j)
Utilizei python para resolução de todas as questões.
Segue o código: [Teste 03 - MS211 - 187793 - Pastebin.com]

Bibliotecas:

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

Variáveis e funções gerais:

```
Y0 = np.array([[1], [1]])
```

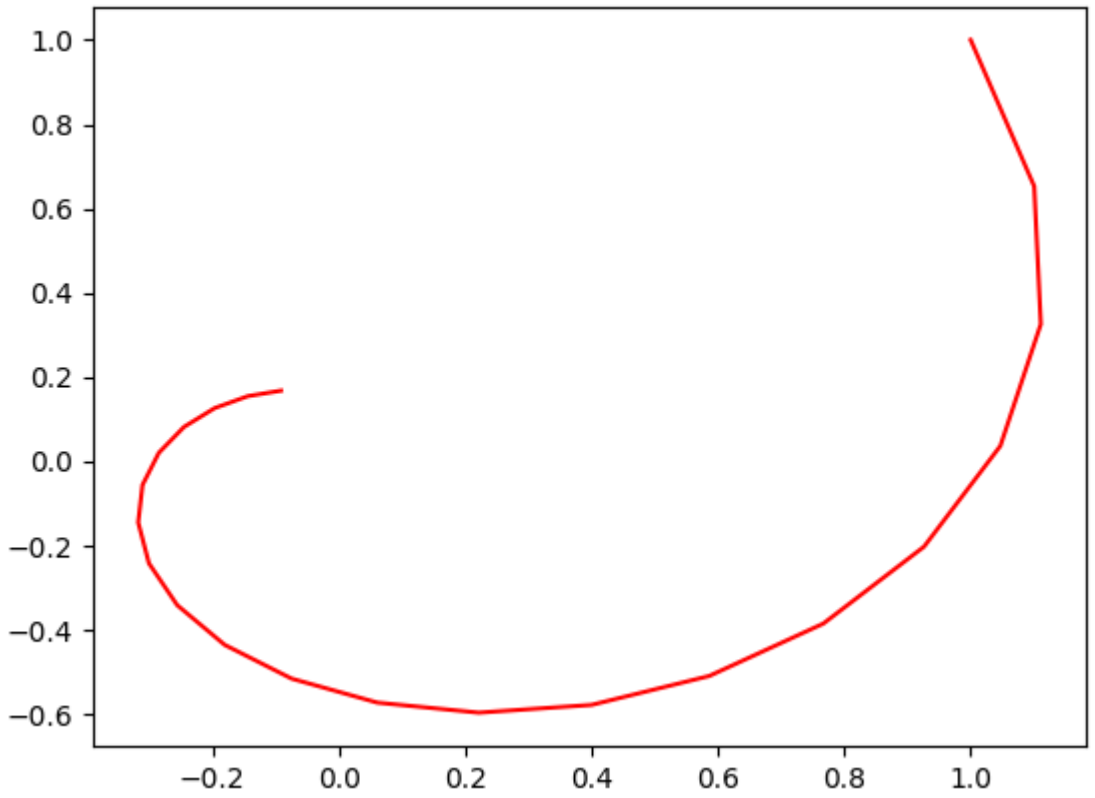
```
def Y_(arr):
    x = float(arr[0])
    y = float(arr[1])
    return np.array([[-x + 2.5 * y], [-2.5 * x - y]])
```

```
def Y__(arr):
    x = float(arr[0])
    y = float(arr[1])
    return np.array([[-5.25 * x - 5 * y], [5 * x - 5.25 * y]])
```

Curva vermelha:

```
def Y(t):
    return np.array([[np.exp(-t) * (np.sin(2.5 * t) + np.cos(2.5 * t))], [np.exp(-t) * (np.cos(2.5 * t) - np.sin(2.5 * t))]])

h = 0.1
tk = np.linspace(0, 20 * h, 21)
Y = np.array([Y(t) for t in tk])

x = Y[:,0]
y = Y[:,1]

plt.plot(x, y, color = 'red')
```

Output:



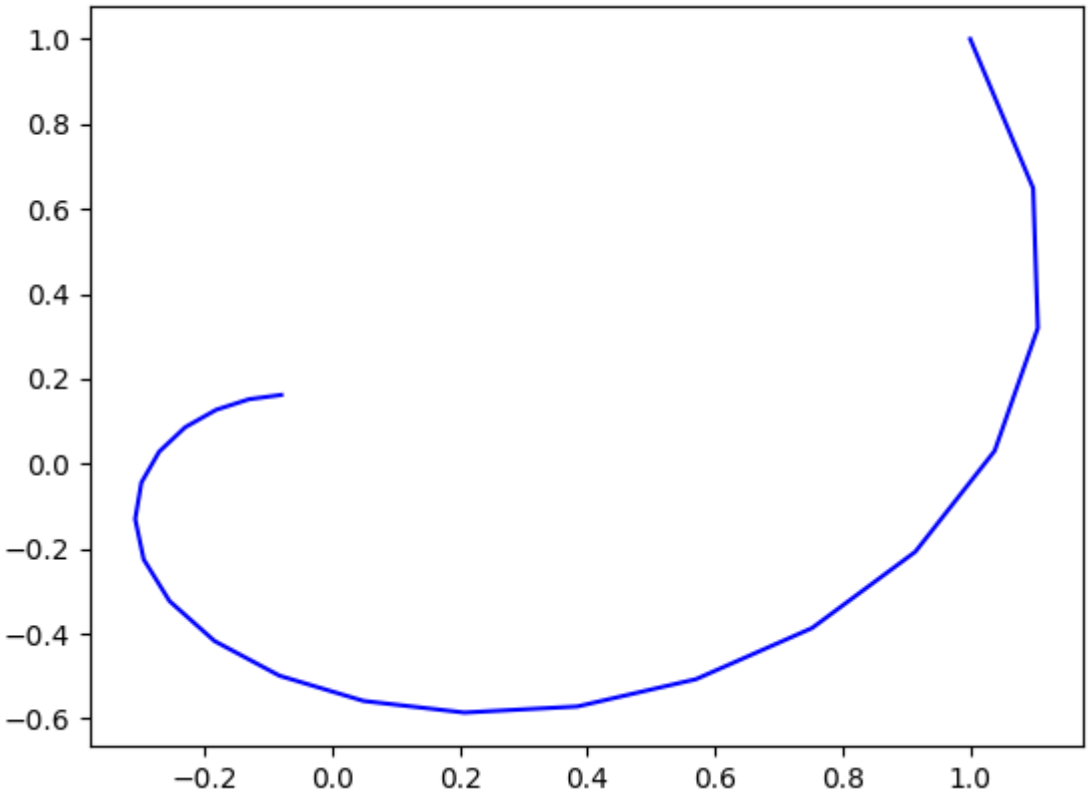

Curva azul:

```
Y = np.array(get_heun_table(21, 0.1, Y0, Y_))[:21,1]
x = []
y = []
for aux in Y:
    x.append(aux[0])
    y.append(aux[1])

plt.plot(x, y, color = 'blue')
```

```
def get_heun_table(it, h, Y0, F):
    Y = [Y0]

    table = []
    for k in range(it):
        Ykp1, k1, k2 = heun_step(Y[k], Y_, h)
        table.append([k * h, np.round(Y[k], 4), np.round(k1, 4), np.round(Y[k] + h * k1, 4), np.round(k2, 4), np.round(h * (k1 + k2) / 2, 4)])

        Y.append(Ykp1)

    table.append([it * h, np.round(Y[-1], 4), None, None, None, None])
    return table
```

```
def heun_step(Yk, F, h):
    k1 = F(Yk)
    k2 = F(Yk + h * k1)
    return Yk + h * (k1 + k2) / 2, k1, k2
```

Output:




Sobrepostas: