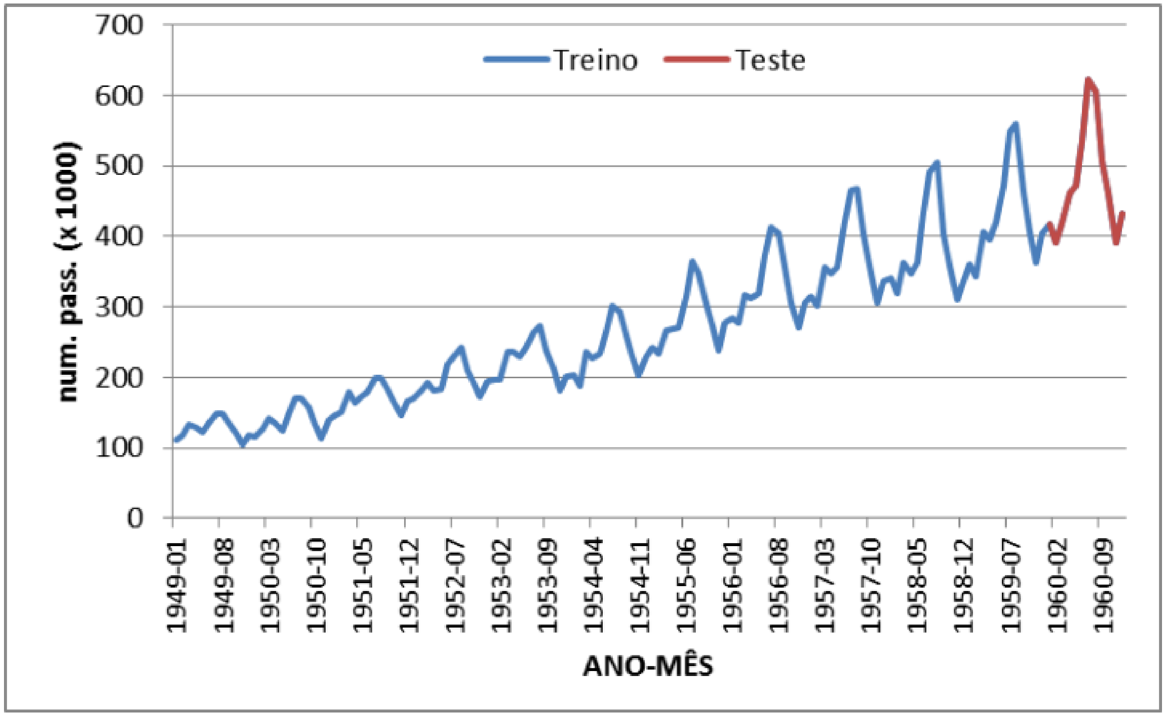
Previsão de Séries Temporais com Programação Genética

Este exercício tem o objetivo de utilizar Programação Genética para gerar um modelo de previsão de séries temporais. Os dados utilizados são referente ao número mensal de passageirosde linhas aéreas internacionais (em milhares/mês), desde janeir/1949 a dezembro/1960, conforme figura abaixo.



O primeiro passo foi criar o dataset com os dados da planilha fornecida. Separou-se em dois datasets: treinamento e teste.
Como **janela de periodicidade**, observou-se que a cada 12 passo o padrão se repetia, o que corresponde a um ano **(12 meses)**. Sendo assim utilizou-se esse valor para gerar uma matriz que contivesse 12 entradas e uma saída.

In [1]:
```
import numpy as np
from numpy import genfromtxt

#lendo do arquivo
data = genfromtxt('serie.txt')
data = np.array([[data[index+pos] for pos in range(13)] for index in range(data.size - 12)])

#separando os conjuntos
DATA_TRAIN_SIZE = data[:,1].size - 12
data_train = data[:DATA_TRAIN_SIZE,:]
data_test = data[DATA_TRAIN_SIZE:,:]
```

Para a implementação do algoritmo de Programação Genética, utilizou-se o **DEAP** *(Distributed Evolutionary Algorithms in Python)*. Abaixo realiza-se o import dos modulos que serão utilizados.

In [2]:
```
import operator
import math
import random

from deap import algorithms
from deap import base
from deap import creator
from deap import tools
from deap import gp
```

Define-se agora a estrutura da árvore que será gerada nas evoluções da PG. Esta árvore terá 12 entradas que correspnde a janela definida de 12 meses. Também serão definidas as funções e os terminais utilizados.

In [3]:
```
#definicao da estrutura de um conjunto que terá 12 entradas
pset = gp.PrimitiveSet("MAIN", 12)

pset.addPrimitive(operator.add, 2)
pset.addPrimitive(operator.sub, 2)
pset.addPrimitive(operator.mul, 2)
pset.addPrimitive(operator.neg, 1)
pset.addPrimitive(math.cos, 1)
pset.addPrimitive(math.sin, 1)
```

In [4]:
```
pset.addEphemeralConstant("rand101", lambda: random.randint(-180,180)) #terminal que gera um numero aleatorio entre [-1,1]
```

Os argumentos(entradas) serão renomeados para facilitar o entendimento da arvore gerada posteriormente. Cada argumento representa um mês do ano.

In [5]:
```
pset.renameArguments(ARG0='x1')
pset.renameArguments(ARG1='x2')
pset.renameArguments(ARG2='x3')
pset.renameArguments(ARG3='x4')
pset.renameArguments(ARG4='x5')
pset.renameArguments(ARG5='x6')
pset.renameArguments(ARG6='x7')
pset.renameArguments(ARG7='x8')
pset.renameArguments(ARG8='x9')
pset.renameArguments(ARG9='x10')
pset.renameArguments(ARG10='x11')
pset.renameArguments(ARG11='x12')
```

Na sequencia, será definido o alicerce da PG:

- característica do problema(minimização ou maximização)
- método de geração da população(grow, full, half and half)
- função de fitness
- função de crossover
- função de mutação
- função de seleção
- definição da profundidade máxima da árvore

In [6]:
```
#característica do problema
creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
#tipo do indivíduo = árvore
creator.create("Individual", gp.PrimitiveTree, fitness=creator.FitnessMin)

toolbox = base.Toolbox()
#método de geração da população(árvore)
toolbox.register("expr", gp.genHalfAndHalf, pset=pset, min_=1, max_=2)
toolbox.register("individual", tools.initIterate, creator.Individual, toolbox.expr)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
#compila a arvore gerada em uma função
toolbox.register("compile", gp.compile, pset=pset)

#função de fitness que compara a arvore gerada com os dados de treinamento
def fitness(individual):
    # Tranforma a expressao da arvore em um função invocável
    func = toolbox.compile(individual)

    # calcula o erro quadrado médio entre a expressao e o valor real 'y'
    sqerrors = ((func(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12)-y)**2 for x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,y in data_train)

    return math.fsum(sqerrors) / len(data_train),

#define para o framework qual será a função de fitness utilizada
toolbox.register("evaluate", fitness)
#define o operador de seleção como torneio de tamanho 3
toolbox.register("select", tools.selTournament, tournsize=3)
#define função de cruzamento como corte em um ponto
toolbox.register("mate", gp.cxOnePoint)
#cria função de mutação utilizando o método de geração de árvore full
toolbox.register("expr_mut", gp.genFull, min_=0, max_=2)
#define a função de mutação para o framework
toolbox.register("mutate", gp.mutUniform, expr=toolbox.expr_mut, pset=pset)

#quando houver cruzamento o tamanho da arvore nao pode passar de 50
toolbox.decorate("mate", gp.staticLimit(key=operator.attrgetter("height"), max_value=50))
#quando houver muatção o tamanho da arvore nao pode passar de 50
toolbox.decorate("mutate", gp.staticLimit(key=operator.attrgetter("height"), max_value=50))
```

Com toda a estrutura definida agora é possível criar a função que de fato irá rodar algoritmo.

In [7]:
```
def algorithm():
    random.seed(318)
    #CXPB  - Probabilidade de crossover
    #MUTPB - Probabilidade de mutação
    #NGEN  - Numero de gerações
    CXPB, MUTPB, NGEN = 0.6, 0.1, 200

    pop = toolbox.population(n=300)
    hof = tools.HallOfFame(1)

    stats_fit = tools.Statistics(lambda ind: ind.fitness.values)
    stats_size = tools.Statistics(len)
    mstats = tools.MultiStatistics(fitness=stats_fit, size=stats_size)
    mstats.register("avg", np.mean)
    mstats.register("std", np.std)
    mstats.register("min", np.min)
    mstats.register("max", np.max)

    # Realiza a chamada do algortimo definido no capitulo 7 do KOZA.
    # Passa como parâmetros:
    #     - população
    #     - toolbox que é o objeto que representa o framework DEAP
    #     - parametros evolucionais (probabilidades de crossover e mutação e gerações)
    #     - objeto com os placeholders de estatística
    #     - halloffame - individuo(s) selecionado(s)
    #     - verbos que representa se será impresso nas saida a evolução do algoritmo
    pop, log = algorithms.eaSimple(pop, toolbox, CXPB, MUTPB, NGEN, stats=mstats,
                                   halloffame=hof, verbose=True)
    # print log
    return pop, log, hof
```

In [8]:
```
pop, log, hof = algorithm()
```

		fitness				size			
gen	nevals	avg	max	min	std	avg	max	min	std
0	300	1.65318e+13	4.95914e+15	1188.26	2.85838e+14	3.20667	7	2	1.34063
1	199	3.96242e+08	3.6103e+10	1064.13	3.01159e+09	3.24333	10	1	1.30797
2	180	1.15695e+08	9.04468e+09	1058.99	9.9515e+08	3.49333	8	1	1.36746
3	199	4.68512e+07	1.03565e+10	1049.75	6.25313e+08	3.84667	9	1	1.53074
4	189	1.16897e+08	9.60767e+09	578.25	1.00456e+09	4.11	9	1	1.82882
5	178	1.62549e+08	3.11986e+10	578.25	1.92961e+09	4.12	11	1	2.09418
6	179	1.11676e+08	1.00963e+10	578.25	9.67879e+08	4.00667	11	1	2.17255
7	196	2.05817e+08	3.94031e+10	578.25	2.40321e+09	3.85333	11	1	2.35623
8	182	15832.3	742821	578.25	61177.1	3.40333	13	1	2.6958
9	181	1.67272e+08	1.02181e+10	578.25	1.17857e+09	3.07667	13	1	2.64401
10	185	1.8051e+08	1.04962e+10	578.25	1.2709e+09	3.67333	16	1	2.73617
11	180	6.50896e+12	1.95265e+15	578.25	1.12549e+14	3.87333	16	1	2.48676
12	196	2.01198e+08	3.1708e+10	578.25	2.00327e+09	3.66333	17	1	2.02402
13	205	1.29798e+08	3.61293e+10	293.583	2.08621e+09	3.4	17	1	1.58535
14	208	1.97564e+13	4.42297e+15	578.25	2.68994e+14	3.10333	9	1	0.916145
15	190	1.2318e+08	8.43946e+09	578.25	9.48599e+08	3.2	9	1	0.979796
16	201	5.08611e+17	1.52583e+20	578.25	8.79471e+18	3.16667	9	1	0.975819
17	192	5.59521e+07	8.4859e+09	578.25	6.82645e+08	3.1	9	1	0.888819
18	194	2.31927e+11	6.95381e+13	578.25	4.00808e+12	3.09	9	1	0.939095
19	183	6.68441e+07	9.67101e+09	578.25	7.39366e+08	3.07	9	1	0.681982
20	197	8.74651e+11	2.62355e+14	578.25	1.51218e+13	3.11667	8	1	0.759203
21	209	2.82502e+07	8.04407e+09	578.25	4.6417e+08	3.11	9	1	0.835404
22	176	5.92236e+07	9.03088e+09	578.25	7.2269e+08	3.17	9	1	0.788099
23	187	5.21991e+08	1.56589e+11	578.25	9.02561e+09	3.11667	8	1	0.709264
24	178	2.41686e+11	7.24785e+13	578.25	4.17756e+12	3.11	8	1	0.77324
25	197	8.23463e+07	9.80545e+09	578.25	8.10109e+08	3.12333	9	1	0.796736
26	201	1.08013e+08	8.35549e+09	578.25	8.96014e+08	3.15333	9	1	1.00821
27	208	2.14754e+08	3.40964e+10	578.25	2.2038e+09	3.22333	9	1	0.95923
28	199	8.62232e+07	8.72623e+09	578.25	8.57677e+08	3.2	9	1	0.890693
29	197	5.62735e+07	8.65036e+09	578.25	6.48805e+08	3.14	9	1	0.848764
30	175	3.01283e+07	9.03088e+09	578.25	5.20527e+08	3.09333	8	1	0.551926
31	179	2.4758e+08	3.06985e+10	578.25	2.08703e+09	3.12	8	1	0.687217
32	189	1.42788e+08	8.91071e+09	578.25	1.07778e+09	3.13333	9	1	0.771722
33	189	2.84413e+07	8.01578e+09	578.25	4.62852e+08	3.12333	9	1	0.792542
34	216	9.72762e+07	1.03565e+10	578.25	8.77972e+08	3.09333	9	1	0.760234
35	179	1.37165e+08	1.28538e+10	578.25	1.10853e+09	3.05667	9	1	0.616541
36	198	5.50855e+07	8.15428e+09	578.25	6.63543e+08	3.11	8	1	0.681591
37	205	1.00311e+11	3.00567e+13	578.25	1.73242e+12	3.11	9	1	0.900685
38	196	6.06894e+07	7.82571e+09	578.25	6.45172e+08	3.02667	8	1	0.72981
39	203	5.13764e+08	1.35606e+11	578.25	7.84903e+09	3.11	8	1	0.760197
40	177	3.14542e+10	9.40778e+12	578.25	5.42248e+11	3.18667	10	1	0.940827
41	188	7.86124e+07	8.18853e+09	578.25	7.82307e+08	3.14333	11	1	0.869553
42	210	8.48926e+07	1.08301e+10	578.25	8.54351e+08	3.23	9	1	0.947154
43	193	29311.8	348858	578.25	46912.8	3.08	7	1	0.643117
44	200	2.86066e+07	7.34646e+09	578.25	4.29067e+08	3.09333	9	1	0.681632
45	190	4.76078e+08	1.34047e+11	578.25	7.74111e+09	3.09333	9	1	0.696148
46	190	1.25209e+13	3.75624e+15	578.25	2.16505e+14	3.08333	9	1	0.79774
47	174	4.91982e+17	1.47591e+20	578.25	8.50693e+18	3.15	9	1	0.770822
48	194	1.65873e+08	9.19754e+09	578.25	1.15112e+09	3.13667	8	1	0.671805
49	182	3.26414e+12	9.79165e+14	578.25	5.64378e+13	3.14667	9	1	0.742847
50	194	1.59989e+12	4.79908e+14	578.25	2.76613e+13	3.14667	9	1	0.859354

51	186	1.83437e+13	5.50307e+15	578.25	3.1719e+14	3.11	8	1	0.728858
52	202	1.34094e+08	1.07169e+10	578.25	1.05042e+09	3.07333	9	1	0.796632
53	183	2.36509e+08	3.77678e+10	578.25	2.89269e+09	3.07	9	1	0.807321
54	197	1.11893e+08	2.50815e+10	578.25	1.5245e+09	3.06667	9	1	0.694422
55	178	1.53374e+08	3.84627e+10	578.25	2.25775e+09	3.1	8	1	0.665833
56	176	1.52048e+08	1.10544e+10	578.25	1.18355e+09	3.18667	9	1	0.855466
57	195	1.0063e+08	9.80339e+09	578.25	9.04422e+08	3.24	9	1	1.00784
58	177	4.13515e+11	1.24037e+14	578.25	7.14934e+12	3.11333	9	1	0.766261
59	189	5.93221e+07	9.09239e+09	578.25	7.24009e+08	3.05333	6	1	0.526456
60	199	2.64839e+07	7.93809e+09	578.25	4.5754e+08	3.12333	9	1	0.753739
61	188	1.60129e+08	9.61866e+09	578.25	1.12847e+09	3.15667	8	1	0.769495
62	191	4.17226e+12	1.25165e+15	578.25	7.21436e+13	3.17667	9	1	0.958535
63	204	6.17145e+16	1.85103e+19	578.25	1.06691e+18	3.09667	9	1	0.841025
64	193	7.42808e+07	8.15233e+09	578.25	7.37551e+08	3.12333	9	1	0.792542
65	200	5.90263e+07	9.98733e+09	578.25	7.26164e+08	3.10667	8	1	0.75407
66	186	1.23414e+08	1.06507e+10	578.25	1.02922e+09	3.20333	8	1	0.956725
67	189	1.49415e+08	9.5603e+09	578.25	1.06597e+09	3.13	9	1	0.706942
68	191	4.85025e+08	1.26994e+11	578.25	7.35538e+09	3.22667	9	1	0.928416
69	204	7.90994e+07	8.59199e+09	578.25	7.76659e+08	3.12	8	1	0.790949
70	186	9.83476e+07	8.33778e+09	578.25	8.31786e+08	3.13333	9	1	0.784573
71	186	3.27027e+07	9.80339e+09	578.25	5.65053e+08	3.07333	7	1	0.560912
72	215	2.74524e+08	3.26014e+10	578.25	2.4402e+09	3.14667	9	1	0.863224
73	194	2.00464e+13	6.01386e+15	578.25	3.46631e+14	3.17	9	1	0.976951
74	190	2.58186e+11	7.74473e+13	578.25	4.46396e+12	3.12333	9	1	0.906342
75	190	3.37602e+12	1.01278e+15	578.25	5.83755e+13	3.13667	8	1	0.646521
76	203	3.37258e+12	1.01174e+15	578.25	5.83153e+13	3.20333	8	1	0.895538
77	192	7.31229e+07	7.46492e+09	578.25	7.17681e+08	3.13333	9	1	0.81377
78	175	1.44427e+08	9.12847e+09	578.25	1.11009e+09	3.03333	7	1	0.620931
79	206	2.64554e+07	7.63812e+09	578.25	4.40514e+08	3.07667	9	1	0.719344
80	177	3.37801e+12	1.01338e+15	321.583	5.84097e+13	3.12667	8	1	0.681143
81	190	6.58726e+15	1.97618e+18	578.25	1.13904e+17	3.13667	9	1	0.773297
82	182	2.31934e+07	6.93614e+09	578.25	3.99787e+08	3.02333	8	1	0.645075
83	188	2.95594e+12	8.86768e+14	578.25	5.11122e+13	3.10667	9	1	0.76286
84	207	1.95863e+11	5.86983e+13	578.25	3.38329e+12	3.17	9	1	0.864735
85	173	3.74857e+07	1.12396e+10	578.25	6.47835e+08	3.03333	7	1	0.488763
86	189	1.51565e+08	1.323e+10	578.25	1.19485e+09	3.14	9	1	0.81265
87	207	4.59647e+08	1.29735e+11	578.25	7.49093e+09	3.04333	9	1	0.664421
88	191	3.17594e+07	4.57073e+09	578.25	3.4338e+08	3.1	9	1	0.732575
89	206	6.56444e+12	1.96932e+15	578.25	1.13509e+14	3.12667	9	1	0.695669
90	190	5.53951e+17	1.66185e+20	578.25	9.57871e+18	3.10333	9	1	0.752322
91	201	4.55011e+06	1.35626e+09	578.25	7.81715e+07	3	7	1	0.439697
92	183	1.05778e+08	1.36022e+10	578.25	1.07232e+09	3.06333	8	1	0.734386
93	184	7.76557e+07	8.43061e+09	578.25	7.73909e+08	3.08667	8	1	0.711212
94	187	5.95165e+08	1.50406e+11	578.25	8.70353e+09	3.16	9	1	0.88
95	195	6.13999e+07	8.768e+09	578.25	6.89015e+08	3.15	8	1	0.894893
96	183	5.4907e+07	8.2098e+09	578.25	6.31368e+08	3.09333	6	1	0.641318
97	209	1.3199e+13	3.95968e+15	578.25	2.28231e+14	3.14667	9	1	0.897305
98	215	9.7617e+10	2.92579e+13	578.25	1.68638e+12	3.13667	9	1	0.885808
99	191	5.64962e+07	9.23988e+09	578.25	6.92187e+08	3.07667	8	1	0.671408
100	178	6.98236e+07	9.67101e+09	578.25	7.74629e+08	3.08667	8	1	0.667699
101	202	1.17676e+08	9.09446e+09	578.25	1.01243e+09	3.08	9	1	0.697806
102	183	1.12172e+08	8.88921e+09	578.25	9.53077e+08	3.15	7	1	0.707696
103	156	2.51765e+07	7.54648e+09	578.25	4.34968e+08	3.10333	9	1	0.738911
104	187	1.69932e+08	3.28956e+10	578.25	2.01133e+09	3.13333	9	1	0.817856

105	192	4.94474e+07	7.75576e+09	578.25	5.98693e+08	3.11	8	1	0.76457
106	173	2.19311e+08	9.98944e+09	578.25	1.33581e+09	3.05333	6	1	0.563166
107	200	5.59735e+07	8.49813e+09	578.25	6.82978e+08	3.10333	8	1	0.75674
108	207	7.40756e+17	2.22227e+20	578.25	1.28089e+19	3.19667	9	1	0.971934
109	201	1.43054e+13	4.29157e+15	578.25	2.47361e+14	3.24	9	1	0.892039
110	186	4.68551e+17	1.40562e+20	578.25	8.10179e+18	3.24333	9	1	1.00869
111	186	4.44566e+12	1.33365e+15	578.25	7.687e+13	3.15667	9	1	0.88249
112	213	2.47678e+07	7.42132e+09	578.25	4.27754e+08	3.10333	8	1	0.743408
113	187	1.24103e+08	1.08709e+10	578.25	1.06357e+09	3.10333	9	1	0.863706
114	163	3.48928e+12	1.04677e+15	578.25	6.03343e+13	3.11667	8	1	0.665624
115	188	4.1866e+07	7.12657e+09	578.25	5.15419e+08	3.13	8	1	0.621638
116	209	1.81239e+13	5.43711e+15	400.917	3.13388e+14	3.13333	9	1	0.865384
117	180	1.02615e+08	1.11868e+10	578.25	9.92654e+08	3.02333	8	1	0.519091
118	188	8.06586e+07	8.46598e+09	578.25	8.03348e+08	3.14333	8	1	0.750193
119	194	3.37707e+06	1.00526e+09	578.25	5.79403e+07	3.16	9	1	0.845222
120	187	8.40234e+07	8.57554e+09	578.25	8.35858e+08	3.1	9	1	0.862168
121	194	5.8028e+07	8.45549e+09	578.25	6.74395e+08	3.09333	9	1	0.696148
122	206	1.27952e+13	3.83853e+15	578.25	2.21248e+14	3.18667	9	1	1.05443
123	192	6.45648e+07	1.03586e+10	578.25	7.81061e+08	3.16	9	1	0.935094
124	194	1.35813e+08	1.7086e+10	578.25	1.2558e+09	3.12667	9	1	0.90772
125	177	4.18703e+12	1.2561e+15	578.25	7.23999e+13	3.02333	9	1	0.568145
126	190	1.90489e+08	2.90241e+10	578.25	1.86997e+09	3.16	9	1	0.864716
127	176	9.16279e+11	2.74875e+14	578.25	1.58435e+13	3.05667	8	1	0.632552
128	205	2.39233e+07	7.16969e+09	578.25	4.13251e+08	3.03	7	1	0.531445
129	201	2.45394e+11	7.36022e+13	578.25	4.24233e+12	3.17667	9	1	0.901178
130	194	5.90047e+17	1.77013e+20	578.25	1.02028e+19	3.11333	8	1	0.816796
131	206	2.13756e+08	8.77002e+09	578.25	1.29321e+09	3.2	9	1	0.948683
132	217	2.9662e+07	8.88921e+09	578.25	5.12361e+08	3.14	8	1	0.766203
133	177	5.43901e+07	8.2843e+09	578.25	6.63751e+08	3.12333	8	1	0.735837
134	191	3.11563e+07	8.81461e+09	578.25	5.08865e+08	3.14667	9	1	0.729261
135	210	1.27347e+08	1.08263e+10	578.25	1.10153e+09	3.14	9	1	0.766203
136	196	1.41685e+08	8.33181e+09	578.25	1.00443e+09	3.15667	9	1	0.926709
137	177	5.29907e+07	8.2453e+09	578.25	6.46996e+08	3.12	9	1	0.72037
138	180	1.51889e+07	4.54894e+09	578.25	2.62193e+08	3.12333	8	1	0.67438
139	204	1.36725e+06	2.08233e+08	578.25	1.64166e+07	3.10667	8	1	0.579041
140	201	9.18498e+07	8.15233e+09	578.25	8.12281e+08	3.1	8	1	0.772442
141	165	2.84613e+07	8.53196e+09	350.25	4.9177e+08	3.11333	9	1	0.820867
142	193	7.47383e+10	2.23921e+13	350.25	1.29065e+12	3.08333	9	1	0.699802
143	207	1.14317e+08	9.60762e+09	350.25	9.85968e+08	3.05667	9	1	0.692909
144	179	1.29599e+18	2.51186e+20	350.25	1.64851e+19	3.07667	9	1	0.619776
145	202	2.76424e+08	1.09349e+10	350.25	1.53224e+09	3.16333	8	1	0.866019
146	206	5.82392e+07	9.14045e+09	350.25	7.11357e+08	3.09	9	1	0.708449
147	200	446170	1.25812e+08	350.25	7.25021e+06	3.13333	8	1	0.784573
148	215	2.86129e+07	8.57552e+09	350.25	4.9428e+08	3.08	8	1	0.697806
149	199	8.89317e+10	2.66471e+13	350.25	1.5359e+12	3.16	13	1	0.935094
150	190	1.41334e+07	4.23325e+09	350.25	2.43998e+08	3.12333	8	1	0.821455
151	161	7.807e+06	2.33573e+09	350.25	1.34628e+08	3.01333	9	1	0.559603
152	187	9.30416e+07	9.98733e+09	350.25	8.76168e+08	3.14333	8	1	0.690016
153	175	6.03636e+17	1.81091e+20	350.25	1.04378e+19	3.20667	9	1	0.90404
154	193	1.43957e+08	3.17684e+10	350.25	1.90129e+09	3.15	9	1	0.879867
155	199	1.04204e+08	8.27714e+09	350.25	8.97045e+08	3.07667	7	1	0.526741
156	184	1.57626e+10	4.71001e+12	350.25	2.71476e+11	3.18	9	1	0.887468
157	176	7.02416e+07	7.78477e+09	350.25	6.67303e+08	3.1	9	1	0.723418
158	210	5.11282e+12	9.68507e+14	350.25	6.45431e+13	3.15333	9	1	0.80197

159	184	7.42501e+11	2.22717e+14	350.25	1.28371e+13	3.09333	9	1	0.715045
160	176	1.49356e+08	1.0405e+10	350.25	1.07775e+09	3.14333	9	1	0.784935
161	176	4.88353e+17	1.46506e+20	300.917	8.44441e+18	3.1	9	1	0.718795
162	205	1.73334e+08	1.00362e+10	350.25	1.1673e+09	3.09	8	1	0.633956
163	193	1.14322e+11	3.42891e+13	350.25	1.97638e+12	3.14	9	1	0.757452
164	203	8.25905e+07	9.6376e+09	350.25	8.27952e+08	3.14	9	1	0.832907
165	176	1.21216e+13	3.62452e+15	350.25	2.08912e+14	3.12	9	1	0.843959
166	189	5.50493e+07	8.49813e+09	350.25	6.53727e+08	3.09	9	1	0.727026
167	195	6.93243e+07	8.15325e+09	350.25	6.87651e+08	3.09	8	1	0.758441
168	184	2.90107e+07	7.96708e+09	350.25	4.60986e+08	3.13333	9	2	0.659966
169	188	4.7124e+12	1.40979e+15	350.25	8.12581e+13	3.13667	9	1	0.866788
170	187	3.22572e+11	9.67517e+13	350.25	5.57664e+12	3.14333	9	1	0.793382
171	208	1.91845e+08	1.09319e+10	350.25	1.28074e+09	3.15	9	1	0.938527
172	190	1.14947e+08	1.05466e+10	350.25	9.9594e+08	3.15667	9	1	0.874903
173	193	1.81894e+08	3.06895e+10	350.25	1.92908e+09	3.18	8	1	0.825187
174	188	2.00671e+09	6.02006e+11	350.25	3.46988e+10	3.11333	9	1	0.730631
175	213	1.00357e+08	1.4521e+10	350.25	1.04751e+09	3.14667	9	1	0.738347
176	201	4.7562e+12	1.42684e+15	350.25	8.22412e+13	3.14667	8	1	0.855466
177	186	3.98751e+11	1.19589e+14	350.25	6.89294e+12	3.22	9	1	0.958262
178	186	5.86912e+07	8.81468e+09	350.25	6.77114e+08	3.13667	9	1	0.738008
179	201	1.39057e+13	4.1717e+15	350.25	2.40452e+14	3.06333	7	1	0.528509
180	210	1.31493e+08	3.1681e+10	350.25	1.87856e+09	3.11333	8	1	0.783468
181	191	6.18604e+06	9.07141e+08	350.25	7.0879e+07	3.04333	9	1	0.766891
182	198	3.30208e+07	8.45565e+09	350.25	4.91329e+08	3.19	9	1	0.945463
183	203	9.52333e+07	8.87228e+09	350.25	8.47244e+08	3.15	9	1	0.91697
184	205	2.59154e+08	3.43762e+10	350.25	2.26675e+09	3.13667	9	1	0.855174
185	202	2.33502e+07	6.99878e+09	350.25	4.034e+08	3.02	7	1	0.621504
186	204	1.77475e+08	3.09543e+10	350.25	1.93213e+09	3.07	8	1	0.811439
187	200	1.67472e+08	9.39064e+09	350.25	1.17724e+09	3.12667	8	1	0.772845
188	193	6.66057e+07	1.0878e+10	350.25	8.15956e+08	3.10333	8	1	0.761132
189	210	1.00618e+11	3.01593e+13	350.25	1.73834e+12	3.05333	9	1	0.77706
190	179	1.28839e+13	3.86514e+15	350.25	2.22782e+14	3.08	9	1	0.791791
191	203	7.7307e+07	9.07476e+09	350.25	7.89236e+08	3.13667	9	1	0.882037
192	181	9.19056e+07	1.083e+10	350.25	9.21646e+08	3.03	8	1	0.675105
193	186	3.46912e+12	1.04067e+15	350.25	5.99828e+13	3.23333	9	1	1.05778
194	209	3.41596e+12	1.02478e+15	350.25	5.9067e+13	3.19333	9	1	0.957056
195	183	1.17984e+08	9.62189e+09	350.25	9.7315e+08	3.19333	9	1	0.932357
196	188	8.4096e+16	2.52288e+19	350.25	1.45416e+18	3.05667	9	1	0.739452
197	190	1.09397e+08	8.87738e+09	350.25	9.43174e+08	3.19667	9	1	0.958117
198	198	1.06634e+18	3.19902e+20	350.25	1.84387e+19	3.16333	9	1	0.950433
199	208	1.06354e+08	8.59911e+09	350.25	8.90712e+08	3.18	10	1	0.987049
200	203	7.51878e+07	7.5146e+09	350.25	6.92508e+08	3.07667	9	1	0.764279

Resultados

1. Análise de parâmetros e funções

O individuo nessa implementação é gerado utilizando o método **HalfAndHalf** onde metade da expressao é gerada com o **grow** e outra metade com **full**.
Utilizou-se uma probalidade de crossover de 60% e de mutaçao de 10%. Como operador de crossover escolheu-se o cruzamento de ponto único ounde escolhe-se um ponto aleatório da árvore, corta-o e troca com o da outra árvore. Como operador de mutação, escolheu-se randomicamente um ponto em um individuo(árvore), então trocou-se a sub-arvore no ponto gerando um pedaço utilizando o método **full** onde cada folha tem profundidade máxima entre 0 e 2.
Para função de fitness utilizou-se o erro quadrado médio.

Foram feitos varios testes(cenários) com combinações de funções para a Programação Genética, que podem ser resumidos nos três abaixo:

- Cenário simplista
  - Funções: soma e subtração.
  - Terminais: randomico entre [-1,1]
  - Obteve um erro médio de aproximadamente 9% e uma função bem simples. $f = x2 + 13$
- Cenário coerente
  - Funções: soma, subtração, coseno, seno, negação, multiplicação.
  - Terminais: randomico entre [-180,180]
  - Obteve um erro médio de aproximadamente 5% e um função simples também. $f = x1 + 27$
- Cenário complexo
  - Funções: soma, subtração, divisão, coseno, seno, negação, multiplicação, exponencial
  - Terminais: randomico entre [-180,180]
  - Obteve um erro médio superior a 10% e função muito complexa. Árvore muito grande.

Sendo assim o cenário coerente foi utilizado para apresentar os demais resultados.

2. Melhor expressão-S obtida

In [9]:
```
str(hof[0])
```

Out[9]:
```
'add(x1, 27)'
```

3. Equação correspondente(simplificada)

$$f(x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12) = x1 + 27$$

4. Planilha com os dados obtidos

In [21]:
```
import pandas as pd

#Colunas da tabela
cols = ['x1','x2','x3','x4','x5','x6','x7','x8','x9','x10','x11','x12','VALOR REAL','VALOR PG','ERRO(%)']

#Compila  melhor indivíduo chamar a funçã com os resultados
function = gp.compile(hof[0],pset)
table_train = []
for x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,y in data_train:
    pg = function(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12)
    table_train.append([x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,y,pg,round(abs(100-pg*100/y),2)])

table_test = []
for x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,y in data_test:
    pg = function(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12)
    table_test.append([x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,y,pg,round(abs(100-pg*100/y),2)])

print('\n========\n|TREINO|\n========')
table_train = np.array(table_train)
df_train = pd.DataFrame(table_train,columns = cols)
print(df_train.to_string())

print('\n=======\n|TESTE|\n=======')
table_test = np.array(table_test)
df_test = pd.DataFrame(table_test,columns = cols)
print(df_test.to_string())
```

========
|TREINO|
========

	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	VALOR REAL	VALOR PG	ERRO(%)
0	112.0	118.0	132.0	129.0	121.0	135.0	148.0	148.0	136.0	119.0	104.0	118.0	115.0	139.0	20.87
1	118.0	132.0	129.0	121.0	135.0	148.0	148.0	136.0	119.0	104.0	118.0	115.0	126.0	145.0	15.08
2	132.0	129.0	121.0	135.0	148.0	148.0	136.0	119.0	104.0	118.0	115.0	126.0	141.0	159.0	12.77
3	129.0	121.0	135.0	148.0	148.0	136.0	119.0	104.0	118.0	115.0	126.0	141.0	135.0	156.0	15.56
4	121.0	135.0	148.0	148.0	136.0	119.0	104.0	118.0	115.0	126.0	141.0	135.0	125.0	148.0	18.40
5	135.0	148.0	148.0	136.0	119.0	104.0	118.0	115.0	126.0	141.0	135.0	125.0	149.0	162.0	8.72
6	148.0	148.0	136.0	119.0	104.0	118.0	115.0	126.0	141.0	135.0	125.0	149.0	170.0	175.0	2.94
7	148.0	136.0	119.0	104.0	118.0	115.0	126.0	141.0	135.0	125.0	149.0	170.0	170.0	175.0	2.94
8	136.0	119.0	104.0	118.0	115.0	126.0	141.0	135.0	125.0	149.0	170.0	170.0	158.0	163.0	3.16
9	119.0	104.0	118.0	115.0	126.0	141.0	135.0	125.0	149.0	170.0	170.0	158.0	133.0	146.0	9.77
10	104.0	118.0	115.0	126.0	141.0	135.0	125.0	149.0	170.0	170.0	158.0	133.0	114.0	131.0	14.91
11	118.0	115.0	126.0	141.0	135.0	125.0	149.0	170.0	170.0	158.0	133.0	114.0	140.0	145.0	3.57
12	115.0	126.0	141.0	135.0	125.0	149.0	170.0	170.0	158.0	133.0	114.0	140.0	145.0	142.0	2.07
13	126.0	141.0	135.0	125.0	149.0	170.0	170.0	158.0	133.0	114.0	140.0	145.0	150.0	153.0	2.00
14	141.0	135.0	125.0	149.0	170.0	170.0	158.0	133.0	114.0	140.0	145.0	150.0	178.0	168.0	5.62
15	135.0	125.0	149.0	170.0	170.0	158.0	133.0	114.0	140.0	145.0	150.0	178.0	163.0	162.0	0.61
16	125.0	149.0	170.0	170.0	158.0	133.0	114.0	140.0	145.0	150.0	178.0	163.0	172.0	152.0	11.63
17	149.0	170.0	170.0	158.0	133.0	114.0	140.0	145.0	150.0	178.0	163.0	172.0	178.0	176.0	1.12
18	170.0	170.0	158.0	133.0	114.0	140.0	145.0	150.0	178.0	163.0	172.0	178.0	199.0	197.0	1.01
19	170.0	158.0	133.0	114.0	140.0	145.0	150.0	178.0	163.0	172.0	178.0	199.0	199.0	197.0	1.01
20	158.0	133.0	114.0	140.0	145.0	150.0	178.0	163.0	172.0	178.0	199.0	199.0	184.0	185.0	0.54
21	133.0	114.0	140.0	145.0	150.0	178.0	163.0	172.0	178.0	199.0	199.0	184.0	162.0	160.0	1.23
22	114.0	140.0	145.0	150.0	178.0	163.0	172.0	178.0	199.0	199.0	184.0	162.0	146.0	141.0	3.42
23	140.0	145.0	150.0	178.0	163.0	172.0	178.0	199.0	199.0	184.0	162.0	146.0	166.0	167.0	0.60
24	145.0	150.0	178.0	163.0	172.0	178.0	199.0	199.0	184.0	162.0	146.0	166.0	171.0	172.0	0.58
25	150.0	178.0	163.0	172.0	178.0	199.0	199.0	184.0	162.0	146.0	166.0	171.0	180.0	177.0	1.67
26	178.0	163.0	172.0	178.0	199.0	199.0	184.0	162.0	146.0	166.0	171.0	180.0	193.0	205.0	6.22
27	163.0	172.0	178.0	199.0	199.0	184.0	162.0	146.0	166.0	171.0	180.0	193.0	181.0	190.0	4.97
28	172.0	178.0	199.0	199.0	184.0	162.0	146.0	166.0	171.0	180.0	193.0	181.0	183.0	199.0	8.74
29	178.0	199.0	199.0	184.0	162.0	146.0	166.0	171.0	180.0	193.0	181.0	183.0	218.0	205.0	5.96
30	199.0	199.0	184.0	162.0	146.0	166.0	171.0	180.0	193.0	181.0	183.0	218.0	230.0	226.0	1.74
31	199.0	184.0	162.0	146.0	166.0	171.0	180.0	193.0	181.0	183.0	218.0	230.0	242.0	226.0	6.61
32	184.0	162.0	146.0	166.0	171.0	180.0	193.0	181.0	183.0	218.0	230.0	242.0	209.0	211.0	0.96
33	162.0	146.0	166.0	171.0	180.0	193.0	181.0	183.0	218.0	230.0	242.0	209.0	191.0	189.0	1.05
34	146.0	166.0	171.0	180.0	193.0	181.0	183.0	218.0	230.0	242.0	209.0	191.0	172.0	173.0	0.58
35	166.0	171.0	180.0	193.0	181.0	183.0	218.0	230.0	242.0	209.0	191.0	172.0	194.0	193.0	0.52
36	171.0	180.0	193.0	181.0	183.0	218.0	230.0	242.0	209.0	191.0	172.0	194.0	196.0	198.0	1.02
37	180.0	193.0	181.0	183.0	218.0	230.0	242.0	209.0	191.0	172.0	194.0	196.0	196.0	207.0	5.61
38	193.0	181.0	183.0	218.0	230.0	242.0	209.0	191.0	172.0	194.0	196.0	196.0	236.0	220.0	6.78
39	181.0	183.0	218.0	230.0	242.0	209.0	191.0	172.0	194.0	196.0	196.0	236.0	235.0	208.0	11.49
40	183.0	218.0	230.0	242.0	209.0	191.0	172.0	194.0	196.0	196.0	236.0	235.0	229.0	210.0	8.30
41	218.0	230.0	242.0	209.0	191.0	172.0	194.0	196.0	196.0	236.0	235.0	229.0	243.0	245.0	0.82
42	230.0	242.0	209.0	191.0	172.0	194.0	196.0	196.0	236.0	235.0	229.0	243.0	264.0	257.0	2.65
43	242.0	209.0	191.0	172.0	194.0	196.0	196.0	236.0	235.0	229.0	243.0	264.0	272.0	269.0	1.10
44	209.0	191.0	172.0	194.0	196.0	196.0	236.0	235.0	229.0	243.0	264.0	272.0	237.0	236.0	0.42
45	191.0	172.0	194.0	196.0	196.0	236.0	235.0	229.0	243.0	264.0	272.0	237.0	211.0	218.0	3.32
46	172.0	194.0	196.0	196.0	236.0	235.0	229.0	243.0	264.0	272.0	237.0	211.0	180.0	199.0	10.56
47	194.0	196.0	196.0	236.0	235.0	229.0	243.0	264.0	272.0	237.0	211.0	180.0	201.0	221.0	9.95
48	196.0	196.0	236.0	235.0	229.0	243.0	264.0	272.0	237.0	211.0	180.0	201.0	204.0	223.0	9.31
49	196.0	236.0	235.0	229.0	243.0	264.0	272.0	237.0	211.0	180.0	201.0	204.0	188.0	223.0	18.62

50	236.0	235.0	229.0	243.0	264.0	272.0	237.0	211.0	180.0	201.0	204.0	188.0	235.0	263.0	11.91
51	235.0	229.0	243.0	264.0	272.0	237.0	211.0	180.0	201.0	204.0	188.0	235.0	227.0	262.0	15.42
52	229.0	243.0	264.0	272.0	237.0	211.0	180.0	201.0	204.0	188.0	235.0	227.0	234.0	256.0	9.40
53	243.0	264.0	272.0	237.0	211.0	180.0	201.0	204.0	188.0	235.0	227.0	234.0	264.0	270.0	2.27
54	264.0	272.0	237.0	211.0	180.0	201.0	204.0	188.0	235.0	227.0	234.0	264.0	302.0	291.0	3.64
55	272.0	237.0	211.0	180.0	201.0	204.0	188.0	235.0	227.0	234.0	264.0	302.0	293.0	299.0	2.05
56	237.0	211.0	180.0	201.0	204.0	188.0	235.0	227.0	234.0	264.0	302.0	293.0	259.0	264.0	1.93
57	211.0	180.0	201.0	204.0	188.0	235.0	227.0	234.0	264.0	302.0	293.0	259.0	229.0	238.0	3.93
58	180.0	201.0	204.0	188.0	235.0	227.0	234.0	264.0	302.0	293.0	259.0	229.0	203.0	207.0	1.97
59	201.0	204.0	188.0	235.0	227.0	234.0	264.0	302.0	293.0	259.0	229.0	203.0	229.0	228.0	0.44
60	204.0	188.0	235.0	227.0	234.0	264.0	302.0	293.0	259.0	229.0	203.0	229.0	242.0	231.0	4.55
61	188.0	235.0	227.0	234.0	264.0	302.0	293.0	259.0	229.0	203.0	229.0	242.0	233.0	215.0	7.73
62	235.0	227.0	234.0	264.0	302.0	293.0	259.0	229.0	203.0	229.0	242.0	233.0	267.0	262.0	1.87
63	227.0	234.0	264.0	302.0	293.0	259.0	229.0	203.0	229.0	242.0	233.0	267.0	269.0	254.0	5.58
64	234.0	264.0	302.0	293.0	259.0	229.0	203.0	229.0	242.0	233.0	267.0	269.0	270.0	261.0	3.33
65	264.0	302.0	293.0	259.0	229.0	203.0	229.0	242.0	233.0	267.0	269.0	270.0	315.0	291.0	7.62
66	302.0	293.0	259.0	229.0	203.0	229.0	242.0	233.0	267.0	269.0	270.0	315.0	364.0	329.0	9.62
67	293.0	259.0	229.0	203.0	229.0	242.0	233.0	267.0	269.0	270.0	315.0	364.0	347.0	320.0	7.78
68	259.0	229.0	203.0	229.0	242.0	233.0	267.0	269.0	270.0	315.0	364.0	347.0	312.0	286.0	8.33
69	229.0	203.0	229.0	242.0	233.0	267.0	269.0	270.0	315.0	364.0	347.0	312.0	274.0	256.0	6.57
70	203.0	229.0	242.0	233.0	267.0	269.0	270.0	315.0	364.0	347.0	312.0	274.0	237.0	230.0	2.95
71	229.0	242.0	233.0	267.0	269.0	270.0	315.0	364.0	347.0	312.0	274.0	237.0	278.0	256.0	7.91
72	242.0	233.0	267.0	269.0	270.0	315.0	364.0	347.0	312.0	274.0	237.0	278.0	284.0	269.0	5.28
73	233.0	267.0	269.0	270.0	315.0	364.0	347.0	312.0	274.0	237.0	278.0	284.0	277.0	260.0	6.14
74	267.0	269.0	270.0	315.0	364.0	347.0	312.0	274.0	237.0	278.0	284.0	277.0	317.0	294.0	7.26
75	269.0	270.0	315.0	364.0	347.0	312.0	274.0	237.0	278.0	284.0	277.0	317.0	313.0	296.0	5.43
76	270.0	315.0	364.0	347.0	312.0	274.0	237.0	278.0	284.0	277.0	317.0	313.0	318.0	297.0	6.60
77	315.0	364.0	347.0	312.0	274.0	237.0	278.0	284.0	277.0	317.0	313.0	318.0	374.0	342.0	8.56
78	364.0	347.0	312.0	274.0	237.0	278.0	284.0	277.0	317.0	313.0	318.0	374.0	413.0	391.0	5.33
79	347.0	312.0	274.0	237.0	278.0	284.0	277.0	317.0	313.0	318.0	374.0	413.0	405.0	374.0	7.65
80	312.0	274.0	237.0	278.0	284.0	277.0	317.0	313.0	318.0	374.0	413.0	405.0	355.0	339.0	4.51
81	274.0	237.0	278.0	284.0	277.0	317.0	313.0	318.0	374.0	413.0	405.0	355.0	306.0	301.0	1.63
82	237.0	278.0	284.0	277.0	317.0	313.0	318.0	374.0	413.0	405.0	355.0	306.0	271.0	264.0	2.58
83	278.0	284.0	277.0	317.0	313.0	318.0	374.0	413.0	405.0	355.0	306.0	271.0	306.0	305.0	0.33
84	284.0	277.0	317.0	313.0	318.0	374.0	413.0	405.0	355.0	306.0	271.0	306.0	315.0	311.0	1.27
85	277.0	317.0	313.0	318.0	374.0	413.0	405.0	355.0	306.0	271.0	306.0	315.0	301.0	304.0	1.00
86	317.0	313.0	318.0	374.0	413.0	405.0	355.0	306.0	271.0	306.0	315.0	301.0	356.0	344.0	3.37
87	313.0	318.0	374.0	413.0	405.0	355.0	306.0	271.0	306.0	315.0	301.0	356.0	348.0	340.0	2.30
88	318.0	374.0	413.0	405.0	355.0	306.0	271.0	306.0	315.0	301.0	356.0	348.0	355.0	345.0	2.82
89	374.0	413.0	405.0	355.0	306.0	271.0	306.0	315.0	301.0	356.0	348.0	355.0	422.0	401.0	4.98
90	413.0	405.0	355.0	306.0	271.0	306.0	315.0	301.0	356.0	348.0	355.0	422.0	465.0	440.0	5.38
91	405.0	355.0	306.0	271.0	306.0	315.0	301.0	356.0	348.0	355.0	422.0	465.0	467.0	432.0	7.49
92	355.0	306.0	271.0	306.0	315.0	301.0	356.0	348.0	355.0	422.0	465.0	467.0	404.0	382.0	5.45
93	306.0	271.0	306.0	315.0	301.0	356.0	348.0	355.0	422.0	465.0	467.0	404.0	347.0	333.0	4.03
94	271.0	306.0	315.0	301.0	356.0	348.0	355.0	422.0	465.0	467.0	404.0	347.0	305.0	298.0	2.30
95	306.0	315.0	301.0	356.0	348.0	355.0	422.0	465.0	467.0	404.0	347.0	305.0	336.0	333.0	0.89
96	315.0	301.0	356.0	348.0	355.0	422.0	465.0	467.0	404.0	347.0	305.0	336.0	340.0	342.0	0.59
97	301.0	356.0	348.0	355.0	422.0	465.0	467.0	404.0	347.0	305.0	336.0	340.0	318.0	328.0	3.14
98	356.0	348.0	355.0	422.0	465.0	467.0	404.0	347.0	305.0	336.0	340.0	318.0	362.0	383.0	5.80
99	348.0	355.0	422.0	465.0	467.0	404.0	347.0	305.0	336.0	340.0	318.0	362.0	348.0	375.0	7.76
100	355.0	422.0	465.0	467.0	404.0	347.0	305.0	336.0	340.0	318.0	362.0	348.0	363.0	382.0	5.23
101	422.0	465.0	467.0	404.0	347.0	305.0	336.0	340.0	318.0	362.0	348.0	363.0	435.0	449.0	3.22
102	465.0	467.0	404.0	347.0	305.0	336.0	340.0	318.0	362.0	348.0	363.0	435.0	491.0	492.0	0.20
103	467.0	404.0	347.0	305.0	336.0	340.0	318.0	362.0	348.0	363.0	435.0	491.0	505.0	494.0	2.18

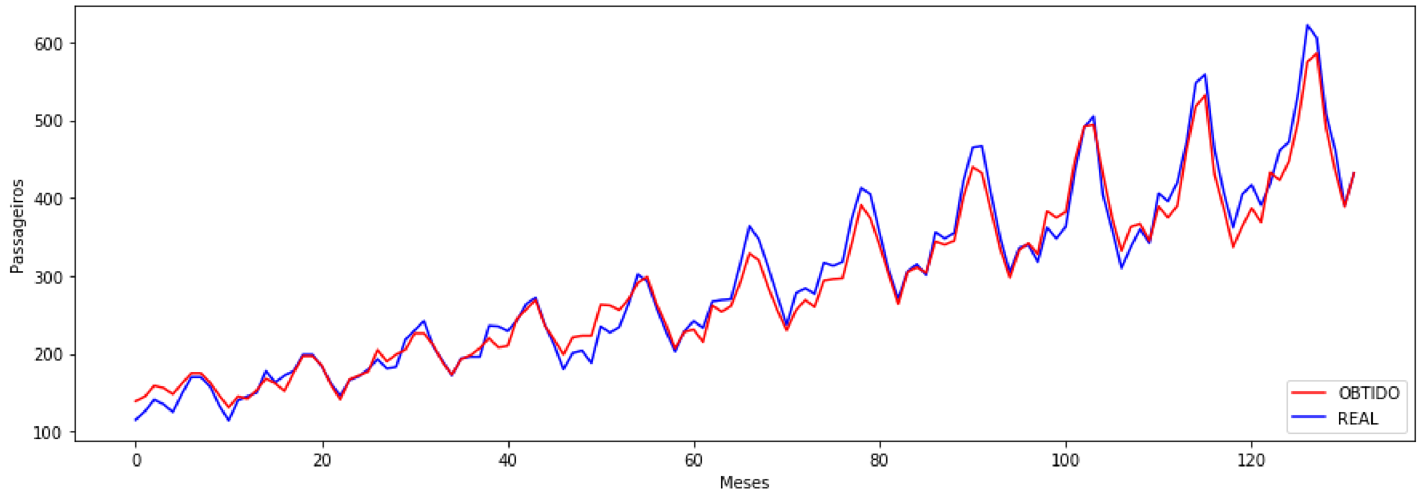
104	404.0	347.0	305.0	336.0	340.0	318.0	362.0	348.0	363.0	435.0	491.0	505.0	404.0	431.0	6.68
105	347.0	305.0	336.0	340.0	318.0	362.0	348.0	363.0	435.0	491.0	505.0	404.0	359.0	374.0	4.18
106	305.0	336.0	340.0	318.0	362.0	348.0	363.0	435.0	491.0	505.0	404.0	359.0	310.0	332.0	7.10
107	336.0	340.0	318.0	362.0	348.0	363.0	435.0	491.0	505.0	404.0	359.0	310.0	337.0	363.0	7.72
108	340.0	318.0	362.0	348.0	363.0	435.0	491.0	505.0	404.0	359.0	310.0	337.0	360.0	367.0	1.94
109	318.0	362.0	348.0	363.0	435.0	491.0	505.0	404.0	359.0	310.0	337.0	360.0	342.0	345.0	0.88
110	362.0	348.0	363.0	435.0	491.0	505.0	404.0	359.0	310.0	337.0	360.0	342.0	406.0	389.0	4.19
111	348.0	363.0	435.0	491.0	505.0	404.0	359.0	310.0	337.0	360.0	342.0	406.0	396.0	375.0	5.30
112	363.0	435.0	491.0	505.0	404.0	359.0	310.0	337.0	360.0	342.0	406.0	396.0	420.0	390.0	7.14
113	435.0	491.0	505.0	404.0	359.0	310.0	337.0	360.0	342.0	406.0	396.0	420.0	472.0	462.0	2.12
114	491.0	505.0	404.0	359.0	310.0	337.0	360.0	342.0	406.0	396.0	420.0	472.0	548.0	518.0	5.47
115	505.0	404.0	359.0	310.0	337.0	360.0	342.0	406.0	396.0	420.0	472.0	548.0	559.0	532.0	4.83
116	404.0	359.0	310.0	337.0	360.0	342.0	406.0	396.0	420.0	472.0	548.0	559.0	463.0	431.0	6.91
117	359.0	310.0	337.0	360.0	342.0	406.0	396.0	420.0	472.0	548.0	559.0	463.0	407.0	386.0	5.16
118	310.0	337.0	360.0	342.0	406.0	396.0	420.0	472.0	548.0	559.0	463.0	407.0	362.0	337.0	6.91
119	337.0	360.0	342.0	406.0	396.0	420.0	472.0	548.0	559.0	463.0	407.0	362.0	405.0	364.0	10.12

=======
|TESTE|
=======

	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	VALOR REAL	VALOR PG	ERRO(%)
0	360.0	342.0	406.0	396.0	420.0	472.0	548.0	559.0	463.0	407.0	362.0	405.0	417.0	387.0	7.19
1	342.0	406.0	396.0	420.0	472.0	548.0	559.0	463.0	407.0	362.0	405.0	417.0	391.0	369.0	5.63
2	406.0	396.0	420.0	472.0	548.0	559.0	463.0	407.0	362.0	405.0	417.0	391.0	419.0	433.0	3.34
3	396.0	420.0	472.0	548.0	559.0	463.0	407.0	362.0	405.0	417.0	391.0	419.0	461.0	423.0	8.24
4	420.0	472.0	548.0	559.0	463.0	407.0	362.0	405.0	417.0	391.0	419.0	461.0	472.0	447.0	5.30
5	472.0	548.0	559.0	463.0	407.0	362.0	405.0	417.0	391.0	419.0	461.0	472.0	535.0	499.0	6.73
6	548.0	559.0	463.0	407.0	362.0	405.0	417.0	391.0	419.0	461.0	472.0	535.0	622.0	575.0	7.56
7	559.0	463.0	407.0	362.0	405.0	417.0	391.0	419.0	461.0	472.0	535.0	622.0	606.0	586.0	3.30
8	463.0	407.0	362.0	405.0	417.0	391.0	419.0	461.0	472.0	535.0	622.0	606.0	508.0	490.0	3.54
9	407.0	362.0	405.0	417.0	391.0	419.0	461.0	472.0	535.0	622.0	606.0	508.0	461.0	434.0	5.86
10	362.0	405.0	417.0	391.0	419.0	461.0	472.0	535.0	622.0	606.0	508.0	461.0	390.0	389.0	0.26
11	405.0	417.0	391.0	419.0	461.0	472.0	535.0	622.0	606.0	508.0	461.0	390.0	432.0	432.0	0.00

5. Gráfico com os dados reais e os gerados pela função obtida por PG

In [11]:
```
import matplotlib.pyplot as plt

meses = np.arange(132)


fig, ax1 = plt.subplots()
ax1.set_xlabel("Meses")
ax1.set_ylabel("Passageiros")

line1 = ax1.plot(meses, np.concatenate((table_train[:,12],table_test[:,12]),axis=0), "b-", label="REAL")
line2 = ax1.plot(meses, np.concatenate((table_train[:,13],table_test[:,13]),axis=0), "r-", label="OBTIDO")


lns = line2 + line1
labs = [l.get_label() for l in lns]
ax1.legend(lns, labs, loc=4)

fig.set_size_inches(15, 5, forward=True)

plt.show()
```



In [17]:
```
error_final = np.concatenate((table_train[:,14],table_test[:,14]),axis=0)
print('Erro médio treino: {}%'.format(sum(table_train[:,14])/len(table_train)))
print('Erro médio teste: {}%'.format(sum(table_test[:,14])/len(table_test)))
print('Erro médio total: {}%'.format(sum(error_final)/len(error_final)))
```

```
Erro médio treino: 5.277083333333333%
Erro médio teste: 4.745833333333333%
Erro médio total: 5.228787878787878%
```

6. Considerações finais

- Aumentando-se o numero de gerações não contribui para uma melhora do erro.
- As equações que envolviam seno e coseno não apresentaram-se melhor.
- Os dados iniciais apresentam um erro maior pois, devido a estrutura do problema, eles nao têm dados suficientes de treinamento.
- A adição e funções mais complexas como exponencial implica em incluir restrições para não dar erro ao calcular o fitness ou gerar um filho.