|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **НУЛП, ІКНІ, САП** | | Тема | оцінка | підпис |
| СПКс-11 | 4 | Оптимізація цільової функції за допомогою генетичного алгоритму |  |  |
| Киценюк М.Л. | |
| № залікової: 1508503 | |
| Методи нечіткої логіки та еволюційні алгоритми при автоматизованому проектуванні | | Викладач: | |
| Кривий Р.З. | |

**Мета роботи**: реалізувати генетичний алгоритм пошуку максимального і мінімального значення цільової функції згідно варіанту.

**Завдання:**

**Варіант -4**

Розробити програму, яка реалізовує генетичний алгоритм пошуку максимального і мінімального значення цільової функції - f(x) = 4- 5x +47x2 -3dx3 в інтервалі x = [-10, 53].



**Максимальне і мінімальне значення цільової функції**

Графік функції згідно варіанту показаний на рис.1.

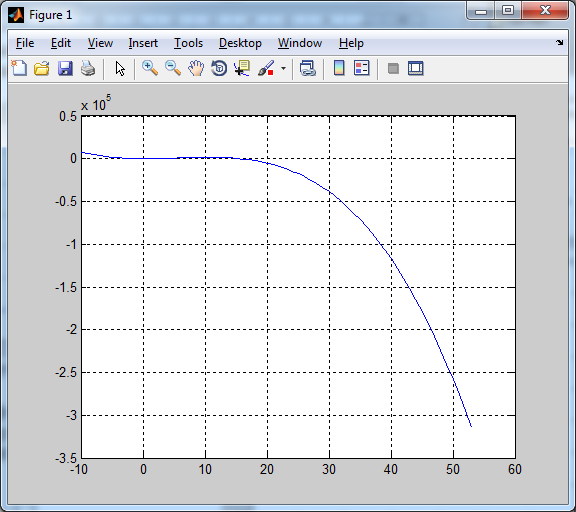


Рис.1. Графік функції на інтервалі [-10;53]

Для знаходження мінімуму функції було вибрано інтервал [0;53]. Пошук мінімуму цільової функції для перевірки реалізовано у Matlab за допомогою функції fminbnd() і рівний: -3.1487e+05, при x = 53.0000.

Для знаходження максимуму функції було вибрано інтервал [-10;10]. Пошук мінімуму цільової функції для перевірки реалізовано у Matlab за допомогою функції fminbnd() змінивши функцію на протилежну і рівний: 7.7519e+03 при x = -10.0000

. Графік функції на інтервалі [-10;10] показаний на рис.2.

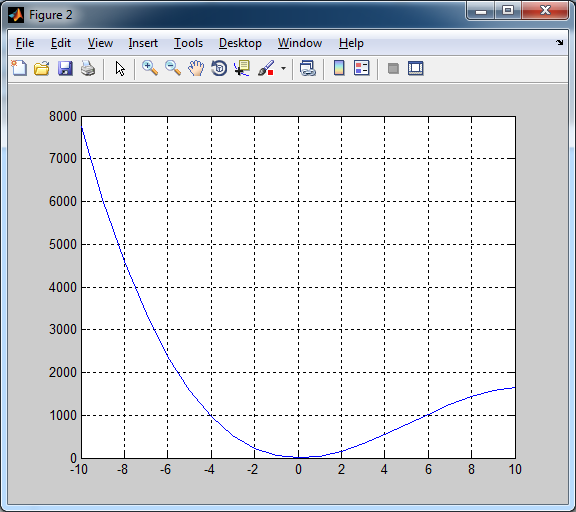


Рис.2. Графік функції на інтервалі[-10;10]

**Результати виконання програми**

Для виконання індивідуального завдання була вибрано використовувати турнірний відбір, одно точковий кросинговер і точкова мутація.

Результати:

|  |  |  |  |
| --- | --- | --- | --- |
| Кількість поколінь, розмір популяції | 10 | 100 | 300 |
| fmin | -527264.75 | -569032.4 | -578881.56 |
| fmax | 1.1711334 | 2.1166277 | 2.1333163 |

Отже похибка при знаходженні мінімуму і максимуму залежать від кількості поколінь і розміру популяції, при їхньому збільшенні похибки прямують до 0.

Результат виконання програми для десяти поколінь з розміром популяції 10 для знаходження мінімуму функції:

Початкова популяція

0 [31.9825] fitness = -146376.1

1 [3.7723591] fitness = -846.7543

2 [4.8147783] fitness = -1446.4819

3 [23.747078] fitness = -66795.77

4 [46.059452] fitness = -393079.2

5 [8.350518] fitness = -5063.9907

6 [14.823505] fitness = -20171.512

7 [44.029438] fitness = -347397.22

8 [41.452637] fitness = -294653.2

9 [23.477943] fitness = -64846.543

Нова популяція

0 [46.059452] fitness = -393079.2

1 [46.059452] fitness = -393079.2

2 [46.059452] fitness = -393079.2

3 [46.059452] fitness = -393079.2

4 [46.059452] fitness = -393079.2

5 [46.059452] fitness = -393079.2

6 [44.029438] fitness = -347397.22

7 [46.059452] fitness = -393079.2

8 [20.066097] fitness = -43261.54

9 [0.4642408] fitness = -10.750779

Найкраще значення покоління -393079.2

Нова популяція

0 [46.059452] fitness = -393079.2

1 [46.059452] fitness = -393079.2

2 [46.059452] fitness = -393079.2

3 [46.059452] fitness = -393079.2

4 [46.059452] fitness = -393079.2

5 [46.059452] fitness = -393079.2

6 [46.059452] fitness = -393079.2

7 [46.059452] fitness = -393079.2

8 [31.07761] fitness = -135592.75

9 [50.99577] fitness = -520333.7

Найкраще значення покоління -520333.7

Нова популяція

0 [50.99577] fitness = -520333.7

1 [46.059452] fitness = -393079.2

2 [46.059452] fitness = -393079.2

3 [46.059452] fitness = -393079.2

4 [46.059452] fitness = -393079.2

5 [50.99577] fitness = -520333.7

6 [50.99577] fitness = -520333.7

7 [50.99577] fitness = -520333.7

8 [51.240444] fitness = -527264.75

9 [25.151857] fitness = -77591.11

Найкраще значення покоління -527264.75

Нова популяція

0 [51.240444] fitness = -527264.75

1 [50.99577] fitness = -520333.7

2 [51.240444] fitness = -527264.75

3 [50.99577] fitness = -520333.7

4 [50.99577] fitness = -520333.7

5 [51.240444] fitness = -527264.75

6 [51.240444] fitness = -527264.75

7 [50.99577] fitness = -520333.7

8 [13.201616] fitness = -15157.732

9 [42.849327] fitness = -322529.72

Найкраще значення покоління -527264.75

Нова популяція

0 [51.240444] fitness = -527264.75

1 [51.240444] fitness = -527264.75

2 [51.240444] fitness = -527264.75

3 [51.240444] fitness = -527264.75

4 [51.240444] fitness = -527264.75

5 [51.240444] fitness = -527264.75

6 [51.240444] fitness = -527264.75

7 [51.240444] fitness = -527264.75

8 [42.84251] fitness = -322389.6

9 [49.401157] fitness = -476634.1

Найкраще значення покоління -527264.75

Нова популяція

0 [51.240444] fitness = -527264.75

1 [51.240444] fitness = -527264.75

2 [51.240444] fitness = -527264.75

3 [51.240444] fitness = -527264.75

4 [51.240444] fitness = -527264.75

5 [51.240444] fitness = -527264.75

6 [51.240444] fitness = -527264.75

7 [51.240444] fitness = -527264.75

8 [22.72272] fitness = -59575.46

9 [30.140541] fitness = -124989.69

Найкраще значення покоління -527264.75

Нова популяція

0 [51.240444] fitness = -527264.75

1 [51.240444] fitness = -527264.75

2 [51.240444] fitness = -527264.75

3 [51.240444] fitness = -527264.75

4 [51.240444] fitness = -527264.75

5 [51.240444] fitness = -527264.75

6 [51.240444] fitness = -527264.75

7 [51.240444] fitness = -527264.75

8 [15.259561] fitness = -21678.203

9 [43.55761] fitness = -337308.28

Найкраще значення покоління -527264.75

Нова популяція

0 [51.240444] fitness = -527264.75

1 [51.240444] fitness = -527264.75

2 [51.240444] fitness = -527264.75

3 [51.240444] fitness = -527264.75

4 [51.240444] fitness = -527264.75

5 [51.240444] fitness = -527264.75

6 [51.240444] fitness = -527264.75

7 [51.240444] fitness = -527264.75

8 [9.839626] fitness = -7455.622

9 [31.458385] fitness = -140064.39

Найкраще значення покоління -527264.75

Нова популяція

0 [51.240444] fitness = -527264.75

1 [51.240444] fitness = -527264.75

2 [51.240444] fitness = -527264.75

3 [51.240444] fitness = -527264.75

4 [51.240444] fitness = -527264.75

5 [51.240444] fitness = -527264.75

6 [51.240444] fitness = -527264.75

7 [51.240444] fitness = -527264.75

8 [26.474083] fitness = -88736.83

9 [46.340263] fitness = -399694.44

Найкраще значення покоління -527264.75

Результат:

[51.240444] fitness = -527264.75

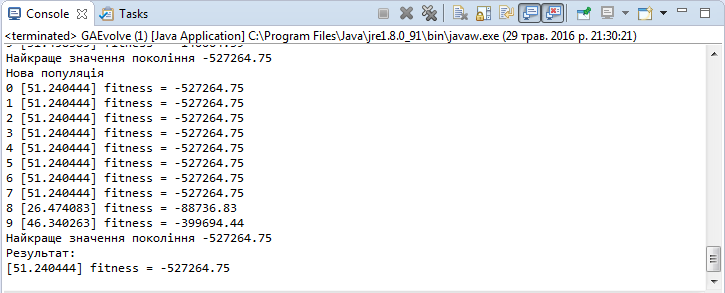


Рис.3. Результати виконання програми для знаходження мінімуму

Результат виконання програми для десяти поколінь з розміром популяції 10 для знаходження максимуму функції:

Початкова популяція

0 [-8.349713] fitness = -1486.615

1 [-4.46373] fitness = -645.333

2 [0.089702606] fitness = 1.1711334

3 [6.649683] fitness = -2991.6206

4 [-1.9606743] fitness = -146.26414

5 [-9.990171] fitness = -1647.6519

6 [-6.3048353] fitness = -1082.901

7 [7.6161366] fitness = -4087.6748

8 [2.303671] fitness = -295.61868

9 [-1.6962986] fitness = -110.11474

Нова популяція

0 [-1.6962986] fitness = -110.11474

1 [-1.6962986] fitness = -110.11474

2 [0.089702606] fitness = 1.1711334

3 [0.089702606] fitness = 1.1711334

4 [0.089702606] fitness = 1.1711334

5 [0.089702606] fitness = 1.1711334

6 [-8.349713] fitness = -1486.615

7 [0.089702606] fitness = 1.1711334

8 [-6.0592794] fitness = -1025.9055

9 [2.6973133] fitness = -412.30792

Найкраще значення покоління 1.1711334

Нова популяція

0 [0.089702606] fitness = 1.1711334

1 [0.089702606] fitness = 1.1711334

2 [0.089702606] fitness = 1.1711334

3 [0.089702606] fitness = 1.1711334

4 [0.089702606] fitness = 1.1711334

5 [0.089702606] fitness = 1.1711334

6 [0.089702606] fitness = 1.1711334

7 [0.089702606] fitness = 1.1711334

8 [3.276785] fitness = -624.5896

9 [5.4396257] fitness = -1898.7739

Найкраще значення покоління 1.1711334

Нова популяція

0 [0.089702606] fitness = 1.1711334

1 [0.089702606] fitness = 1.1711334

2 [0.089702606] fitness = 1.1711334

3 [0.089702606] fitness = 1.1711334

4 [0.089702606] fitness = 1.1711334

5 [0.089702606] fitness = 1.1711334

6 [0.089702606] fitness = 1.1711334

7 [0.089702606] fitness = 1.1711334

8 [-7.75656] fitness = -1386.9337

9 [-8.545308] fitness = -1515.327

Найкраще значення покоління 1.1711334

Нова популяція

0 [0.089702606] fitness = 1.1711334

1 [0.089702606] fitness = 1.1711334

2 [0.089702606] fitness = 1.1711334

3 [0.089702606] fitness = 1.1711334

4 [0.089702606] fitness = 1.1711334

5 [0.089702606] fitness = 1.1711334

6 [0.089702606] fitness = 1.1711334

7 [0.089702606] fitness = 1.1711334

8 [4.183296] fitness = -1061.0376

9 [-7.787261] fitness = -1392.5192

Найкраще значення покоління 1.1711334

Нова популяція

0 [0.089702606] fitness = 1.1711334

1 [0.089702606] fitness = 1.1711334

2 [0.089702606] fitness = 1.1711334

3 [0.089702606] fitness = 1.1711334

4 [0.089702606] fitness = 1.1711334

5 [0.089702606] fitness = 1.1711334

6 [0.089702606] fitness = 1.1711334

7 [0.089702606] fitness = 1.1711334

8 [-0.4002037] fitness = -3.3343494

9 [-9.541287] fitness = -1623.1863

Найкраще значення покоління 1.1711334

Нова популяція

0 [0.089702606] fitness = 1.1711334

1 [0.089702606] fitness = 1.1711334

2 [0.089702606] fitness = 1.1711334

3 [0.089702606] fitness = 1.1711334

4 [0.089702606] fitness = 1.1711334

5 [0.089702606] fitness = 1.1711334

6 [0.089702606] fitness = 1.1711334

7 [0.089702606] fitness = 1.1711334

8 [-2.6627913] fitness = -261.2963

9 [2.1377983] fitness = -252.7979

Найкраще значення покоління 1.1711334

Нова популяція

0 [0.089702606] fitness = 1.1711334

1 [0.089702606] fitness = 1.1711334

2 [0.089702606] fitness = 1.1711334

3 [0.089702606] fitness = 1.1711334

4 [0.089702606] fitness = 1.1711334

5 [0.089702606] fitness = 1.1711334

6 [0.089702606] fitness = 1.1711334

7 [0.089702606] fitness = 1.1711334

8 [1.6222763] fitness = -142.61348

9 [7.5944786] fitness = -4060.809

Найкраще значення покоління 1.1711334

Нова популяція

0 [0.089702606] fitness = 1.1711334

1 [0.089702606] fitness = 1.1711334

2 [0.089702606] fitness = 1.1711334

3 [0.089702606] fitness = 1.1711334

4 [0.089702606] fitness = 1.1711334

5 [0.089702606] fitness = 1.1711334

6 [0.089702606] fitness = 1.1711334

7 [0.089702606] fitness = 1.1711334

8 [1.9826021] fitness = -216.03554

9 [0.9894657] fitness = -51.86851

Найкраще значення покоління 1.1711334

Нова популяція

0 [0.089702606] fitness = 1.1711334

1 [0.089702606] fitness = 1.1711334

2 [0.089702606] fitness = 1.1711334

3 [0.089702606] fitness = 1.1711334

4 [0.089702606] fitness = 1.1711334

5 [0.089702606] fitness = 1.1711334

6 [0.089702606] fitness = 1.1711334

7 [0.089702606] fitness = 1.1711334

8 [-2.27003] fitness = -193.74994

9 [-8.59713] fitness = -1522.5564

Найкраще значення покоління 1.1711334

Результат:

[0.089702606] fitness = 1.1711334

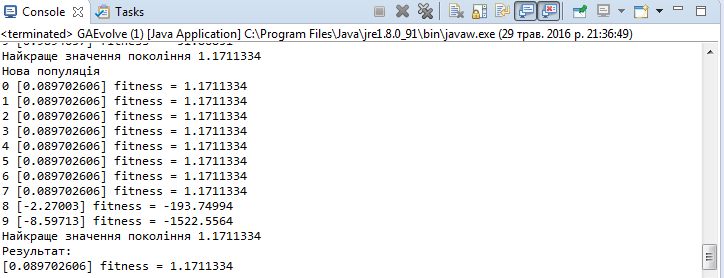


Рис.4. Результати виконання програми для знаходження максимуму

Код програми на мові програмування Java:

**class GAEvolve**

**public** **class** GAEvolve {

**public** **float**[] best\_fitness;// best\_fitness[i] є найкращим значенням для

// і-го покоління даної еволюції

**public** GAIndividual best\_ind; // найкращий індивідум(х) даної еволюції

**public** GAEvolve(**int** generations, **int** pop\_size, **int** genome\_size, **int** xrate,

**int** mrate, **float**[] min\_range, **float**[] max\_range) {

// xrate: частота схрещування

// mrate: частота мутації

// xrate і mrate можуть змінюватися під час покоління генерації

best\_fitness = **new** **float**[generations];

GAPopulation gap = **new** GAPopulation(pop\_size, genome\_size, min\_range,

max\_range);

best\_fitness[0] = gap.ind[gap.best\_index].fitness;

**for** (**int** i = 1; i < generations; i++) {

gap = gap.generate(gap, xrate, mrate, min\_range, max\_range);

best\_fitness[i] = gap.ind[gap.best\_index].fitness;

// if(i<11)

System.***out***

.println("Найкраще значення покоління " + best\_fitness[i]);

}

best\_ind = gap.ind[gap.best\_index];

}

**public** **static** **void** main(String[] args) {

// вибір критерію пошуку мінімуму або максимуму

// для пошуку мінімуму

// float[] min = new float[] { 5f };

// float[] max = new float[] { 15f };

// для пошуку максимуму

**float**[] min = **new** **float**[] { -10f };

**float**[] max = **new** **float**[] { 10 };

GAEvolve gae = **new** GAEvolve(10, 20, 1, 80, 20, min, max);

System.***out***.println("Результат:");

System.***out***.println(gae.best\_ind);

}

}

**class GAIndividual**

**import** java.util.Random;

**public** **class** GAIndividual {

**private** **static** Random *randg* = **new** Random(); // Рандомний генератор

**public** **int** genome\_size;

**public** **float**[] genome;

**public** **float** fitness;

**public** GAIndividual(**int** gsize, **float**[] min\_range, **float**[] max\_range) {

// створити випадкового індивідума довжиною gsize

// і-й ген повине бути в діапазоні між min\_range[i] і max\_range[i]

genome\_size = gsize;

genome = **new** **float**[genome\_size];

**for** (**int** i = 0; i < genome\_size; i++) {

genome[i] = *randg*.nextFloat() \* (max\_range[i] - min\_range[i])

+ min\_range[i];

}

evalFitness();// оцінити придатність цього нового індивідума

}

**public** GAIndividual(**float** d[]) {

// Створити індивідума, що його ген є таки же як d []

genome\_size = d.length;

genome = **new** **float**[genome\_size];

**for** (**int** i = 0; i < genome\_size; i++) {

genome[i] = d[i];

}

evalFitness();// оцінити придатність цієї нового індивідума

}

**public** GAIndividual mutate(**float**[] min\_range, **float**[] max\_range) {

// rate це шанс кожного гена мутувати

**float** rate = 1.0f / (**float**) genome\_size;

**float**[] result = **new** **float**[genome\_size];

**for** (**int** i = 0; i < genome\_size; i++)

result[i] = genome[i];

// застосування точкової мутації

**for** (**int** i = 0; i < genome\_size; i++)

**if** ((**float**) Math.*random*() < rate)

result[i] = *randg*.nextFloat() \* (max\_range[i] - min\_range[i])

+ min\_range[i];

**return** **new** GAIndividual(result);

}

**public** **static** GAIndividual xover1p(GAIndividual f, GAIndividual m) {

// одноточковий кросинговер

Random rng = **new** Random();

**int** xpoint = 1 + rng.nextInt(1);

**float**[] child = **new** **float**[f.genome\_size];

**for** (**int** i = 0; i < xpoint; i++) {

child[i] = f.genome[i];

}

**for** (**int** i = xpoint; i < f.genome\_size; i++) {

child[i] = m.genome[i];

}

**return** **new** GAIndividual(child);

}

**public** String toString() {

String s = "[";

s += genome[0] + "]";

s += " fitness = " + fitness;

**return** s;

}

**private** **void** evalFitness() {

**int** a = 2;

**int** b = -5;

**int** c = -47;

**int** d = -3;

// цільова функція

fitness = (a + b \* genome[0] + c \* genome[0] \* genome[0] + d

\* genome[0] \* genome[0] \* genome[0]);

}

}

**class GAPopulation**

**import** java.util.Random;

**public** **class** GAPopulation {

**private** **static** Random *randg* = **new** Random();// Рандомний генератор

**public** **int** pop\_size;

**public** GAIndividual[] ind;

**public** **int** best\_index; // індекс найкращого індивідума в масиві ind

// best\_fitness = ind[best\_index].fitness

**public** GAPopulation(**int** psize, **int** gsize, **float**[] min\_range,

**float**[] max\_range) {

// створення рандомної популяції кількісью pop\_size

// psize: довжина популяції

// gsize: довжина геному

pop\_size = psize;

ind = **new** GAIndividual[pop\_size];

System.***out***.println("Початкова популяція");

**for** (**int** i = 0; i < pop\_size; i++)

{ind[i] = **new** GAIndividual(gsize, min\_range, max\_range);

System.***out***.println(i+ " "+ ind[i]+" ");}

//System.out.println();

evaluate();

}

**public** GAPopulation(GAIndividual[] p) {

// Створення популяції з такимиж індивідумами як в p

pop\_size = p.length;

ind = **new** GAIndividual[pop\_size];

System.***out***.println("Нова популяція");

**for** (**int** i = 0; i < pop\_size; i++){

ind[i] = p[i];

System.***out***.println(i+ " "+ ind[i]+" ");}

evaluate();

}

**public** GAPopulation generate(GAPopulation p, **int** xrate, **int** mrate,

**float**[] min\_range, **float**[] max\_range) {

//Створення нової популяції з р, xrate відсотків індивідумів нового населення є

//схрещування, mrate відсотків з них створюються в результаті мутації, а інші по відтворення.

**if** (xrate < 0 || xrate > 100 || mrate < 0 || mrate > 100

|| xrate + mrate > 100)

System.***err***.println("error: xrate і/чи mrate неправилно встановлені");

GAIndividual[] newg = **new** GAIndividual[p.pop\_size];

**int** newg\_index = 0;

**int** xn = xrate \* p.pop\_size / 100;

//xn: Кількість нащадків, які будуть схрешення

**int** mn = mrate \* p.pop\_size / 100;

// mn: кількість нащадків які будуть створенні мутацією

// схрещування:

**for** (**int** i = 0; i < xn; i++) {

// select to parents for cross-over:

**int** p1 = p.tr\_select();

**int** p2 = p.tr\_select();

newg[newg\_index++] = GAIndividual.*xover1p*(p.ind[p1], p.ind[p2]);

}

// мутація:

**for** (**int** i = 0; i < mn; i++)

newg[newg\_index++] = p.ind[p.tr\_select()].mutate(min\_range,

max\_range);

// відтворення:

**for** (**int** i = newg\_index; i < p.pop\_size; i++)

newg[i] = p.ind[p.tr\_select()];

**return** **new** GAPopulation(newg);

}

**public** **int** tr\_select() {

//турнірна вибірка розміром pop\_size/10

//вона повертає індекс вибраного особи в ind []

**int** s\_index = *randg*.nextInt(pop\_size);

// індекс вибраного індивідума

**float** s\_fitness = ind[s\_index].fitness;

**int** tr\_size = Math.*min*(10, pop\_size);

**for** (**int** i = 1; i < tr\_size; i++) {

**int** tmp = *randg*.nextInt(pop\_size);

// вибір критерію пошуку мінімуму або максимуму

**if** (ind[tmp].fitness > s\_fitness) {//< для min//>для max

s\_index = tmp;

s\_fitness = ind[tmp].fitness;

}

}

**return** s\_index;

}

**private** **void** evaluate() {

//оцінювання

**int** best = 0;

// індекс найкрощого індивідума

**float** best\_fitness = ind[0].fitness;

// сума придатності особин даної популяції

**for** (**int** i = 1; i < pop\_size; i++) {

// вибір критерію пошуку мінімуму або максимуму

**if** (ind[i].fitness > best\_fitness) {//< для min//>для max

best = i;

best\_fitness = ind[i].fitness;

}

}

best\_index = best;

}

**public** String toString() {

String s = "best individual = " + ind[best\_index];

**return** s;

}

}

Код програми на мові програмування Matlab для знаходження екстремумів функції:

clc

a=2;

b=-5;

c=47;

d=-3;

x=-10:1:53

y=a+b\*x+c\*x.^2+d\*x.^3;

plot(x,y)

grid on

minX=fminbnd(@(x) a+b\*x+c\*x^2+d\*x^3,0,53)

minY=a+b\*minX+c\*minX^2+d\*minX^3

x=-10:1:10

y=a+b\*x+c\*x.^2+d\*x.^3;

figure

plot(x,y)

grid on

maxX=fminbnd(@(x) -(a+b\*x+c\*x^2+d\*x^3),-10,10)

maxY=a+b\*maxX+c\*maxX^2+d\*maxX^3

**Висновки**

Виконавши лабораторну роботу я реалізував за допомогою мови Java програмне забезпечення для пошуку оптимумів функції за допомогою генетичного алгоритму.