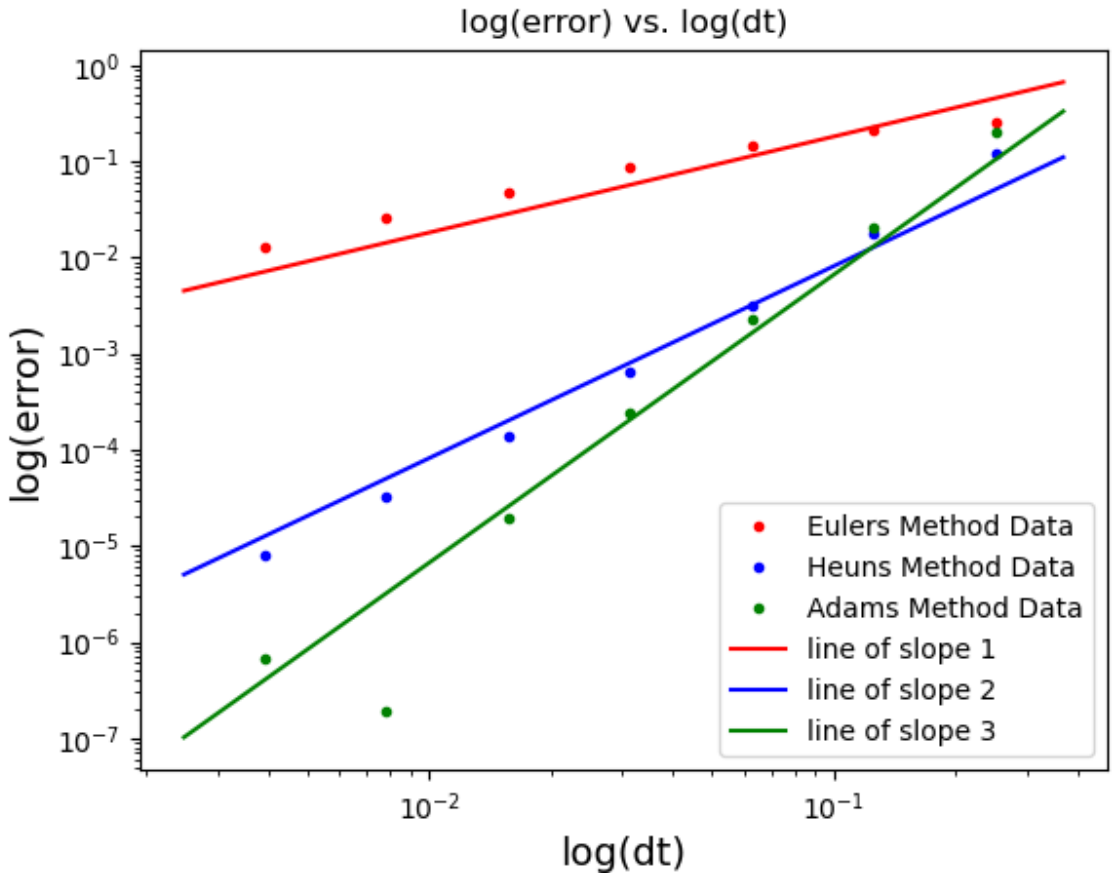

Figure 1: The data (red, blue, and green circles) from Euler's, Heun's, and Adam's methods respectively plotted on a log(error) vs. log(dt) plot using python. Lines of slope 1, 2, and 3 (red, blue, and green solid lines) are also plotted to show the order of each method respectively.

Appendix

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

def dydt(t, y):
    return (-3) * y * np.sin(t)

def ytrue(t):
    return (np.pi * np.exp(3 * (np.cos(t) - 1)) / np.sqrt(2))

t0 = 0
y0 = np.pi / np.sqrt(2)
dt = np.array([2**(-2), 2**(-3), 2**(-4), 2**(-5), 2**(-6),
2**(-7), 2**(-8)])
ans = ytrue(5)

def forward_euler(t0, y0, dt, dydt, ans):
    errorlist = np.empty(len(dt))

    for j in range(len(dt)):
        dtvals = np.arange(0, 5 + dt[j], dt[j])

        for i in range(len(dtvals)):
            if i == 0:
                y = y0
                yvals = np.array(y0)
            else:
                y = y + dt[j] * dydt(dtvals[i-1], y)
                yvals = np.append(yvals, y)

        errorlist[j] = np.abs(ans - y)

    return yvals, errorlist

solfe = forward_euler(t0, y0, dt, dydt, ans)

def heun(t0, y0, dt, dydt, ans):
    errorlist = np.empty(len(dt))

    for j in range(len(dt)):
        dtvals = np.arange(0, 5 + dt[j], dt[j])

        for i in range(len(dtvals)):
            if i == 0:
```

```
            y = y0
            yvals = np.array(y0)
         else:
            y = y + (dt[j] / 2) * (dydt(dtvals[i-1], y) +
dydt(dtvals[i-1] + dt[j], y + dt[j] * dydt(dtvals[i-1], y)))
            yvals = np.append(yvals, y)

      errorlist[j] = np.abs(ans - y)

   return yvals, errorlist

solheun = heun(t0, y0, dt, dydt, ans)

def RK2(t0, y0, dtval, dydt):
   return y0 + dtval * dydt(t0 + (dtval/2), y0 + (dtval/2) *
dydt(t0, y0))

def adams(t0, y0, dt, dydt, ans):
   errorlist = np.empty(len(dt))

   for j in range(len(dt)):
      dtvals = np.arange(0, 5 + dt[j], dt[j])

      for i in range(len(dtvals)):
         if i == 0:
            y = y0
            yvals = np.array([y0])
         elif i == 1:
            yprev = y0
            y = RK2(t0, y0, dt[j], dydt)
            yvals = np.append(yvals, y)
         else:
            ypred = y + (dt[j]/2) * (3 * dydt(dtvals[i-1], y) -
dydt(dtvals[i-2], yprev))
            yprev = y
            y = y + (dt[j]/2) * (dydt(dtvals[i-1] + dt[j],
ypred) + dydt(dtvals[i-1], y))
            yvals = np.append(yvals, y)

      errorlist[j] = np.abs(ans - y)

   return yvals, errorlist

soladams = adams(t0, y0, dt, dydt, ans)
```

```
plt.loglog(dt, solfe[1], '.r', label='Eulers Method Data')
plt.loglog(dt, solheun[1], '.b', label='Heuns Method Data')
plt.loglog(dt, soladams[1], '.g', label='Adams Method Data')
xaxis = np.linspace(-6, -1, 1000)
plt.loglog(np.exp(xaxis), np.exp(xaxis + 0.6), '-r', label='line
of slope 1')
plt.loglog(np.exp(xaxis), np.exp(2 * xaxis - 0.2), '-b',
label='line of slope 2')
plt.loglog(np.exp(xaxis), np.exp(3 * xaxis + 1.9), '-g',
label='line of slope 3')
plt.xlabel('log(dt)', fontsize=14)
plt.ylabel('log(error)', fontsize=14)
plt.title('log(error) vs. log(dt)')
plt.legend(loc='lower right')
plt.show()
```