your results.

Python Code:

```
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('fast')


#Lagrange Interpolation
def lag(x,y,n_sample):
  L = np.poly1d([0.0])
  for j in np.arange(len(x)):
    p = np.poly1d([1])
    for i in np.arange(len(x)):
      if(i != j):
        p = p*np.poly1d([[(1.0)/(x[j]-x[i])),((-x[i])/(x[j]-x[i]))])
      else:
        continue
    L =  L + y[j]*p
  xL = n_sample
  yL = L(xL)
  return(yL)


#Defining the given function
def f(x):
  z = 1 / (25*pow(x,2)+1)
  return(z)

#Calculation of 10 points
x1 = np.linspace(-1,1,10)
y1 = f(x1)

#Calculation of 5 points and its lagrange polynomial
x2 = np.linspace(-1,1,5)
y2 = f(x2)

#Calculation of 3 points and its lagrange polynomial
x3 = np.linspace(-1,1,3)
y3 = f(x3)
```

```
#Creating n_sample array containing values from -1 to 1 to plot graphs
n_sample = np.linspace(-1,1,1000)

# To Plot 4 figures for each case
fig, axs = plt.subplots(2, 2,figsize=(25,15))

axs[0, 0].plot(n_sample, f(n_sample),label = r'$\frac{1}{25x^{2}+1}$',)
axs[0, 0].set_title('Original Function')
axs[0, 0].legend(loc = 'upper right')
axs[0, 0].grid()

axs[0, 1].plot(n_sample, f(n_sample),label = 'Original Function')
axs[0, 1].scatter(x1, y1, label = '10 points',color='black')
axs[0, 1].plot(n_sample, lag(x1,y1,n_sample), label = 'Lagrangian Polynomial')
axs[0, 1].set_title('10 Points Interpolation')
axs[0, 1].legend(loc = 'upper right')
axs[0, 1].grid()

axs[1, 0].plot(n_sample, f(n_sample),label = 'Original Function')
axs[1, 0].scatter(x2, y2, label = '5 points',color='black')
axs[1, 0].plot(n_sample, lag(x2,y2,n_sample), label = 'Lagrangian Polynomial')
axs[1, 0].set_title('5 Points Interpolation')
axs[1, 0].legend(loc = 'upper right')
axs[1, 0].grid()

axs[1, 1].plot(n_sample, f(n_sample),label = 'Original Function')
axs[1, 1].scatter(x3, y3, label = '3 points',color='black')
axs[1, 1].plot(n_sample, lag(x3,y3,n_sample), label = 'Lagrangian Polynomial')
axs[1, 1].set_title('3 Points Interpolation')
axs[1, 1].legend(loc = 'upper right')
axs[1, 1].grid()

for ax in axs.flat:
  ax.set(xlabel='x', ylabel='y')

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

Output:



Explanation:

- Figure on Top Left: It shows the graph of the original function.
- Figure on Top Right: Interpolation with 10 points. Initially 10 points from -1 to +1 are taken to construct the Lagrangian Polynomial. The 10 points are shown as black dots, the original function is given in the blue line and the Lagrangian function is given in the orange line. When 10 points are taken for interpolation, the Lagrangian polynomial passes through the 10 points. Compared to 5 and 3 points interpolation, 10 points interpolation better approximates the given function.
- Figure on Bottom Left: Interpolation with 5 points.
- Figure on Bottom Right: Interpolation with 3 points.
- As the number of datapoints increases, the interpolated function better approximates the given function, but the oscillation starts to increase.