Classe Edo

```python
import numpy as np
import pandas as pd
import plotly.graph_objects as go
import plotly.express as px

class Edo:

  def __init__(self, y_inicial, dom, a,b, f_function,\
               analytic_function=None, all_error=[], output=pd.DataFrame([]), \
              analytic_solution=False, bidimensional=False):
    self.y = y_inicial
    self.dom = dom
    self.a = a
    self.b = b
    self.f_function = f_function
    self.analytic_solution = analytic_solution
    self.analytic_function = analytic_function
    self.bidimensional = bidimensional

    self.h = (self.b - self.a)/self.dom
    self.size = self.dom-1
    self.all_x = np.arange(self.a, self.b, self.h)
    self.all_y = []
    self.all_y_euller =[self.y]
    self.all_y_analitica = []
    self.yh = [self.y]
    self.y_pm = [self.y]
    self.y_newton = [self.y]

    self.all_y_euller_2d = None
    self.all_y_rk4_2d = None
    self.all_error = all_error
    self.output = output


  def f(self, x, y):
    return self.f_function(x, y)

  def analitica(self, x, **kwargs):
    if 'y' in kwargs:
      y = list(kwargs.values())[1]
      return self.analytic_function(x,y)
    else:
      return self.analytic_function(x)

  def df(self, x, y):
    return -1.2

  def report_euller(self):
    for i in range(len(self.all_x)):
      if len(self.all_y_euller)<=self.size:
        self.all_y_euller.append(self.all_y_euller[i]+self.h*self.f(self.all_x[i],self.all_y_euller[i]))

    if self.analytic_solution:
      self.all_y_analitica = list(map(self.analitica, self.all_x))
      self.all_error = abs(np.array(self.all_y_euller)-np.array(self.all_y_analitica))


  def report_euller_melhorado(self):
    for i in np.arange(0, len(self.all_x)):
      if len(self.yh)<=self.size:
          self.k1 = self.h*self.f(self.all_x[i],self.all_y_euller[i])
          self.k2 = self.h*self.f(self.all_x[i+1],self.all_y_euller[i]+self.k1)
          self.yh.append(self.yh[i] + ((self.k1+self.k2)/2))

  def report_ponto_medio(self):
    for i in np.arange(0, len(self.all_x)):
      if len(self.y_pm)<=self.size:
          self.k1_pm = self.f(self.all_x[i],self.all_y_euller[i])
          self.k2_pm = self.f(self.all_x[i]+(self.h)/2,self.all_y_euller[i]+self.k1_pm*(self.h/2))
          self.y_pm.append(self.y_pm[i] + self.k2_pm*self.h)

  def report_newton(self):
    for i in np.arange(0, len(self.all_x)):
      if len(self.y_newton)<=self.size:
          z = self.y_newton[i]+self.f(self.all_x[i], self.y_newton[i])*self.h
          count = 0
          while count <= 5:
            F = self.y_newton[i]+self.h*self.f(self.all_x[i+1], z) - z
            dF = self.h*self.df(self.all_x[i+1], z) - 1
            z = z - (F/dF)
            count+=1
          self.y_newton.append(z)

  def report_euller_2d(self):
    self.all_y_euller_2d = np.zeros((len(self.all_x), len(self.y)))
    k = np.zeros((len(self.all_x), len(self.y)))
    self.all_y_euller_2d[0] = self.y

    for i in range(len(self.all_x)-1):
      k[i] = self.f(self.all_x[i], self.all_y_euller_2d[i])
      self.all_y_euller_2d[i+1] = self.h*k[i]+self.all_y_euller_2d[i]


  def report_rk4_2d(self):
    self.all_y_rk4_2d = np.zeros((len(self.all_x), len(self.y)))
    self.all_y_rk4_2d[0] = self.y
    for i in range(len(self.all_x)-1):
      k1 = self.h*np.array((self.f(self.all_x[i], self.all_y_rk4_2d[i])))
      k2 = self.h*np.array(self.f(self.all_x[i]+self.h/2, self.all_y_rk4_2d[i]+k1/2))
      k3 = self.h*np.array(self.f(self.all_x[i]+self.h/2, self.all_y_rk4_2d[i]+k2/2))
      k4 = self.h*np.array(self.f(self.all_x[i]+self.h, self.all_y_rk4_2d[i]+k3))
      self.all_y_rk4_2d[i+1] = self.all_y_rk4_2d[i] + (k1 + 2*k2 +2*k3 + k4)/6


  @property
  def df_output(self):
    if self.bidimensional:
      self.report_euller_2d()
      self.report_rk4_2d()
      self.output = pd.DataFrame({'Passo':self.all_x, 'Y_Euller': self.all_y_euller_2d[:,0], \
                                  'P_Euller': self.all_y_euller_2d[:,1],'Y_RK4_2D':self.all_y_rk4_2d[:,0], \
                                  'P_RK4_2D': self.all_y_rk4_2d[:,1]})

      return self.output
    else:
      self.report_euller()
      self.report_euller_melhorado()
      self.report_ponto_medio()
      self.report_newton()

      if self.analytic_solution:
        self.output = pd.DataFrame({'Passo':self.all_x, 'Euller':self.all_y_euller, \
                        'Analítica':self.all_y_analitica, 'Erro':self.all_error,\
                        'Euller Melhorado':self.yh, 'Ponto Médio':self.y_pm,\
                        'Newton':self.y_newton})
      else:
        self.output = pd.DataFrame({'Passo':self.all_x, 'Euller':self.all_y_euller, \
                        'Euller Melhorado':self.yh, 'Ponto Médio':self.y_pm,\
                        'Newton':self.y_newton})
      return self.output

  def plot_output(self):
    fig = go.Figure()
    if self.bidimensional:
      fig.add_trace(go.Scatter(x=self.output['Passo'], y=self.output['Y_Euller'],
                        mode='lines',
                        name='Y(t) Euller'))
      fig.add_trace(go.Scatter(x=self.output['Passo'], y=self.output['Y_RK4_2D'],
                  mode='lines',
                  name='Y(t) Runge-Kutta 4ª Ordem'))
      fig.add_trace(go.Scatter(x=self.output['Passo'], y=self.output['P_Euller'],
                        mode='lines',
                        name='P(t) Euller',
                        line = {'color': '#341f97',
                            'dash': 'dash'}))
      fig.add_trace(go.Scatter(x=self.output['Passo'], y=self.output['P_RK4_2D'],
                        mode='lines',
                        name='P(t) Runge-Kutta 4ª Ordem',
                        line = {'color': '#FF914D',
                            'dash': 'dash'}))

      fig.update_layout(title_text='Resultados por Método', title_x=0.5,\
                        xaxis_title='t', yaxis_title='y(t), p(t)',\
                        height = 400, width = 800, font={'size':10})
      fig.show()
    else:
      fig.add_trace(go.Scatter(x=self.output['Passo'], y=self.output['Euller'],
                          mode='lines',
                          name='Euller'))
      fig.add_trace(go.Scatter(x=self.output['Passo'], y=self.output['Euller Melhorado'],
                            mode='lines',
                            name='Euller Melhorado'))
      fig.add_trace(go.Scatter(x=self.output['Passo'], y=self.output['Ponto Médio'],
                            mode='lines',
                            name='Ponto Médio'))
      if 'Newton' in self.output.columns:
        fig.add_trace(go.Scatter(x=self.output['Passo'], y=self.output['Newton'],
                              mode='lines',
                              name='Newton'))
      if self.analytic_solution:
        fig.add_trace(go.Scatter(x=self.output['Passo'], y=self.output['Analítica'],
                            mode='lines',
                            name='Analítica'))

      fig.update_layout(title_text='Resultados por Método', title_x=0.5,\
                        xaxis_title='Passo', yaxis_title='Método',\
                        height = 400, width = 800, font={'size':10})
      fig.show()

  def plot_error_distribution(self):
    if self.analytic_solution:
      fig = px.histogram(self.output, x="Passo", y="Erro", nbins=self.dom,\
                        color_discrete_sequence=['indianred'])
      fig.update_layout(title_text='Distribuição de Erro', title_x=0.5,\
                        xaxis_title='Passo', yaxis_title='Erro',\
                        height = 400, width = 600, font={'size':10})
      fig.show()
```

Atividade 19.04

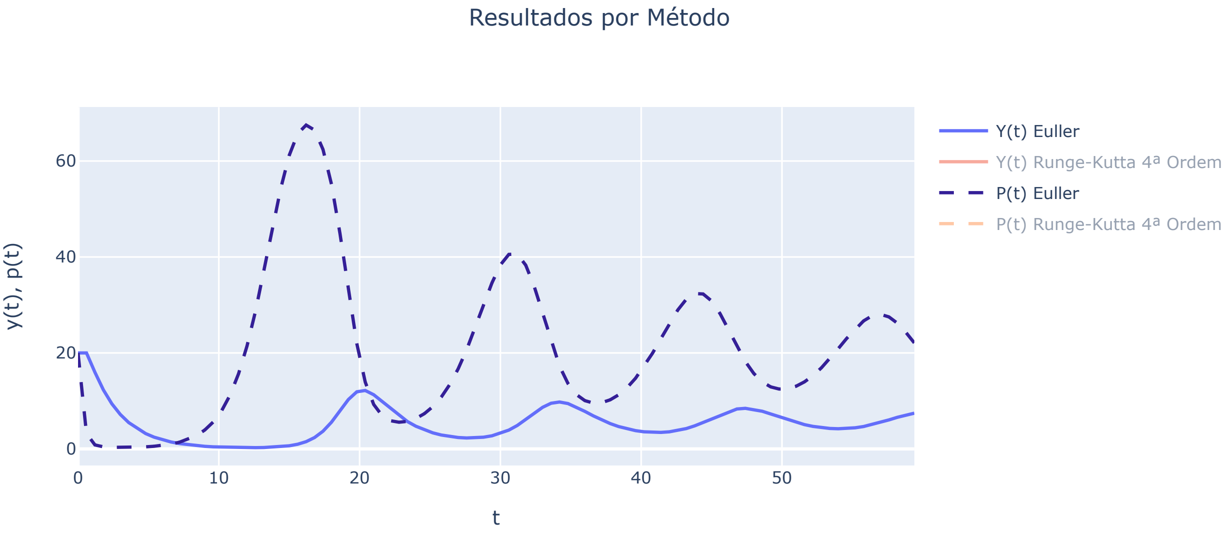
```
def S1(x,y):
  Y = y[0]
  P = y[1]
  dYdt = -0.4*Y + 0.02*P*Y
  dPdt = 0.8*P - 0.01*P*P-0.1*P*Y

  return dYdt, dPdt
```

```
e_1 = Edo(y_inicial=(20,20), dom=100, a=0, b=60, \
                    f_function=S1, analytic_solution=False, bidimensional=True)
```

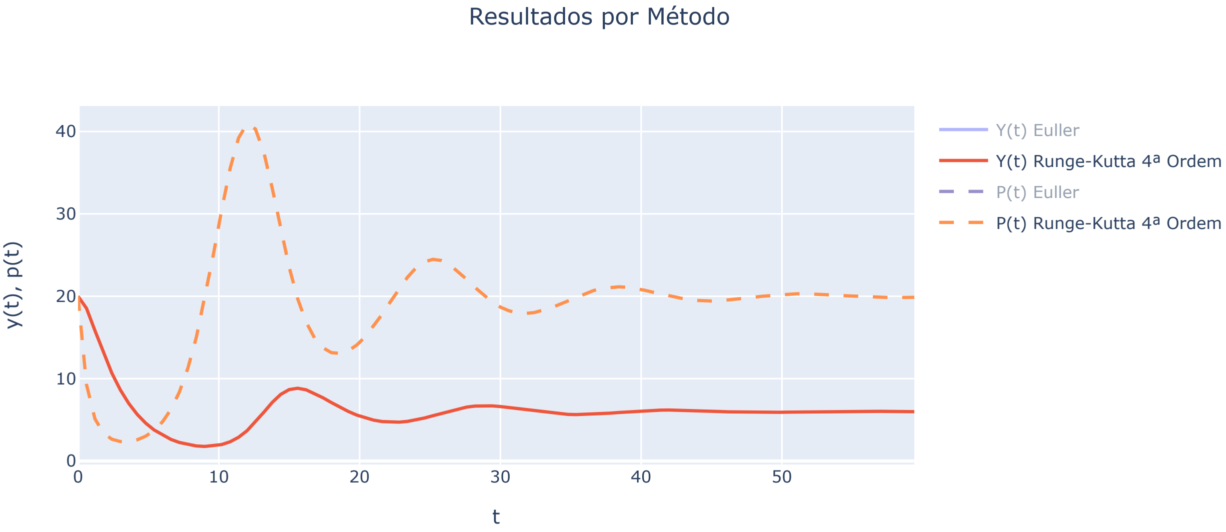
```
e_1.df_output.tail()
```

	Passo	Y_Euller	P_Euller	Y_RK4_2D	P_RK4_2D
95	57.0	5.513288	28.045146	6.033613	19.862383
96	57.6	6.045550	27.510377	6.022980	19.846070
97	58.2	6.590403	26.195511	6.011708	19.843992
98	58.8	7.080374	24.293789	6.000780	19.854692
99	59.4	7.445193	22.093132	5.991032	19.875870

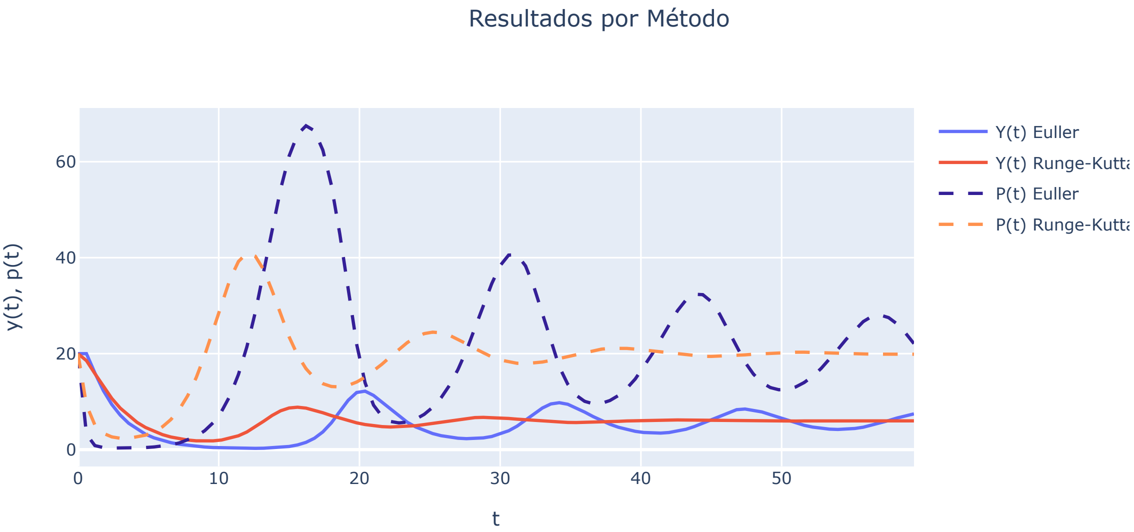
```
  e_1.plot_output()
```



```
e_1.plot_output()
```



```
e_1.plot_output()
```



Produtos pagos do Colab - Cancelar contratos

✓ 0s conclusão: 22:16