Tarea 3.1

Encontrar, usando el método de AGs, la raíz real del polinomio P(x) = 5x5- 3x4-x3-5x2-x-3 en el intervalo [0.5 1], usando números reales, una población de 100 listas, un solo tipo de cruce y selección por clasificación. Dibuje un cuadro de aptitud contra generación.

Para resolver este ejercicio, se realiza mediante la implementación de AGs.

```
import math               # Libreria que se usa para realizar operaciones matemátic
import random             # Libreria usada para generar números aleatorios
import numpy as np        # Encuentra una raiz real entre 0.5 y 1.5
import matplotlib.pyplot as plt
```

Para poder evidenciar de mejor modo el desarrollo del ejercicio, se tienen cromosomas de longitud 100 bits, una poblacion de solamente 10 individuos, 200 generaciones, y 1% de mutación. Se tiene una población tan reducida ya que de lo contrario, la obtención del problema sería muy rápida.

```
longCrom = 100
K = 100
M = 200
pm = 0.01
```

```
def ecuacion(x):
    valor = 5*x**5-3*x**4-x**3-5*x**2-x-3
    return valor
```

Definición de Función Objetivo (ecuacion): Se define una función no lineal llamada ecuacion que toma una variable x como entrada y devuelve el valor de la función a optimizar. En este caso, la función es un polinomio de quinto grado. El objetivo es encontrar el valor de x que minimiza esta función.

```
def genera(K,longCrom):
    Pob_nueva = np.zeros([K, longCrom], dtype = int)
    cromosoma = np.zeros([longCrom-1], dtype = int)
    j = 0
    while j < K:
        cromosoma = [random.randint(0,1) for i in range(longCrom)]
        Pob_nueva[j] = cromosoma
        j +=1
    return Pob_nueva
```

Generación de la Población Inicial (genera): Se define una función llamada genera que genera una población inicial de cromosomas. En este contexto, un cromosoma representa una posible solución. La población inicial consiste en una matriz de cromosomas binarios.

```
def evalua( Pob_nueva):
    Apt_total = 0
    vectorX = np.zeros(K, dtype = float)
    probab = np.zeros(K, dtype = float)
    aptitud = np.zeros(K, dtype = float)
    mejor = 50.
    mejorX = 0.

    i = 0
    while(i < K):
        t,x = res_Funcion(Pob_nueva [i])
        vectorX[i] = x
        aptitud[i] = (10 - abs(t))*100
        i += 1
    Apt_total = float(sum(aptitud))
    probab = 10 * [j/Apt_total for j in aptitud]
    maxprobab = np.amax(probab)
    maxIndex = np.argmax(probab)
    mejorx = vectorX[maxIndex]
    probab[maxIndex] = 0.99
    return probab, vectorX, max(aptitud)

def res_Funcion(cromosoma):
    x = decodificar(cromosoma)
    funcion= ecuacion(x)
    return funcion, x

def decodificar(cromosoma):
    xi = 0.5  # Limite inferior en el dominio real
    xf = 1  # Limite superior en el dominio real
    Max = 2 **(longCrom)
    cromPot = [cromosoma[i]*2**(longCrom-i-1) for i in range(longCrom)]
    valorDecimal = sum(cromPot)
    cromPot = 0
    valDeco = ((xf-xi)/Max)*(valorDecimal)+xi
    return valDeco
```

Evaluación de Aptitud (evalua): Se define una función llamada evalua que evalúa la aptitud de cada cromosoma en la población actual. La aptitud se calcula en función del valor de la función objetivo. Además, se realiza una selección basada en la aptitud para determinar cuáles cromosomas tienen más probabilidades de reproducirse. Decodificación de Cromosomas (decodificar): La función decodificar se utiliza para convertir un cromosoma binario en un valor real en un rango predefinido.

```
def cruce(Pob_nueva, Probabilidad):
    maxprobab = np.amax(prob_cromosoma)
    maxIndex = np.argmax(prob_cromosoma)
    len(Pob_nueva)
    i = 0
    while (i < K-1):
        mejorCrom = Pob_nueva[maxIndex]
        if Probabilidad[i] < 0.97:
            rand = random.randint(2,longCrom-1) # Posición de cruce aleatoria
            padre1 = Pob_nueva[i]
            padre2 = Pob_nueva[i+1]
            j = rand
            while(j < longCrom):
                bit = padre1[j]
                padre1[j] = padre2[j]
                padre2[j] = bit
                j += 1
        else:
            if maxIndex %2 == 0:
                Pob_nueva[i] = Pob_nueva[maxIndex]
            else:
                Pob_nueva[i+1] = Pob_nueva[maxIndex]
        i += 2
    return Pob_nueva
```

Cruce de Cromosomas (cruce): La función cruce implementa la operación de cruce (reproducción) entre cromosomas. Se utiliza una estrategia de cruce de un punto aleatorio.

```
def muta(Pob_nueva, pm):
    totalbits = K * longCrom
    segmento = 1/pm
    n_segmentos = totalbits/segmento
    i =0
    while(i < n_segmentos-1):
        aleatorio = random.randint(0, segmento-1)
        posic = int(i * segmento + aleatorio)
        y = int((posic/longCrom))
        cromosoma = Pob_nueva[y]
        x = posic - longCrom * y
        if (cromosoma[x-1] == 0):
            cromosoma[x-1] = 1
        else:
            cromosoma[x-1] = 0
        i +=1
    return Pob_nueva
```

Mutación de Cromosomas (muta): La función muta se encarga de aplicar mutaciones aleatorias a los cromosomas con una probabilidad especificada.

```
def seleccion_ruleta(poblacion, probabilidad):
    chosen = []
    while len(chosen)<= K:
        for n in range(K):
            r = random.random()
            for (i, individuo) in enumerate(poblacion):
                #print(i)
                if i > K:
                    break
                if r <= probabilidad[i]:
                    chosen.append(list(individuo))
                    break
    return chosen
```

Selección de Ruleta (seleccion_ruleta): Esta función realiza la selección de cromosomas basada en una ruleta ponderada. Los cromosomas con una aptitud más alta tienen una mayor probabilidad de ser seleccionados.

Ya definidas todas las funciones, se realizan las iteraciones para encontrar la mejor solución.

Después de cada generación se imprime la mejor solución. Si hay un resultado de valor menor a 0.00001, se terminan los ciclos y se dice que se llegó a una solución.

Por último, al final se generan gráficas donde se muestran los resultados del error y del valor determinado por el algoritmo.

```
Pob_nueva3 = genera(K,longCrom)
prob_cromosoma,vectorX,AptI = evalua(Pob_nueva3)
i = 0

Px=list()
Py=list()
PaptI=list()
while (i < M):
    Pob_vieja = Pob_nueva3
    Pob_nueva1 = seleccion_ruleta(Pob_vieja, prob_cromosoma)
    Pob_nueva2 = cruce(Pob_nueva1, prob_cromosoma)
    Pob_nueva3 = muta(Pob_nueva2, pm)
    prob_cromosoma, vectorX,AptI = evalua(Pob_nueva3)
    maxprobab = np.amax(prob_cromosoma)
    maxIndex = np.argmax(prob_cromosoma)
    mejorx = vectorX[maxIndex]
    x = mejorx
    val = ecuacion(x)
    print("Generación ", i,"Mejor x", x,"mejor solucion ",val)
    if abs(val) < 0.00001:
        print()
        print("Mejor Solucion x", x," con F(x) = ",val)
        i = M
    Px.append(i)
    Py.append(val)
    PaptI.append(AptI)
    i += 1

plt.plot(Px, Py)
plt.xlabel('Generación')
plt.ylabel('Valor')
plt.show()

plt.plot(Px, PaptI)
plt.xlabel('Generación')
plt.ylabel('Aptitud máxima')
plt.show()
```

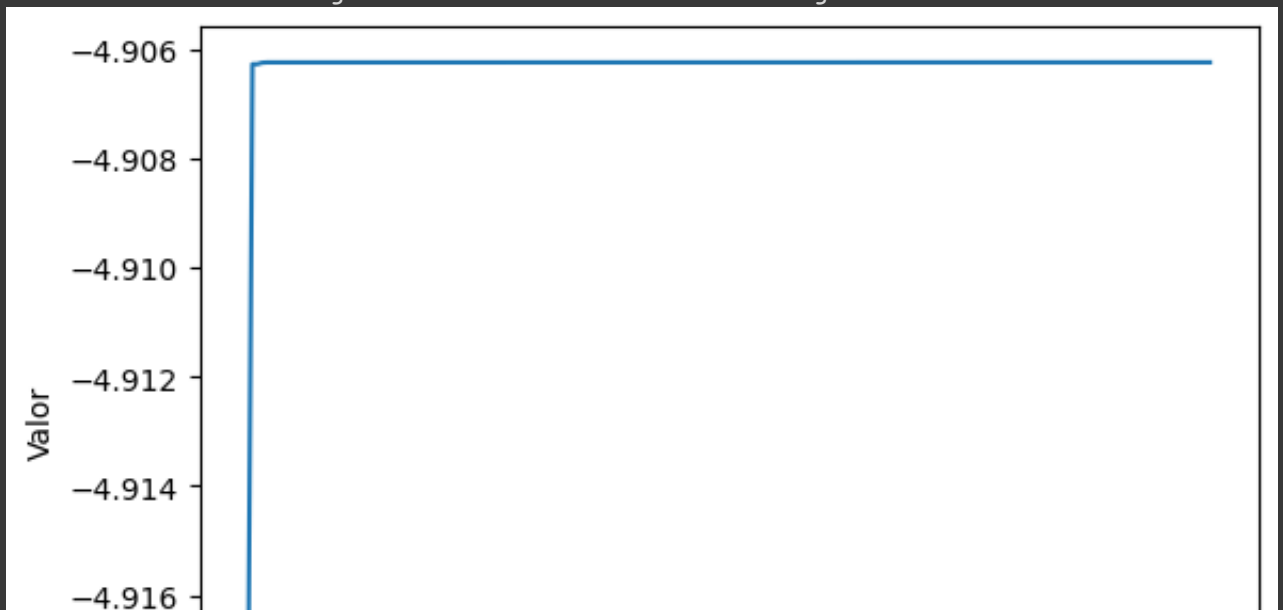
```
Generación  0 Mejor x 0.5019583122124329 mejor solucion  -4.919364387662153
```

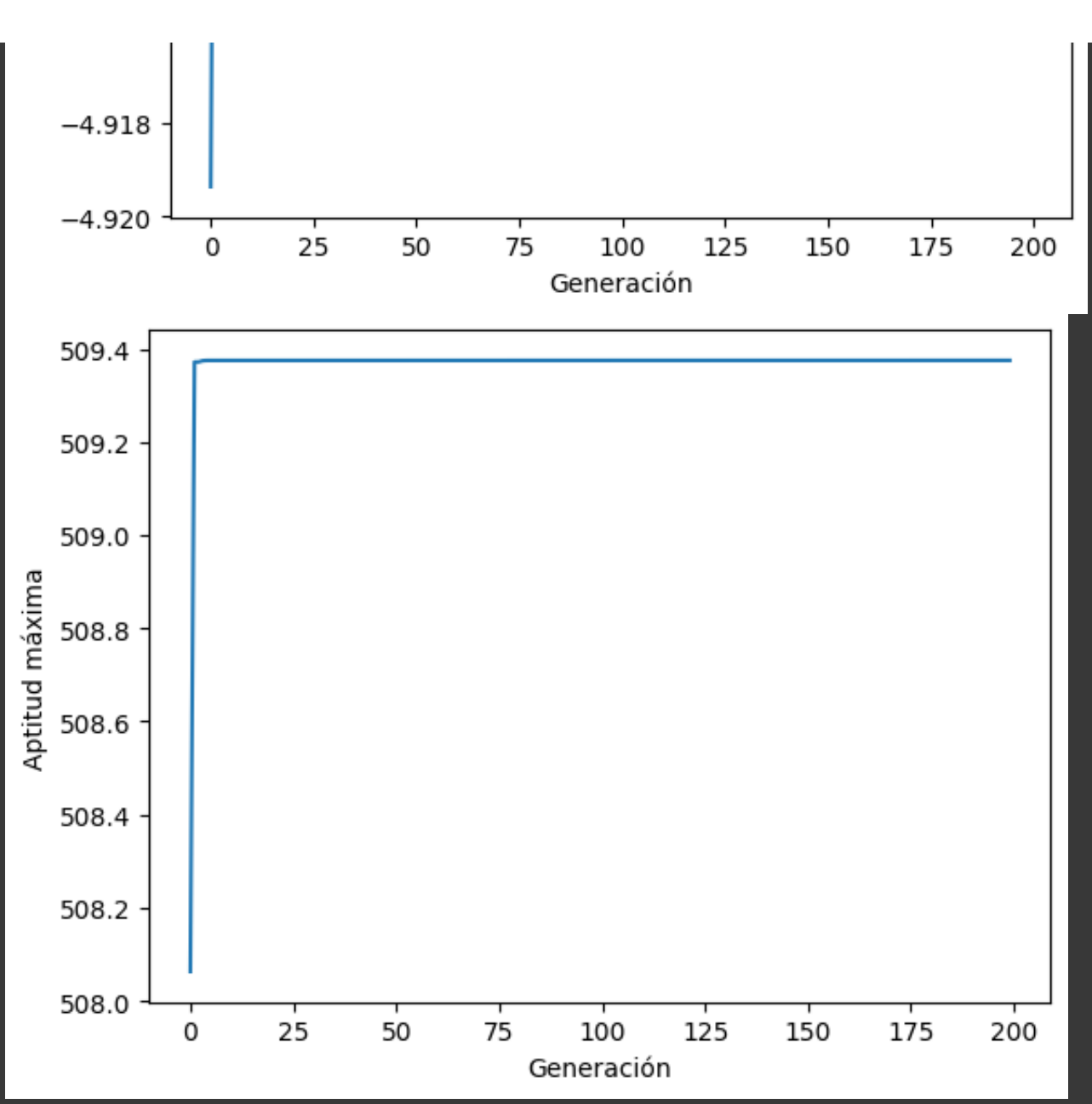
```
Generación  1 Mejor x 0.5000051872124329 mejor solucion  -4.9062846896109535
Generación  2 Mejor x 0.5000049487938538 mejor solucion  -4.906283095175227
Generación  3 Mejor x 0.5000011340965882 mejor solucion  -4.906257584277043
Generación  4 Mejor x 0.5000001804222718 mejor solucion  -4.9062512065740975
Generación  5 Mejor x 0.5000001799566105 mejor solucion  -4.906251203459987
Generación  6 Mejor x 0.500000060747321 mejor solucion  -4.906250406247726
Generación  7 Mejor x 0.5000000598159984 mejor solucion  -4.9062504000195055
Generación  8 Mejor x 0.500000059699583 mejor solucion  -4.906250399240978
Generación  9 Mejor x 0.5000000596923071 mejor solucion  -4.90625039919232
Generación  10 Mejor x 0.5000000596632033 mejor solucion  -4.906250398997688
Generación  11 Mejor x 0.5000000000585585 mejor solucion  -4.90625000039161
Generación  12 Mejor x 0.5000000000003508 mejor solucion  -4.906250000002347
Generación  13 Mejor x 0.5000000000002371 mejor solucion  -4.906250000001586
Generación  14 Mejor x 0.5000000000000098 mejor solucion  -4.906250000000066
Generación  15 Mejor x 0.5000000000000027 mejor solucion  -4.906250000000018
Generación  16 Mejor x 0.5000000000000009 mejor solucion  -4.906250000000006
Generación  17 Mejor x 0.5 mejor solucion  -4.90625
Generación  18 Mejor x 0.5 mejor solucion  -4.90625
Generación  19 Mejor x 0.5 mejor solucion  -4.90625
Generación  20 Mejor x 0.5 mejor solucion  -4.90625
Generación  21 Mejor x 0.5 mejor solucion  -4.90625
Generación  22 Mejor x 0.5 mejor solucion  -4.90625
Generación  23 Mejor x 0.5 mejor solucion  -4.90625
Generación  24 Mejor x 0.5 mejor solucion  -4.90625
Generación  25 Mejor x 0.5 mejor solucion  -4.90625
Generación  26 Mejor x 0.5 mejor solucion  -4.90625
Generación  27 Mejor x 0.5 mejor solucion  -4.90625
Generación  28 Mejor x 0.5 mejor solucion  -4.90625
Generación  29 Mejor x 0.5 mejor solucion  -4.90625
Generación  30 Mejor x 0.5 mejor solucion  -4.90625
Generación  31 Mejor x 0.5 mejor solucion  -4.90625
Generación  32 Mejor x 0.5 mejor solucion  -4.90625
Generación  33 Mejor x 0.5 mejor solucion  -4.90625
Generación  34 Mejor x 0.5 mejor solucion  -4.90625
Generación  35 Mejor x 0.5 mejor solucion  -4.90625
Generación  36 Mejor x 0.5 mejor solucion  -4.90625
Generación  37 Mejor x 0.5 mejor solucion  -4.90625
Generación  38 Mejor x 0.5 mejor solucion  -4.90625
Generación  39 Mejor x 0.5 mejor solucion  -4.90625
Generación  40 Mejor x 0.5 mejor solucion  -4.90625
Generación  41 Mejor x 0.5 mejor solucion  -4.90625
Generación  42 Mejor x 0.5 mejor solucion  -4.90625
Generación  43 Mejor x 0.5 mejor solucion  -4.90625
Generación  44 Mejor x 0.5 mejor solucion  -4.90625
Generación  45 Mejor x 0.5 mejor solucion  -4.90625
Generación  46 Mejor x 0.5 mejor solucion  -4.90625
Generación  47 Mejor x 0.5 mejor solucion  -4.90625
Generación  48 Mejor x 0.5 mejor solucion  -4.90625
Generación  49 Mejor x 0.5 mejor solucion  -4.90625
Generación  50 Mejor x 0.5 mejor solucion  -4.90625
Generación  51 Mejor x 0.5 mejor solucion  -4.90625
Generación  52 Mejor x 0.5 mejor solucion  -4.90625
Generación  53 Mejor x 0.5 mejor solucion  -4.90625
Generación  54 Mejor x 0.5 mejor solucion  -4.90625
```

```
Generación  55 Mejor x 0.5 mejor solucion  -4.90625
Generación  56 Mejor x 0.5 mejor solucion  -4.90625
Generación  57 Mejor x 0.5 mejor solucion  -4.90625
Generación  58 Mejor x 0.5 mejor solucion  -4.90625
Generación  59 Mejor x 0.5 mejor solucion  -4.90625
Generación  60 Mejor x 0.5 mejor solucion  -4.90625
Generación  61 Mejor x 0.5 mejor solucion  -4.90625
Generación  62 Mejor x 0.5 mejor solucion  -4.90625
Generación  63 Mejor x 0.5 mejor solucion  -4.90625
Generación  64 Mejor x 0.5 mejor solucion  -4.90625
Generación  65 Mejor x 0.5 mejor solucion  -4.90625
Generación  66 Mejor x 0.5 mejor solucion  -4.90625
Generación  67 Mejor x 0.5 mejor solucion  -4.90625
Generación  68 Mejor x 0.5 mejor solucion  -4.90625
Generación  69 Mejor x 0.5 mejor solucion  -4.90625
Generación  70 Mejor x 0.5 mejor solucion  -4.90625
Generación  71 Mejor x 0.5 mejor solucion  -4.90625
Generación  72 Mejor x 0.5 mejor solucion  -4.90625
Generación  73 Mejor x 0.5 mejor solucion  -4.90625
Generación  74 Mejor x 0.5 mejor solucion  -4.90625
Generación  75 Mejor x 0.5 mejor solucion  -4.90625
Generación  76 Mejor x 0.5 mejor solucion  -4.90625
Generación  77 Mejor x 0.5 mejor solucion  -4.90625
Generación  78 Mejor x 0.5 mejor solucion  -4.90625
Generación  79 Mejor x 0.5 mejor solucion  -4.90625
Generación  80 Mejor x 0.5 mejor solucion  -4.90625
Generación  81 Mejor x 0.5 mejor solucion  -4.90625
Generación  82 Mejor x 0.5 mejor solucion  -4.90625
Generación  83 Mejor x 0.5 mejor solucion  -4.90625
Generación  84 Mejor x 0.5 mejor solucion  -4.90625
Generación  85 Mejor x 0.5 mejor solucion  -4.90625
Generación  86 Mejor x 0.5 mejor solucion  -4.90625
Generación  87 Mejor x 0.5 mejor solucion  -4.90625
Generación  88 Mejor x 0.5 mejor solucion  -4.90625
Generación  89 Mejor x 0.5 mejor solucion  -4.90625
Generación  90 Mejor x 0.5 mejor solucion  -4.90625
Generación  91 Mejor x 0.5 mejor solucion  -4.90625
Generación  92 Mejor x 0.5 mejor solucion  -4.90625
Generación  93 Mejor x 0.5 mejor solucion  -4.90625
Generación  94 Mejor x 0.5 mejor solucion  -4.90625
Generación  95 Mejor x 0.5 mejor solucion  -4.90625
Generación  96 Mejor x 0.5 mejor solucion  -4.90625
Generación  97 Mejor x 0.5 mejor solucion  -4.90625
Generación  98 Mejor x 0.5 mejor solucion  -4.90625
Generación  99 Mejor x 0.5 mejor solucion  -4.90625
Generación  100 Mejor x 0.5 mejor solucion  -4.90625
Generación  101 Mejor x 0.5 mejor solucion  -4.90625
Generación  102 Mejor x 0.5 mejor solucion  -4.90625
Generación  103 Mejor x 0.5 mejor solucion  -4.90625
Generación  104 Mejor x 0.5 mejor solucion  -4.90625
Generación  105 Mejor x 0.5 mejor solucion  -4.90625
Generación  106 Mejor x 0.5 mejor solucion  -4.90625
Generación  107 Mejor x 0.5 mejor solucion  -4.90625
Generación  108 Mejor x 0.5 mejor solucion  -4.90625
```

```
Generación  109 Mejor x 0.5 mejor solucion  -4.90625
Generación  110 Mejor x 0.5 mejor solucion  -4.90625
Generación  111 Mejor x 0.5 mejor solucion  -4.90625
Generación  112 Mejor x 0.5 mejor solucion  -4.90625
Generación  113 Mejor x 0.5 mejor solucion  -4.90625
Generación  114 Mejor x 0.5 mejor solucion  -4.90625
Generación  115 Mejor x 0.5 mejor solucion  -4.90625
Generación  116 Mejor x 0.5 mejor solucion  -4.90625
Generación  117 Mejor x 0.5 mejor solucion  -4.90625
Generación  118 Mejor x 0.5 mejor solucion  -4.90625
Generación  119 Mejor x 0.5 mejor solucion  -4.90625
Generación  120 Mejor x 0.5 mejor solucion  -4.90625
Generación  121 Mejor x 0.5 mejor solucion  -4.90625
Generación  122 Mejor x 0.5 mejor solucion  -4.90625
Generación  123 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  124 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  125 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  126 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  127 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  128 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  129 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  130 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  131 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  132 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  133 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  134 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  135 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  136 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  137 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  138 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  139 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  140 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  141 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  142 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  143 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  144 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  145 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  146 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  147 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  148 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  149 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  150 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  151 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  152 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  153 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  154 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  155 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  156 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  157 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  158 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  159 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  160 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  161 Mejor x 0.5 mejor solucion  -4.90625
Generación  162 Mejor x 0.5 mejor solucion  -4.90625
```

```
Generación  163 Mejor x 0.5 mejor solucion  -4.90625
Generación  164 Mejor x 0.5 mejor solucion  -4.90625
Generación  165 Mejor x 0.5 mejor solucion  -4.90625
Generación  166 Mejor x 0.5 mejor solucion  -4.90625
Generación  167 Mejor x 0.5 mejor solucion  -4.90625
Generación  168 Mejor x 0.5 mejor solucion  -4.90625
Generación  169 Mejor x 0.5 mejor solucion  -4.90625
Generación  170 Mejor x 0.5 mejor solucion  -4.90625
Generación  171 Mejor x 0.5 mejor solucion  -4.90625
Generación  172 Mejor x 0.5 mejor solucion  -4.90625
Generación  173 Mejor x 0.5 mejor solucion  -4.90625
Generación  174 Mejor x 0.5 mejor solucion  -4.90625
Generación  175 Mejor x 0.5 mejor solucion  -4.90625
Generación  176 Mejor x 0.5 mejor solucion  -4.90625
Generación  177 Mejor x 0.5 mejor solucion  -4.90625
Generación  178 Mejor x 0.5 mejor solucion  -4.90625
Generación  179 Mejor x 0.5 mejor solucion  -4.90625
Generación  180 Mejor x 0.5 mejor solucion  -4.90625
Generación  181 Mejor x 0.5 mejor solucion  -4.90625
Generación  182 Mejor x 0.5 mejor solucion  -4.90625
Generación  183 Mejor x 0.5 mejor solucion  -4.90625
Generación  184 Mejor x 0.5 mejor solucion  -4.90625
Generación  185 Mejor x 0.5 mejor solucion  -4.90625
Generación  186 Mejor x 0.5 mejor solucion  -4.90625
Generación  187 Mejor x 0.5 mejor solucion  -4.90625
Generación  188 Mejor x 0.5 mejor solucion  -4.90625
Generación  189 Mejor x 0.5 mejor solucion  -4.90625
Generación  190 Mejor x 0.5 mejor solucion  -4.90625
Generación  191 Mejor x 0.5 mejor solucion  -4.90625
Generación  192 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  193 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  194 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  195 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  196 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  197 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  198 Mejor x 0.5000000000000001 mejor solucion  -4.90625
Generación  199 Mejor x 0.5000000000000001 mejor solucion  -4.90625
```





Iteraciones Principales: El código principal realiza un ciclo de iteraciones para encontrar la solución óptima. Se ejecutan un número fijo de generaciones (M). En cada generación, se realiza una selección de cromosomas, seguida de cruce y mutación. Luego, se evalúa la aptitud de los cromosomas resultantes y se imprime la mejor solución encontrada hasta el momento.

El objetivo de este código es encontrar un valor de x que minimice la función no lineal ecuacion. El algoritmo genético se utiliza para explorar el espacio de búsqueda y encontrar una solución aproximada al mínimo de la función. Las gráficas generadas muestran cómo la aptitud y los valores de la función cambian a lo largo de las generaciones.

El código se ejecuta durante un número fijo de generaciones, y se detiene si se encuentra una solución cercana al mínimo deseado (cuando abs(val) < 0.00001). En ese caso, se imprime la solución encontrada y se finaliza la ejecución.

Este tipo de algoritmo genético es útil para encontrar soluciones aproximadas en problemas de optimización donde no se puede aplicar un enfoque analítico directo para encontrar el mínimo de la función. En lugar de eso, se utiliza un enfoque de búsqueda heurística basado en la evolución de poblaciones.