Exercicio computacional 1

Alexandre Rosseto Lemos

Considerações

- Indivíduo: parâmetros *a, b* e *c*
- Limites dos parâmetros: [-5, 5]
- Polinômio: y = a + bx + cx^2
- fitness = soma do erro médio quadrático (SSE)
- 10 execuções
- Será utilizada a biblioteca scikit-opt (https://pythonrepo.com/repo/guofei9987-scikit-opt-python-sklearn-utilities)

Bibliotecas

In [1]:
```
# Instalando a biblioteca scikit-opt
!pip install scikit-opt
```

```
Requirement already satisfied: scikit-opt in c:\users\xande\anaconda3\lib\site-packages (0.6.5)
Requirement already satisfied: scipy in c:\users\xande\anaconda3\lib\site-packages (from scikit-opt) (1.5.3)
Requirement already satisfied: numpy in c:\users\xande\anaconda3\lib\site-packages (from scikit-opt) (1.21.2)
```

In [2]:
```
import pandas as pd
import numpy as np
from sko.GA import GA
import matplotlib.pyplot as plt
```

Funções

In [3]:
```
# Funcao polinomial
def f_fun(x, a, b, c):

    # Polinomio definido
    y_hat = a + b*x + c*(x**2)

    return y_hat


# Funcao objetivo
def obj_fun(p):

    # Definindo os parametros
    a, b, c = p

    # Quantidade de amostras
    n = len(df_x)

    # Calcula o sse
    sse = (1/n)*(np.square(f_fun(df_x.values, a, b, c) - df_y.values).sum())

    return sse
```

Carregando os dados

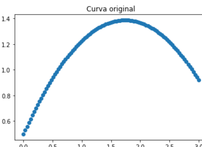
In [4]:
```
df_x = pd.read_csv('x_data.txt', names = ['X'])
df_y = pd.read_csv('y_data.txt', names = ['Y'])

# Previa dos dados
display(df_x.head(), df_y.head())
```

	X
0	0.00
1	0.03
2	0.06
3	0.09
4	0.12

	Y
0	0.4981
1	0.5285
2	0.5584
3	0.5878
4	0.6166

In [5]:
```
# Plot da curva original (conforme os dados nos arquivos txt)
plt.plot(df_x.values, df_y.values, 'o');
plt.title('Curva original');
```



Algoritmo genético

Análise do tamanho da populacao

In [17]:
```
size_list = [100, 200, 300, 400, 500, 1000, 2000, 5000]

list_results = []
for size_p in size_list:
    ga = GA(func = obj_fun, n_dim = 3, size_pop = size_p,
            lb = [-5, -5, -5], ub = [5, 5, 5])

    # Resultados obtidos
    best_params, best_sse = ga.run(10)

    # Salvando o resultado
    list_results.append((size_p,best_sse[0]))
```

In [18]:
```
df_pop_size_res = pd.DataFrame(list_results, columns = ['Pop', 'SSE']).set_index('Pop')
df_pop_size_res
```

Out[18]:

Pop	SSE
100	0.024440
200	0.012893
300	0.072998
400	0.031950
500	0.016861
1000	0.001285
2000	0.003538
5000	0.001914

O melhor resultado (menor SSE) foi obtido quando size_pop = 1000

Obtendo os resultados

In [22]:
```
# Executando o algoritmo 10x e salvando os resultados
results = []
for i in range(10):
    ga = GA(func = obj_fun, n_dim = 3, size_pop = 300,
            lb = [-5, -5, -5], ub = [5, 5, 5])

    # Resultados obtidos
    best_params, best_sse = ga.run(1)

    # Salvando o resultado
    results.append((best_sse[0], best_params))

# Obtendo a média dos resultados
sse_all = [sse for sse,par in results]
sse_med = sum(sse_all)/len(sse_all)

# Salvando os resultados em um dataframe
df_res = pd.DataFrame(results, columns = ['SSE', 'Parametros'])

# Obtendo os melhores parâmetros
best = df_res[df_res['SSE'] == df_res['SSE'].min()]

# Melhor SSE
best_sse_all = best['SSE'].values[0]

# Parametros
a = best['Parametros'].values[0][0]
b = best['Parametros'].values[0][1]
c = best['Parametros'].values[0][2]

print('Resultados Obtidos')
display(df_res)

print('----------')
print(f'Média SSE: {sse_med}')
print(f'Melhor SSE: {best_sse_all}\n\nMelhores parametros:')
print(f'a = {a}\nb = {b}\nc = {c}')
```

```
Resultados Obtidos
```

	SSE	Parametros
0	0.259811	[-0.9377904678716549, 3.2242027573783782, -0.96...
1	0.852813	[-1.1707700503674898, 2.8671057363839, -0.577...
2	0.023806	[0.5647768494842706, 0.8382980960480726, -0.19...
3	1.305149	[-1.2120099083483957, 2.9354215483026316, -0.9...
4	0.860542	[0.8058975324474096, -1.3087981664229983, 0.519...
5	0.729270	[-1.7784633992497878, 3.377180459925386, -0.90...
6	0.089484	[0.3443278025413141, 0.7622880636475090, -0.10...
7	0.394949	[0.7044015504747749, 2.1125271710125597, -0.832...
8	0.035778	[0.8485179830232115, 0.2259257587663523, -0.0...
9	0.131389	[-0.03395397241379339, 1.0572202209921198, -0....

```
----------
Média SSE: 0.4682991268097384
Melhor SSE: 0.023805645855245064

Melhores parametros:
a = 0.5647768494842706
b = 0.8382980960480726
c = -0.19360948498256114
```

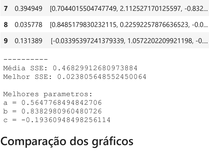
Comparação dos gráficos

In [24]:
```
best_params = [a,b,c]

# Gerando gráfico
y_predict = f_fun(df_x.values, *best_params)

fig, ax = plt.subplots()

ax.plot(df_x.values, df_y.values, '-')
ax.plot(df_x.values, y_predict, '-')
ax.legend(['Curva Original', 'Curva com AG'])
plt.show()
```



In [ ]:
```

```