Практичне заняття № 5

Тема: Дослідження генетичних алгоритмів на задачі пошуку екстремумів функції за допомогою засобів MATLAB

Мета. Вивчити основні принципи генетичного алгоритму та придбати навички оптимізації функцій за допомогою генетичних алгоритмів засобами MATLAB.

Завдання

1. Створити скріпт-файл оптимізації функції згідно з варіантом (табл. 5.1)
2. Побудувати графік функції.
3. Оптимізувати функцію з використанням функцій MATLAB
4. Оптимізувати функцію з використанням GUI інтерфейсу алгоритму.

Таблиця 8.1 – Функції для оптимізації.

№ з/п	Функція	Інтервал	Знайти
1.	$y(x)=3\sqrt[3]{(x+4)^2}-2x-8$	[– 6; – 2]	максимум
2.	$y=\frac{\sqrt[3]{6(x-3)^2}}{(x-1)^2+8}$	[– 4; 8]	мінімум
3.	$z(x,y)=xe^{(-x^2-y^2)}$	[-2; 2]	мінімум
4.	$y=\frac{\sqrt[3]{6(x-3)^2}}{(x-1)^2+8}$	[– 4; 8]	максимум
5.	$z(x,y)=xe^{(-x^2-y^2)}$	[-2; 2]	максимум

Зміст звіту

1. Титульна сторінка.
2. Мета роботи.
3. Завдання.
4. Скріпт-файл оптимізації функцій.
5. Опис виконання по пунктам завдання (хід роботи) із скріншотами.
6. Висновки.

Контрольні питання

1. Перерахуйте основні особливості ГА.
2. Перелічіть генетичні оператори.
3. Які критерії зупинки використовуються для ГА?
4. Опишіть схему класичного ГА.
5. У чому полягають особливості спільного використання генетичних операторів?

Теоретичні відомості

5.1 Основи генетичних алгоритмів

Генетичні алгоритми (*Genetic Algorithms*) є складовою еволюційних методів, оскільки виникли в результаті спроб копіювання еволюції живих організмів. Генетичні алгоритми (ГА) – це процедури пошуку, засновані на механізмах природного відбору і спадкування; в ГА використовується принцип виживання найбільш пристосованих індивідів. ГА мають певні переваги в порівнянні з традиційними методами оптимізації, оскільки поєднують кращі властивості градієнтних методів та методів евристичного пошуку.

Уявімо штучний світ, населений кількістю N індивідів (особин), причому генетичний код кожного індивіду – це деякий розв'язок нашої задачі. Разом усі індивіди утворюють популяцію P_0. Для простоти вважається, що генетичний код кожного індивіда (генотип) записується у вигляді однієї хромосоми (*Chromosome*) X (для людини генотип містить 46 хромосом). У ГА хромосома X індивіда є впорядкованою послідовністю з M генів, тобто хромосома $X = \{G_1, G_2, ..., G_M\}$ – числовий вектор, який описує розв'язок задачі. Генотип усіх індивідів утворює популяцію хромосом $P = \{X_1, X_2, ... , X_N\}$. Для утворення нових хромосом і пошуку серед них найкращих використовуються оператори (методи): схрещування – об'єднання частин хромосом-батьків; мутації – випадкової зміни окремих генів; селекції – вибору хромосом для наступної ітерації алгоритму.

Основні відмінності ГА від традиційних задач оптимізації такі:

1) виконують пошук розв'язку виходячи не з однієї точки, а з деякої популяції;

2) використовують тільки цільову функцію, а не її похідні або іншу додаткову інформацію;

3) використовують ймовірнісні, а не детерміновані правила відбору.

Терміни ГА.

Популяція – кінцева множина особин.

Індивідіви, що входять в популяцію представляються хромосомами (бітовий рядок) з закодованими у них множинами параметрів задачі, точками в просторі пошуку (search points).

Хромосоми (ланцюжки, або кодові послідовності) – це впорядковані послідовності генів.

Ген (властивість, знак або детектор) – атомарний елемент генотипу, зокрема хромосоми.

Генотип – набір хромосом даного індивіду.

Фенотип – набір значень, що відповідають даному генотипу.

Алель – значення конкретного гена (властивості або варіанта властивості).

Локус – позиція гена в хромосомі (ланцюжку).

Покоління – чергова популяція в генетичному алгоритмі

Функція пристосованості (fitness function) – міра пристосованості даного індивіда в популяції. Задача оптимізації цільової функції зводиться до пошуку найбільш пристосованого індивіда.

Методи селекції хромосом

Метод рулетки.

У *методі рулетки* кожній хромосомі X_i ставиться у відповідність сектор колеса рулетки, величина якого пропорційна до функції пристосованості F даної хромосоми.

Ймовірність селекції хромосоми X_i, де $i = 1...N$, дорівнює:

$$v(x_i) = p_s(x_i)100\%$$

де:

$$p_s(x_i) = \frac{F(x_i)}{\sum_{j=1}^{N} F(x_j)}$$

У результаті селекції формується батьківська популяція з чисельністю N, в яку індивіди з великим значенням $p_s(X_i)$ можуть увійти кілька разів, а з малим значенням $p_s(X_i)$ – рідко.

Рангова селекція.

При *ранговій селекції* індивіди популяції впорядковуються за значенням їх функції пристосованості F (за спаданням), де кожній хромосомі X ставиться у відповідність її номер i у списку (ранг).

Ймовірність вибору хромосоми в такому випадку дорівнює:

$$p_s(x_i) = \begin{cases} 1/\mu, 1 \le i \le \mu \\ 0, \mu < i \le N \end{cases}$$

Турнірна селекція.

При *турнірному відборі* з популяції (N хромосом) випадковим способом вибирається t хромосом, краща з яких (переможець) записується у масив батьківських хромосом. Розмір туру $2 \le t < N$ за звичай дорівнює 2. Відбір повторюється, поки кількість батьківських хромосом не стане дорівнювати N.

Генетичні оператори

Оператор схрещування. На першому етапі схрещування вибираються пари хромосом із батьківської популяції. Операції схрещування полягають в обміні фрагментами ланцюжків між двома батьківськими хромосомами. Далі для кожної пари вибирається позиція гена (локус) у хромосомі, який визначає точку схрещування. Якщо хромосома містить L двійкових чисел, то точка схрещування L_K вибирається випадково з інтервалу [1, L]. У результаті схрещування утворюються два нащадки (рис. 5.1):

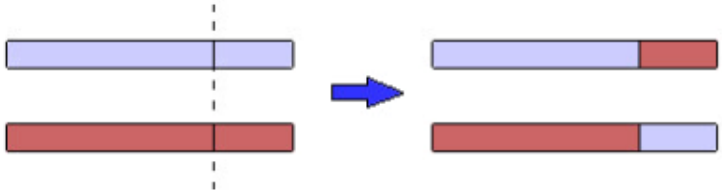

Рисунок 5.1 – Умовна схема схрещування

Оператор мутації з ймовірністю $p_м$ змінює значення гена на певну числову величину (амплітуду мутації A_M). В ГА мутація, імовірність якої звичайно дуже мала ($p_м < 0.1$), може виконуватися на популяції батьків перед схрещуванням або на популяції нащадків.

Класичний генетичний алгоритм

У виді блок-схеми алгоритм наведено на рис. 5.2.

У класичному ГА кожний ген записується у вигляді набору R бітів, тому вся хромосома записується набором двійкових чисел (алелів) довжиною $L = M$ (подібно до молекули ДНК). Кожен ген кодує певну властивість організму.

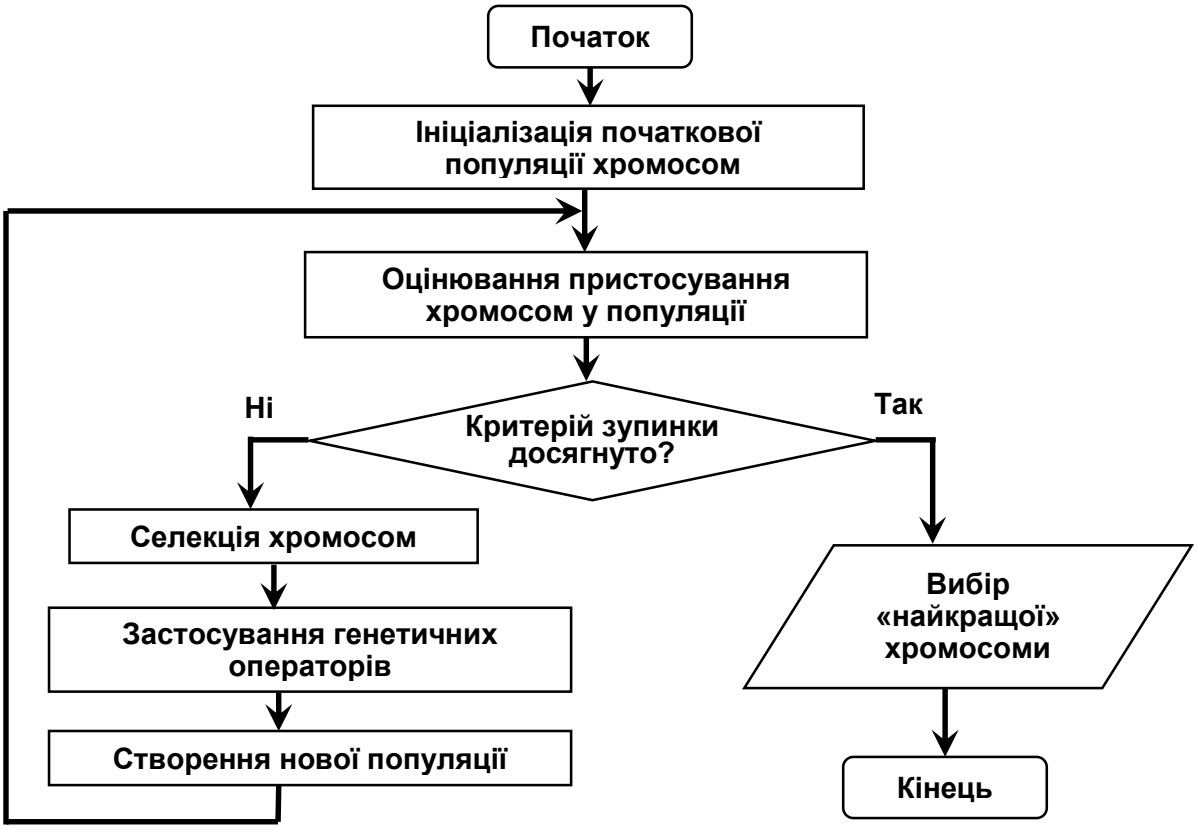

Рисунок 5.2 – Блок-схема класичного генетичного алгоритму

Також основний принцип роботи ГА можна описати за кроками.

1. Генеруємо початкову популяцію з n хромосом;
2. Обчислюємо кожної хромосоми її придатність;
3. Вибираємо пару хромосом-батьків за допомогою одного із способів відбору;
4. Проводимо схрещування (кросинговер) двох батьків із ймовірністю p_c, створюючи двох нащадків;
5. Проводимо мутацію нащадків із ймовірністю p_m;
6. Повторюємо кроки 3-5, доки не буде згенеровано нове покоління популяції, що містить n хромосом;
7. Повторюємо кроки 2-6, доки не буде досягнутий критерій закінчення процесу.

Основними параметрами ГА є:

- ймовірність мутації;
- точність отримання результату;
- кількість ітерацій алгоритму або кількість поколінь;
- обсяг популяції.

5.2 Генетичний алгоритм в MATLAB

Параметри ГА можна задавати командами або через графічний інтерфейс (GUI) генетичного алгоритму.

5.2.1 Приклад мінімізації з використанням команд

Розглянемо основні параметри ГА, які можна встановлювати в командному режимі до запуску алгоритму. При командному способі виклик ГА має такий вид:

```
[x, fval, exitflag, output, population, score] =  ga(fitnessFunction, nvars, options);
```

де: x – знайдене значення аргументів функції, якому відповідає мінімум функції;
fval – значення знайденого мінімуму функції;
exitflag – ознака причини закінчення алгоритму;
output – короткий опис результатів алгоритму;
population – остання популяція хромосом (значення аргументів функції);
score – значення функції пристосованості для хромосом останньої популяції.
ga – ідентифікатор генетичного алгоритму (m-файл в MATLAB);
fitnessFunction – ідентифікатор m-файлу з функцією, що мінімізується;
nvars – кількість аргументів функції;
options – параметри алгоритму, що не задані за замовчуванням.

Основні параметри алгоритму.

Для задання функції, що мінімізується використовується команда:

```
fitnessFunction = @<FaleName>;
```

Приклад: fitnessFunction = @fit_fun,
де fit_fun – ім’я m-файлу з функцією, що мінімізується.

Параметри популяції

'PopulationSize', <кількість індивідів в популяції>

Параметри оператора селекції (відбору)

'SelectionFcn', <ідентифікатор методу селекції>

Ідентифікатори основних методів селекції:

@selectionroulette – рулетка;
@selectiontournament – турнірна селекція
@selectionuniform – батьки вибираються випадковим чином згідно з заданим розподілом та з урахуванням кількості батьківських особин та їх ймовірностей;

Параметри оператора мутації

Цей оператор необхідний для «вибивання» популяції з локального екстремуму та перешкоджає передчасній збіжності. Це досягається за рахунок того, що змінюється випадково вибраний ген у хромосомі.

'MutationFcn', {<ідентифікатор типу мутації>}

Ідентифікатори основних типів мутації:

@mutationgaussian – додає невелике випадкове число (відповідно до розподілу Гауса) до всіх компонентів кожного вектора-індивидіви

@mutationuniform – вибираються випадковим чином компоненти векторів і замість них записуються випадкові числа допустимого діапазону;

@mutationadaptivefeasible генерує набір напрямків залежно від останніх найбільш вдалих і невдалих поколінь і з урахуванням обмежень, що накладаються, просувається вздовж усіх напрямків на різну довжину;

Параметри оператора схрещування (кроссовер)

Кросовер – це операція, коли з двох хромосом породжується одна чи кілька нових хромосом. При цьому хромосоми розрізаються у випадковій точці та обмінюються частинами між собою.

'CrossoverFcn'{<ідентифікатор типу кроссовера>}

Ідентифікатори основних типів кроссовера:

@crossoversinglepoint – одноточковий

@crossovertwopoint – двохточковий

@crossoverarithmetic – арифметичний

@crossoverscattered – генерується випадковий двійковий вектор відповідності батьків

Умови зупинки алгоритму:

Generations – алгоритм зупиняється, коли кількість поколінь (ітерацій) досягає значення Generations.

TimeLimit – алгоритм зупиняється після закінчення певного заданого часу в секундах Time limit.

FitnessLimit – алгоритм зупиняється тоді, коли значення функції пристосованості для найкращої точки поточного популяції буде менше або дорівнювати Fitness limit.

StallGenLimit – алгоритм зупиняється у разі, якщо немає поліпшення для цільової функції у послідовності наступних популяцій один за одним довжиною StallGenLimit.

TolFun – алгоритм зупиняється, якщо середня відносна зміна найкращого значення функції придатності протягом поколінь StallGenLimit менше або дорівнює TolFun. Якщо StallTest є «geometricWeighted», тоді алгоритм зупиняється, якщо середньозважена відносна зміна менше або дорівнює TolFun.

Приклад задання параметрів ГА в скріпт-файлі.

```
  % GA options structure.
fitnessFunction = @ex13;       % Fitness function
nvars = 2;                          % Number of Variables
      % Start with default options
options = gaoptimset;
       % Modify some parameters
options = gaoptimset(options,'PopInitRange',[-4 ; -1 ]);
options = gaoptimset(options,'PopulationSize',10);
options = gaoptimset(options,'Timelimit', 100);
options = gaoptimset(options,'Fitnesslimit', -10000000);
options = gaoptimset(options,'Generations', 100);
options = gaoptimset(options,'CrossoverFcn',{@crossoverarithmetic});
```

```
options = gaoptimset(options,'MutationFcn',{@mutationgaussian 1 1});
options = gaoptimset(options,'Display','off');
options = gaoptimset(options,'SelectionFcn',@selectionroulette);
options = gaoptimset(options,'PlotFcns',{@gaplotbestf @gaplotbestindiv ....
                                @gaplotdistance });
% Run GA
[X,FVAL,REASON,OUTPUT,POPULATION,SCORES]=ga(fitnessFunction,nvars,options);
```

Значення змінних X, FVAL, REASON, OUTPUT, POPULATION, SCORES можна подивитись в робочому полі MATLAB або вивести на екран командою disp().

5.2.2 Приклад мінімізації з використанням GUI-інтерфейсу

Приклад 1. Мінімізувати функцію одного аргументу:

$$f(x) = \sqrt{3 + x^2} + 3 * \cos(x)$$

Створимо m-file для даної функції і збережемо його в робочій директорії MATLAB під ім'ям **fit_fun.m**.

```
function y=fit_fun(x)
y=(3+x.^2.).^(1/2)+3.*cos(x);
end
```

Для початку роботи з GUI інтерфейсом необхідно набрати в командному вікні MATLAB: *optimtool* і натиснути Enter або на головному вікні MATLAB натиснути вкладку APPS і вибрати застосунок Optimization. З'явиться панель роботи оптимізаційного пакету Optimization Tools (рис. 5.3.) до якого входить генетичний алгоритм (ГА).

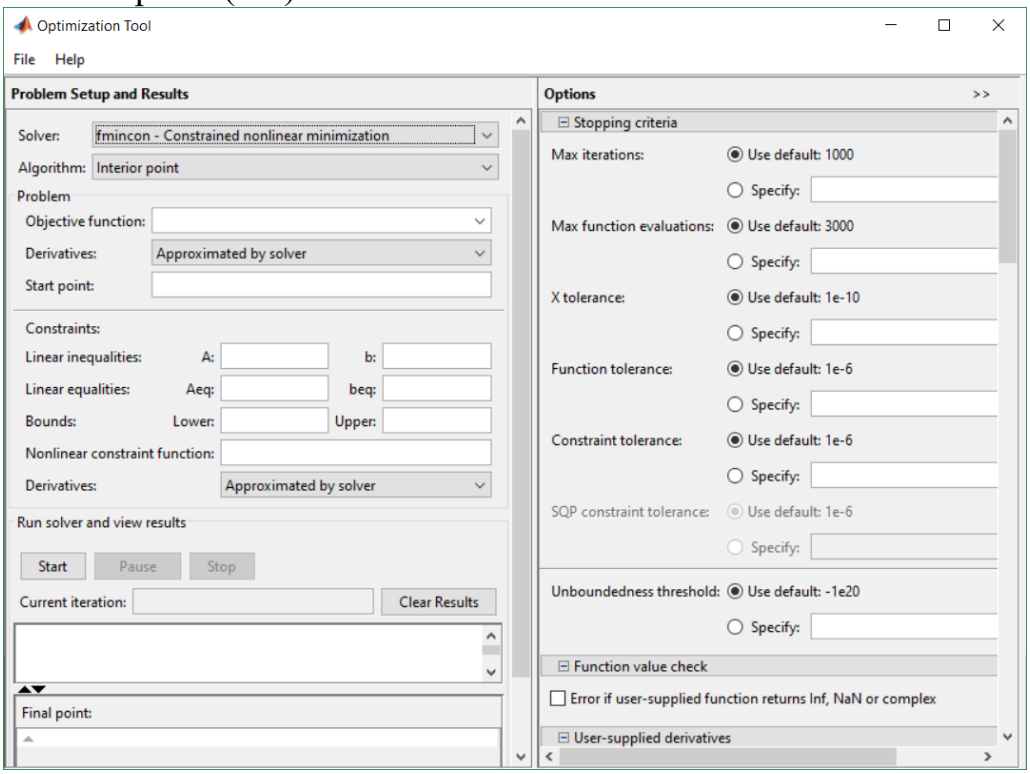

Рис 5.3 – Панель Optimization Tools

В полі Solver обрати ga – Genetic Algorithm (рис. 5.4)

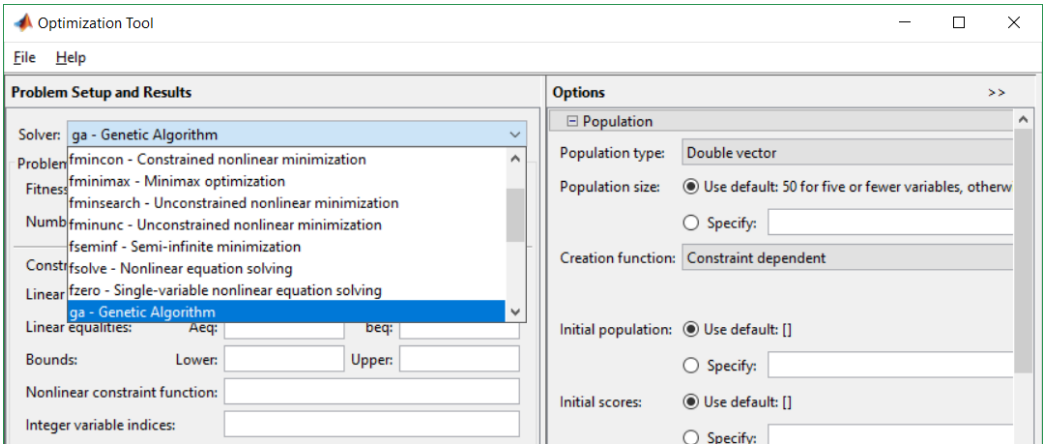

Рис 5.4. – Вибір серед методів оптимізації функції

З’явиться панель генетичного алгоритму (рис. 5.5.)

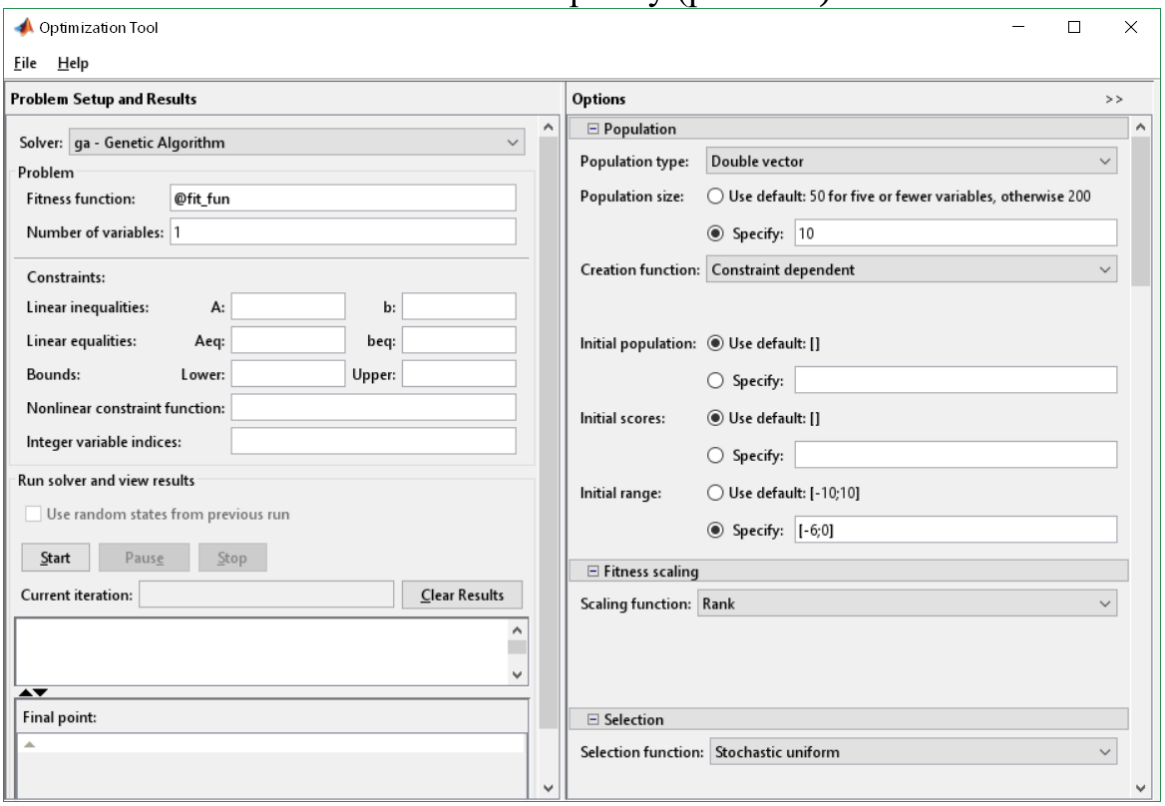

Рисунок 5.5 – Панель генетичного алгоритму

У вікні **Problem Setup and Results** введемо інформацію про функцію для оптимізації

В полі **Fitness function:** введемо ім’я цільової функції @fit_fun

В полі **Number of variables:** визначимо кількість змінних даної функції – 1.

Визначимо параметри ГА (вікно **Options**).

У підвікні **Populations** встановимо значення параметрів ГА:

В полі **Populations size (Specify):** кількість особин в популяції 10,

В полі **Initial range(Specify):** початковий інтервал, на якому буде здійснюватися пошук мінімуму функції = [-6; 0].

В пілвікні **Stopping criteria** (в нижній частині вікна **Options**)

В полі **Generations (Specify):** кількість поколінь = 100.

У підвікні **Plot functions** встановимо прапорці для **best fitness, best individual, distance.**

Клацнемо по кнопці **Start** у вікні **Problem Setup and Results.**

В результаті завершення процесу (рис. 5.6.) у вікні **Final point** з'явиться значення змінної x, що відповідає мінімуму функції, а у вікні **Status and result** можна побачити знайдене мінімальне значення цільової функції.

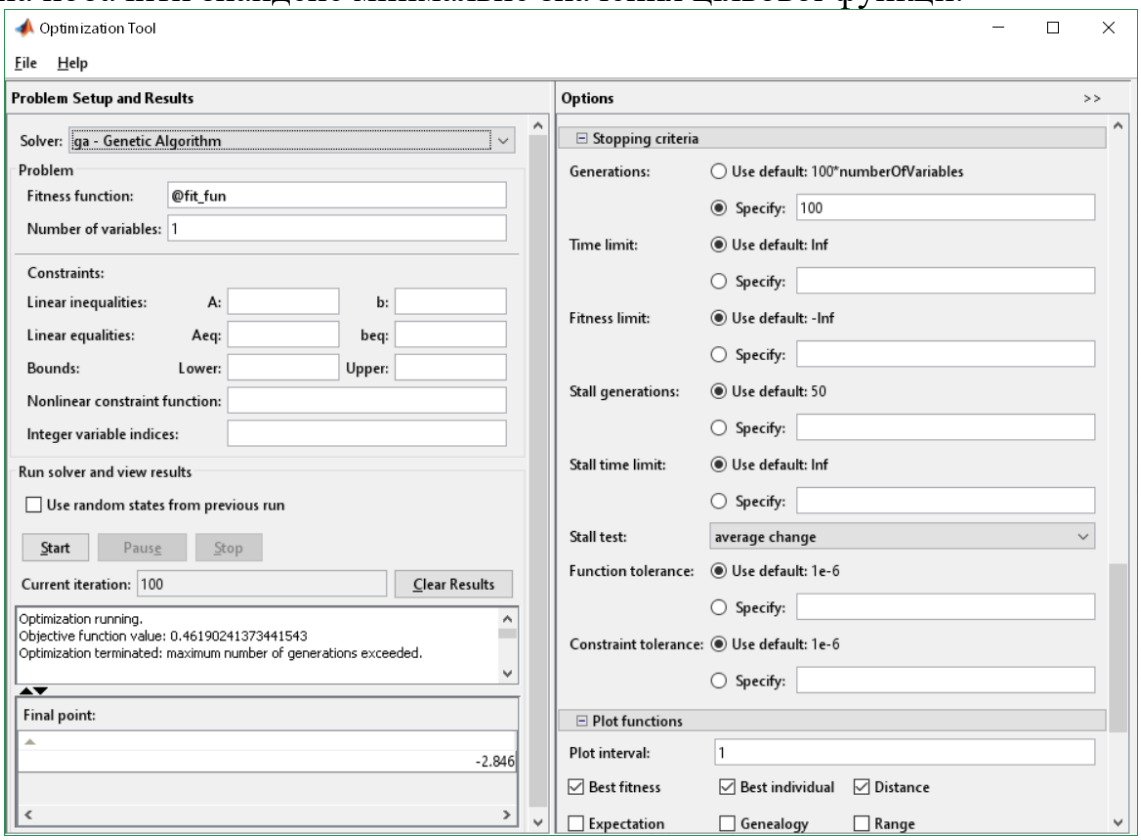

Рисунок 5.6 – Результат ГА для функції $f(x) = \sqrt{3+x^2} + 3*\cos(x)$ (fit_fun).

Для даної задачі результати вийшли наступні мінімум функції досягається в точці x = –2.846 і $f(-2.846) = 0.4619024137344$.

Процес пошуку можна спостерігати на графіках best fitness, best individual, distance (рис.8.7).

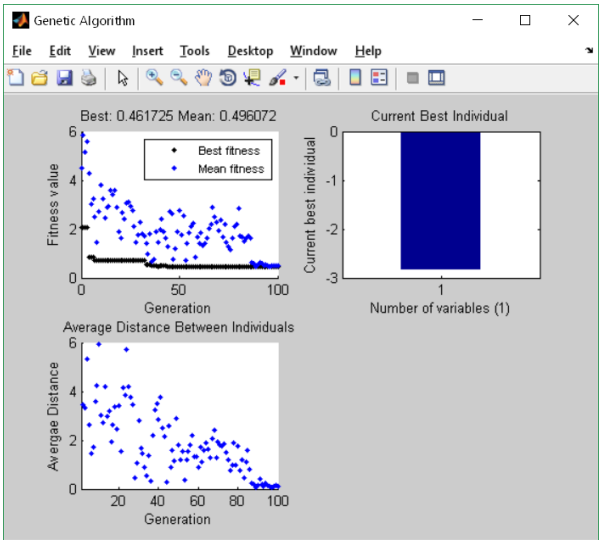

Рисунок 5.7 – Графічний аналіз рішення

Перший малюнок відображає зміну значення цільової функції. Видно, що, починаючи з 90 популяції, алгоритм зійшовся до вирішення. На другому малюнку зображена найкраща особина. Третій малюнок відповідає зміні відстані між особинами в поколіннях. Особи стають однаковими (хеммінгова відстань = 0) в останніх 10 поколіннях. ГА потрібно запустити кілька разів, а потім вибрати оптимальне рішення. Це пов'язано з тим, що початкова популяція формується з використанням генератора випадкових чисел.

Переконатися в правильності рішення можна, побудувавши графік функції (рис. 5.8).

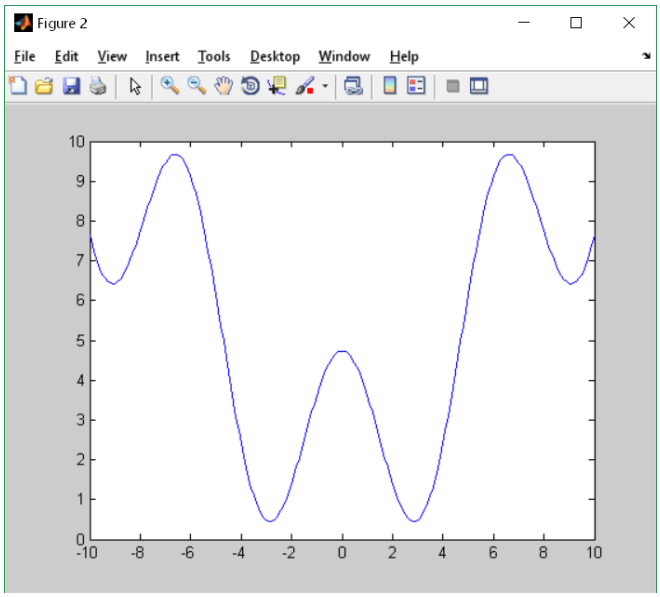

Рисунок 5.8 – Графік функції $f(x) = \sqrt{3+x^2} + 3*\cos(x)$

З графіка видно, що функція має 2 локальних мінімуми, тому при кількох запусках алгоритму можуть бути різні результати. Враховувати це можна задаючи алгоритму різні межі зміни аргументу функції. Для MATLAB версії 2022b при використанні вкладки `APPS` → `Optimization` з'явиться панель (рис. 5.9):

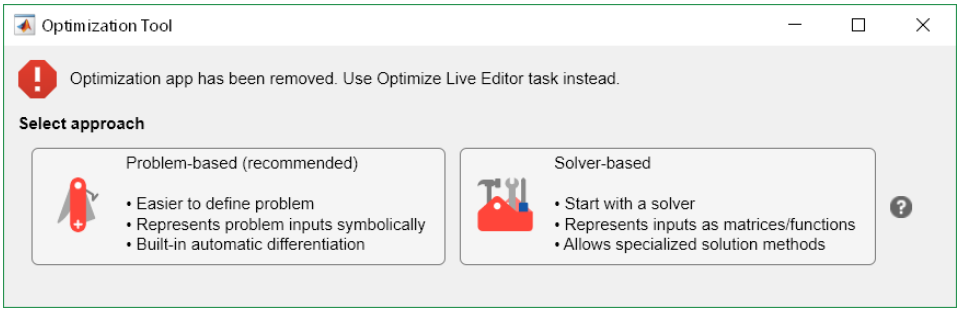

Рисунок 5.9 – Головна панель застосунку `Optimization`

Для роботи з конкретним методом оптимізації треба натиснути кнопку Solver-based (праворуч) для виклику редактора Live Editor (рис. 5.10). Редактор пропонує, як приклад, оптимізацію функції з двома аргументами.

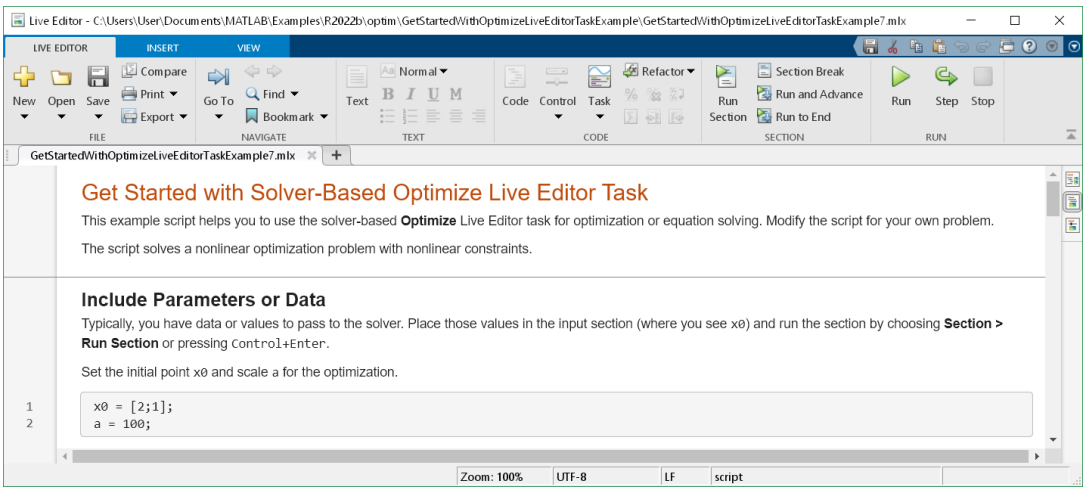

Рисунок 5.10 – Панель редактора Live Editor

Для подальшої роботи в командному вікні якій-небудь змінній присвойте значення 1 для оптимізації функції з одним аргументом або 2 для функції з двома аргументами, наприклад:

>> n=1;

Замініть у прикладі функцію `function f=objectiveeFcn(x,a)` на `function y=fit_fun(x)` або добавте її (рис. 5.11)

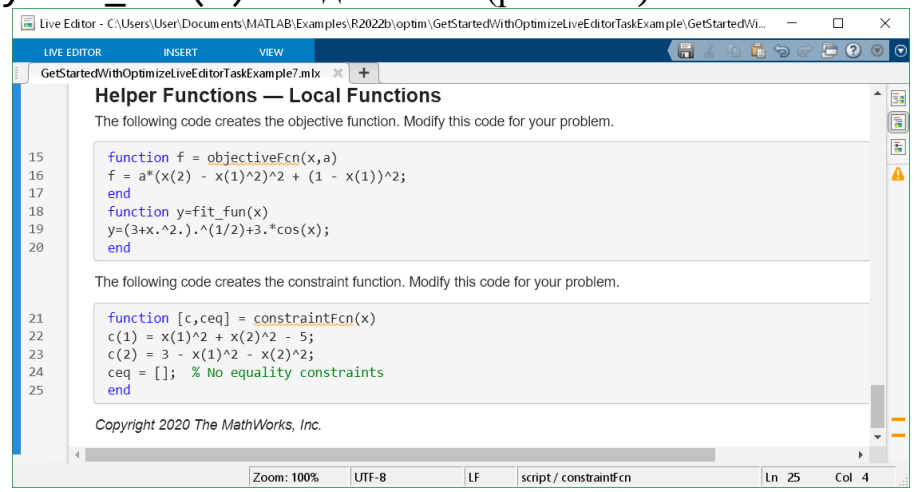

Рисунок 5.11 – Панель редактора Live Editor

Перейдіть до розділу Optimize (рис. 5.12). У полі Specify problem type активуйте, якщо не активовані вікна Nonlinear і Unconstrained, у полі Solver виберіть ga – Genetic algorithm. У полі Objective function активуйте Local function і виберіть функцію fit_fun. У полі Number of variables виберіть n (кількість аргументів функції). У полі Plot увімкніть прапорці Distance, Best individual, Best Function. В меню редактора натисніть Run.

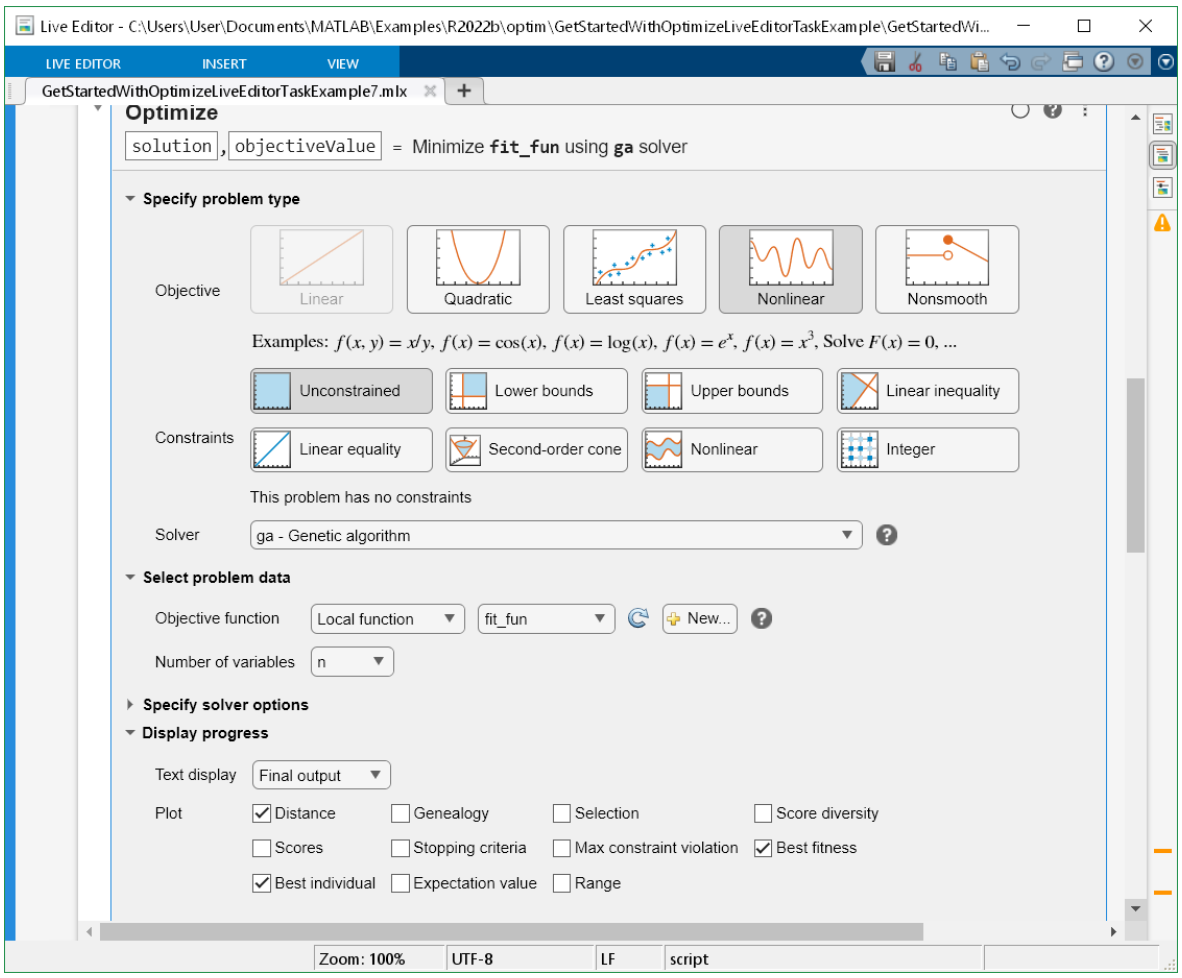

Рисунок 5.12 – Панель редактора Live Editor для оптимізації.

Результат оптимізації можна подивитись в розділі Results.

Приклад 2. Максимізувати функцію двох змінних**:**

$$z(x, y) = \exp(-x^2 - y^2) + \sin(x + y).$$

ГА вирішує тільки завдання мінімізації. Для знаходження максимуму функції *f(x)* слід мінімізувати функцію *–f(x)*.

Створимо M-file для функції *z(x) = –f(x)* і збережемо його в робочій директорії MATLAB під ім’ям ex13.m:

```
function z = ex13(x)
z = – (exp (–x (1).^2–x (2).^2) + sin (x (1) + x (2)));
end;
```

Перейдемо у робоче вікно ГА.

В полі **Fitness function:** введемо ім’я цільової функції @ex13

В полі **Number of variables:** визначимо кількість змінних даної функції – 2.

Визначимо параметри ГА (вікно **Options**).

У підвікні **Populations** встановимо значення параметрів ГА:

В полі **Populations size (Specify):** кількість особин в популяції 10,

В полі **Initial range(Specify):** початковий інтервал, на якому буде здійснюватися пошук мінімуму функції = [–1; 3].

В пілвікні **Stopping criteria** (в нижній частині вікна **Options**)

В полі **Generations (Specify):** кількість поколінь = 100.

У підвікні **Plot functions** встановимо прапорці для **best fitness, best individual, distance.**

Клацнемо по кнопці **Start** у вікні **Problem Setup and Results.**

В результаті завершення процесу (рис. 5.13.)

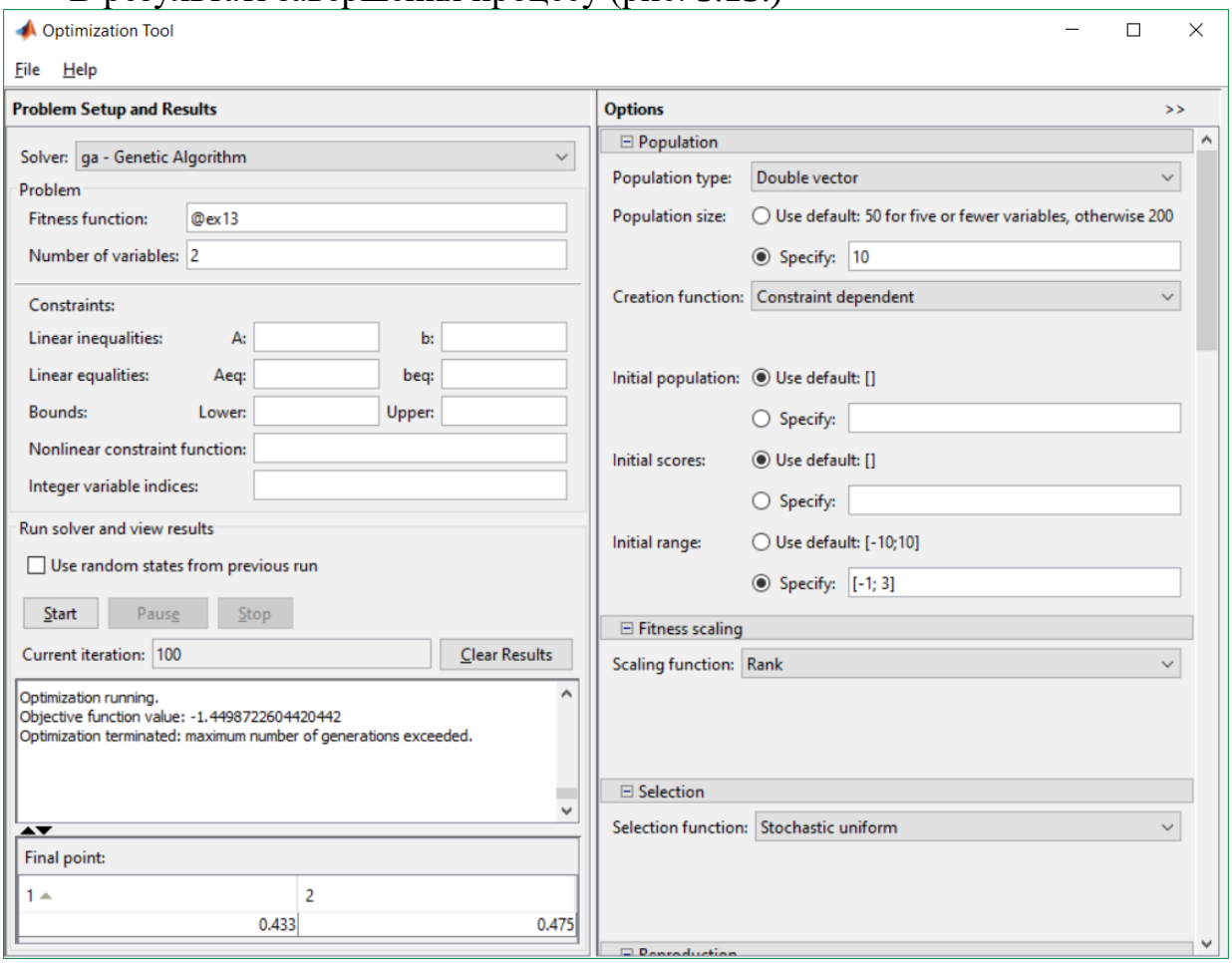

Рисунок 5.13 – Результат ГА для функції $z(x,y) = exp(–x^2 – y^2)+sin\ (x+y)$

У вікні **Final point** з'явиться значення змінних x, у відповідне мінімуму функції z, а у вікні **Status and result** можна побачити знайдене мінімальне значення цільової функції.

Для даної задачі результати вийшли наступні мінімум функції досягається в точці $x = 0.433$, $y=0.475$ і $z(0.433, 0.475) = –1.4498722604420442$.

Для контролю отриманих результатів побудуємо графік функції (рис. 5.14). з графіка можна зробити висновок, що максимум функції приблизно дорівнює 1.1 при x1 = 0.4, x2=0.4. Для побудови графіка скористуємось кодом:

```
[X1,X2]=meshgrid(-1:.1:3,-1:.1:3);
>> Z = (exp(-((X1).*(X1))-((X2).*(X2))+sin(X1+X2)));
%Z=ex13(x);
surf(X1,X2,Z);
```

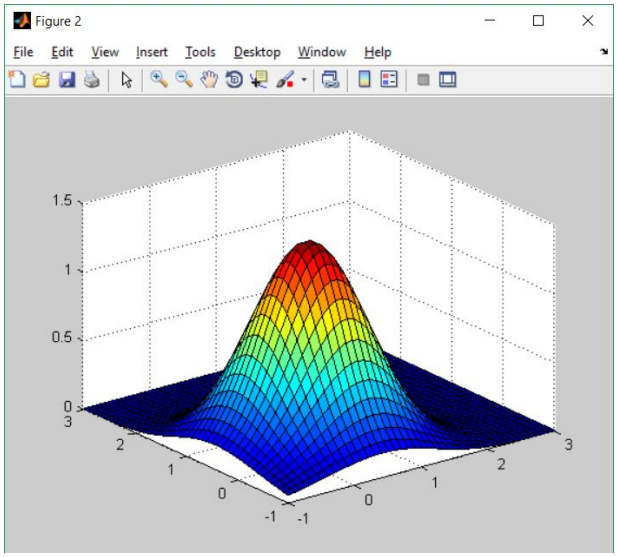

Рисунок 5.14 – Поверхня функції $z(x,y) = exp(–x^2 – y^2)+sin\ (x+y)$

Додаток А

Основні параметри ГА

Формат параметру

options = gaoptimset('param1', value1, 'param2', value2,...)

де: OptionName – ідентифікатор (рядок), значення параметру

Param	опис	приклад
PopInitRange	Інтервал зміни індивідів	options= gaoptimset('PopInitRange',[-4 ; -1]);
PopulationSize	Розмір популяції	options = gaoptimset('PopulationSize',10);
Timelimit'	Завершення алгоритму через заданий час (сек)	options = gaoptimset(options,'Timelimit', 100);
		options = gaoptimset('Fitnesslimit', -10000000);
Generations	Кількість ітерацій алгоритму	options = gaoptimset('Generations', 100)
SelectionFcn	Метод селекції	options = gaoptimset('SelectionFcn',@selectionroulette);

```
fitnessFunction = @ex13;       % Fitness function
nvars = 2;                     % Number of Variables
    % Start with default options
options = gaoptimset;
      % Modify some parameters
options = gaoptimset(options,'PopInitRange',[-4 ; -1 ]);
options = gaoptimset(options,'PopulationSize',10);
options = gaoptimset(options,'Timelimit', 100);
options = gaoptimset(options,'Fitnesslimit', -10000000);
options = gaoptimset(options,'Generations', 100);
options = gaoptimset(options,'CrossoverFcn',{@crossoverarithmetic});
options = gaoptimset(options,'MutationFcn',{@mutationgaussian 1 1});
options = gaoptimset(options,'Display','off');
options = gaoptimset(options,'SelectionFcn',@selectionroulette);
options = gaoptimset(options,'PlotFcns',{@gaplotbestf @gaplotbestindiv ....
                        @gaplotdistance });
% Run GA
[X,FVAL,REASON,OUTPUT,POPULATION,SCORES]=ga(fitnessFunction,nvars,options);
```

Перелік рекомендованої літератури

1. Optimization Toolbox: Solve linear, quadratic, conic, integer, and nonlinear optimization problems [Електронний ресурс] https://www.mathworks.com/products/optimization.htmlp, назва з екрана.

2. Кононюк А.Ю. Нейронні мережі і генетичні алгоритми – К.: «Корнійчук», 2008. – 446 с.