Asignatura: Optimización
Profesor: D. Sc. Gerardo García Gil
2023-A
Ingeniería en Desarrollo de Software
Centro de Enseñanza Técnica Industrial (CETI)

Investigación.

El modelado de ecolocalización para murciélagos se puede resumir de la siguiente manera: cada murciélago virtual vuela aleatoriamente con velocidad vi en la ubicación xi con frecuencia variable o longitud de onda y volumen Ai. A medida que busca y encuentra presas, cambia su ritmo cardíaco, volumen y frecuencia.

La búsqueda se ve reforzada por un paseo aleatorio. La modelización de la ecolocalización de los murciélagos puede ser resumida como sigue: cada murciélago virtual vuela aleatoriamente con la velocidad denominada “vi” en una posición denominada “xi” con la frecuencia variable o también llamado longitud de onda y un volumen Ai. Al mismo tiempo que busca y encuentra su objetivo, cambia de frecuencia, volumen y el nivel de emisión del pulso. El elegir el mejor punto continúa hasta llegar a los criterios de parada.

Para esto se utiliza normalmente alguna técnica de ajuste de frecuencia para controlar el comportamiento de un grupo de murciélagos, y el equilibrio entre exploración y explotación puede ser controlado ajustando los parámetros dependientes en el algoritmo del buscador. La selección del mejor objetivo continúa hasta que se cumplen ciertos criterios de parada que pueden ser definidos por el usuario . Con este fin, la técnica de sintonización de frecuencia se utiliza principalmente para controlar el comportamiento dinámico de los murciélagos, y el equilibrio entre la exploración y la explotación se puede controlar ajustando los parámetros del algoritmo de los murciélagos.

El algoritmo Bat se utiliza en el diseño. Sashikala Mishra, Kailash Shaw y Debahuti Mishra realizaron las clasificaciones de los datos de expresión génica utilizando el modelo BAT-FLANN.Se desarrolló un agregado bat-fuzzy para resolver problemas ergonómicos en el lugar de trabajo. Un enfoque interesante que utilizó el algoritmo “bat” los sistemas difusos mostró una correspondencia confiable entre las predicciones y los datos reales para modelar la energía.

Una comparación detallada del Bat Algorithm (BA) con el Genetic Algorithm (GA), PSO y otros métodos de entrenamiento de redes neuronales feedforward concluyó claramente que BA tiene ventajas sobre otros algoritmos

Existen ciertas variaciones del algoritmo:

- MOBA
- BAT-FLANN
- DABA
- BBA

Desarrollo.

El objetivo de esta práctica es entender el algoritmo “BAT”, editar el código proporcionado por el profesor y añadir una visualización en 3d y en 2d de la función objetivo y de los puntos encontrados así como el punto más óptimo.

El código funciona de la siguiente manera.

Se establecen los parámetros iniciales, como el tamaño de la población (n), el factor de sonorización (A), el factor de derivada del pulso (r), la frecuencia mínima (fmin), la frecuencia máxima (fmax), la tolerancia (tol), el número total de evaluaciones (N_iter) y la dimensión de las variables de búsqueda (d).

Se inicializan los arreglos f y v para almacenar las frecuencias y velocidades de las soluciones.

Se genera una población inicial de soluciones aleatorias y se evalúa la aptitud de cada solución utilizando la función Fun.

Se encuentra la mejor solución actual (best) y se almacena su aptitud mínima (fmin).

Comienza el bucle de iteraciones (k). En cada iteración:

Se actualizan las frecuencias de las soluciones mediante una fórmula basada en la mejor solución actual y las frecuencias mínima y máxima.

Se actualizan las velocidades de las soluciones en función de la diferencia entre cada solución y la mejor solución actual, multiplicada por la frecuencia correspondiente.

Se generan nuevas soluciones (S) sumando las velocidades a las soluciones actuales.

Se verifica si se debe aplicar una tasa de pulso a cada solución. Si se cumple una condición aleatoria (rand > r), se genera una pequeña perturbación en la solución.

Se evalúan las nuevas soluciones y se compara su aptitud con la de las soluciones actuales. Si una nueva solución mejora la aptitud o no supera el factor de sonorización (A), se actualiza la solución actual y su aptitud.

Se actualiza la mejor solución (best) si se encuentra una solución con una aptitud menor a la actual.

Se incrementa el contador de evaluaciones (N_iter).

Se grafica la función de Rosenbrock en 3D y 2D, mostrando los puntos de las soluciones actuales en verde y el mejor punto encontrado en rojo.

Se repiten los pasos segundo y octavo hasta que se alcance la tolerancia especificada.

Al finalizar, se muestra el número total de evaluaciones y se imprime el mejor punto encontrado y su aptitud mínima.

Código:

```
function [best, fmin, N_iter] = para(n, A, r)
% Set default parameters if not provided
if nargin < 1
n = 25;
end
if nargin < 2
A = 0.45;
end
if nargin < 3
r = 0.5;
end
% Frequency range determines the scale
fmin = 0; % Minimum frequency
fmax = 2; % Maximum frequency
% Iteration parameters
tol = 10^(-5); % Stop tolerance
N_iter = 0; % Total number of function evaluations
% Dimension of the search variables
d = 2; % Changed to 2 for Rosenbrock's 2D function
% Initializing arrays
f = zeros(n, 1); % Frequency
v = zeros(n, d); % Velocities
% Initializing the population/solutions
sol = randn(n, d);
Fitness = zeros(n, 1);
% Evaluating fitness of initial solutions
for i = 1:n
Fitness(i) = Fun(sol(i, :));
end
% Finding the best current solution
```

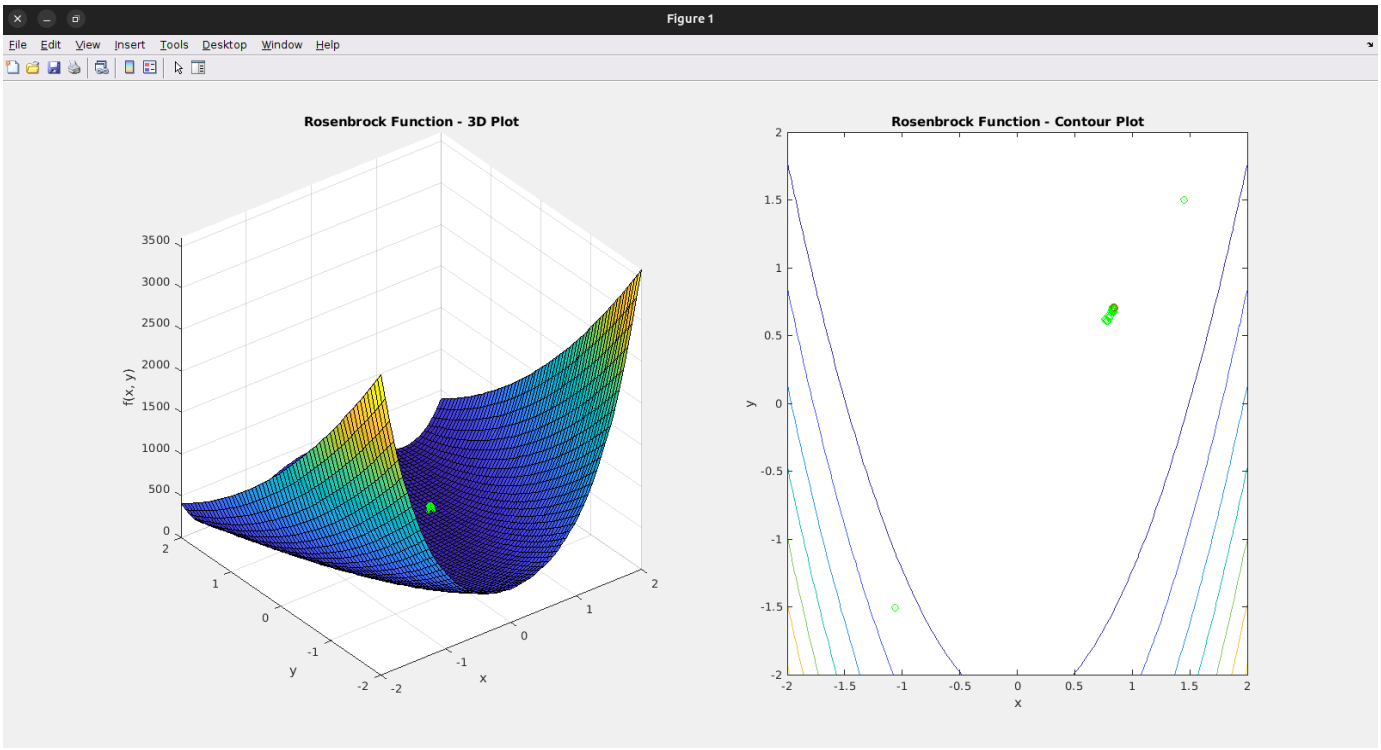
```
[fmin, i] = min(Fitness);
best = sol(i, :);
disp(['Fitness: ', num2str(fmin)]);
% Start the iterations (Bat Algorithm)
while fmin > tol
for i = 1:n
f(i) = fmin + (fmin - fmax) * rand;
v(i, :) = v(i, :) + (sol(i, :) - best) * f(i);
S(i, :) = sol(i, :) + v(i, :);
% Pulse rate
if rand > r
S(i, :) = best + 0.01 * randn(1, d);
end
% Evaluate new solutions
Fnew = Fun(S(i, :));
% If the solution improves or not too loudness
if Fnew <= Fitness(i) && rand < A
sol(i, :) = S(i, :);
Fitness(i) = Fnew;
end
% Update the best solution
if Fnew <= fmin
best = S(i, :);
fmin = Fnew;
end
end
N_iter = N_iter + 1;
disp(['Number of evaluations: ', num2str(N_iter)]);
disp(['Best = ', num2str(best), ' fmin = ', num2str(fmin)]);
% Plotting 3D and 2D graphics
```

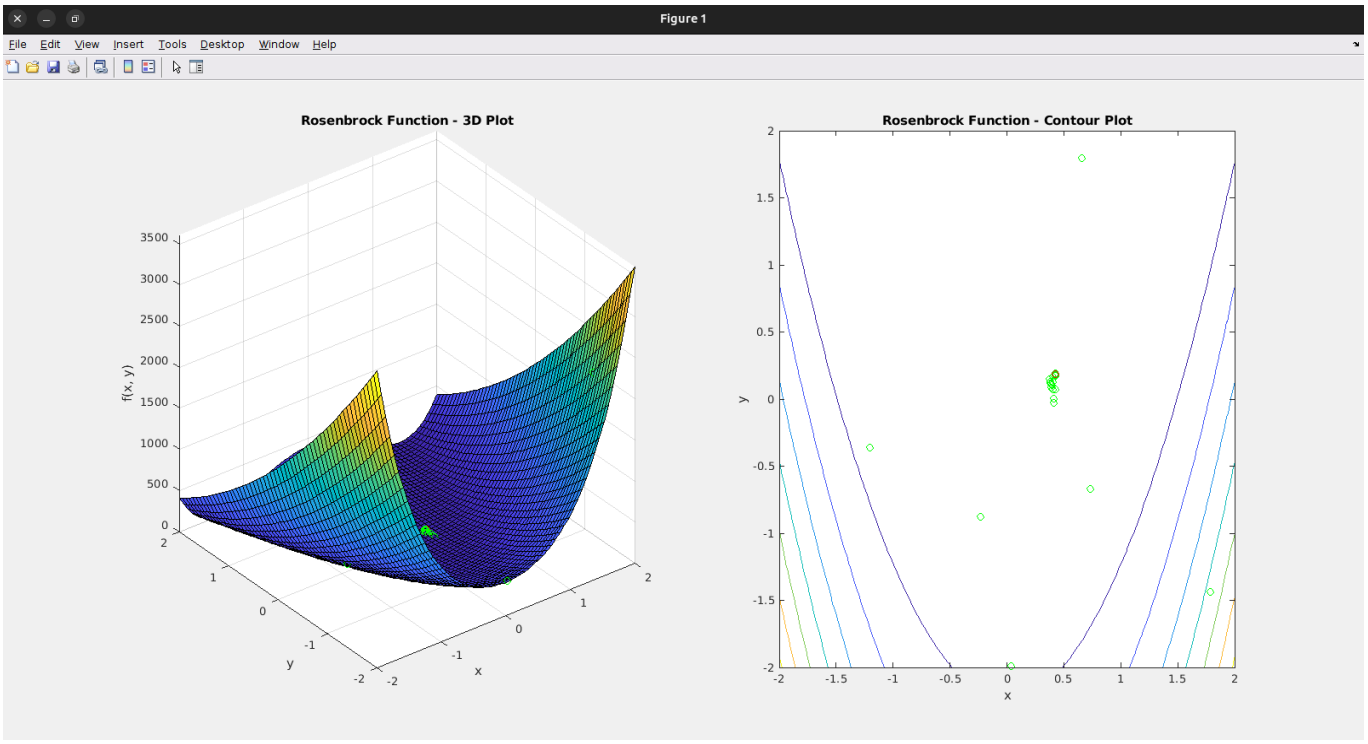
```
figure(1);
subplot(1, 2, 1);
ezsurf(@(u1,u2) (1-u1).^2 + 100*(u2-u1.^2).^2, [-2 2 -2 2]);
hold on;
plot3(sol(:, 1), sol(:, 2), Fitness, 'go');
plot3(best(1), best(2), fmin, 'ro');
hold off;
xlabel('x');
ylabel('y');
zlabel('f(x, y)');
title('Rosenbrock Function - 3D Plot');
subplot(1, 2, 2);
ezcontour(@(u1,u2) (1-u1).^2 + 100*(u2-u1.^2).^2, [-2 2 -2 2]);
hold on;
plot(sol(:, 1), sol(:, 2), 'go');
plot(best(1), best(2), 'ro');
hold off;
xlabel('x');
ylabel('y');
title('Rosenbrock Function - Contour Plot');
pause(0.1); % Pause for a short duration to display the plots
end
% Output/display
disp(['Number of evaluations: ', num2str(N_iter)]);
disp(['Best = ', num2str(best), ' fmin = ', num2str(fmin)]);
% Objective function - Rosenbrock's 2D function
function z = Fun(u)
z = (1 - u(1))^2 + 100 * (u(2) - u(1)^2)^2;
%z=u(1)^2-2*u(1)-11;
% z=(u(3)+2)*u(2)*u(1)^2;
```

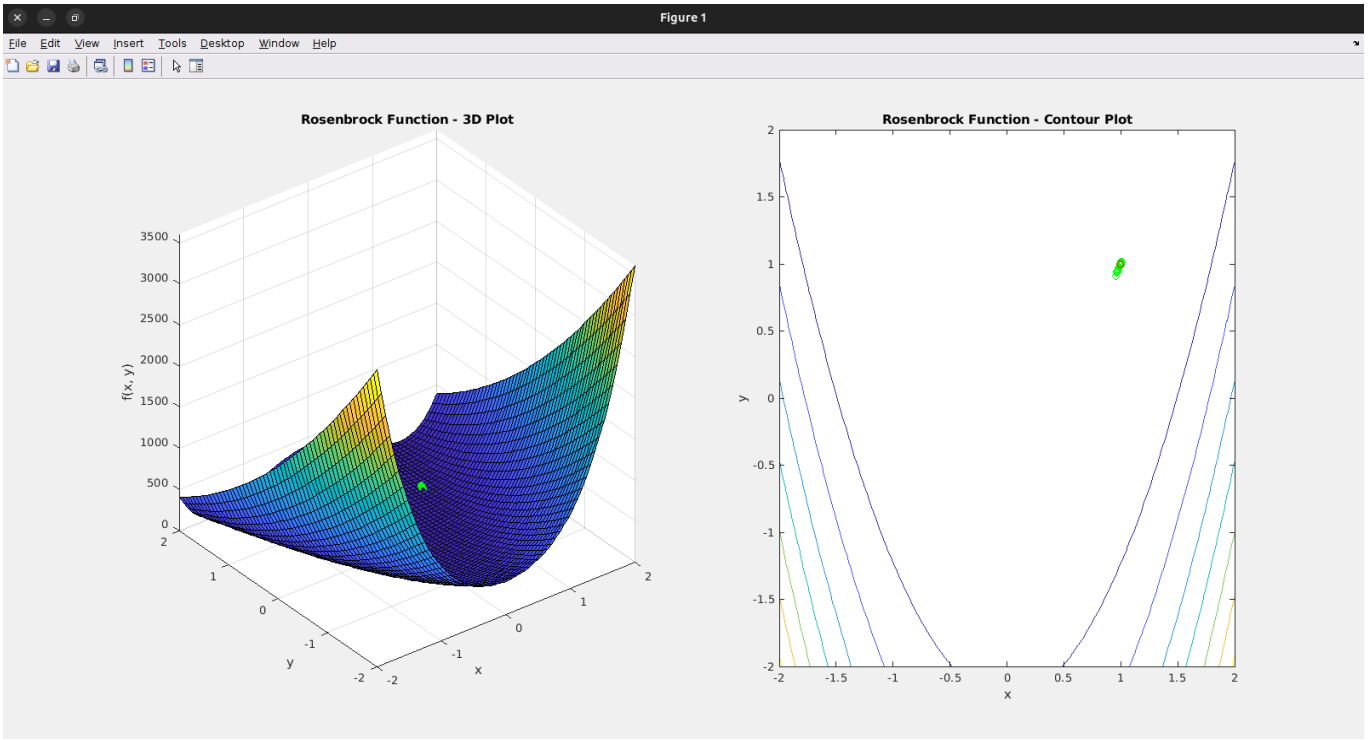
```
%z=0.10471*u(1)^2*u(2)+0.04811*u(3)*u(4)*(14+u(2));

%z=u(1)*exp(-u(1)^2-u(2)^2);

%+(1-u(3))^2;

%ezsurf('(1-u(1))^2+100*(u(2)-u(1)^2)^2+(1-u(3))^2');

end

end
```

Resultados de la práctica.

En las siguientes imágenes podemos observar los puntos verdes simulando los murciélagos que están moviéndose y esparciéndose buscando el punto, se llega un momento en el cual encuentran el punto óptimo y comienzan a concentrarse sobre ese punto haciendo búsquedas más precisas.







Conclusiones.

El objetivo de esta practica se cumplio, logramos añadir código para graficar la funcion objetivo en 3d como en 2d, además logramos dibujar los puntos encontrados por los murciélagos y poder observar cómo se esparcen, y en búsqueda de los puntos más óptimos hasta que comienzan a concentrarse en un punto en específico para dar como resultado los puntos más

óptimos, Fue sencillo realizar esta práctica gracias al código con el que contábamos proporcionado por el profesor. Con esta práctica quedó mucho más claro el funcionamiento del algoritmo y cómo poder aplicarlo.

Referencias.

1. colaboradores de Wikipedia. (2019). Algoritmo de murciélago. Wikipedia, La Enciclopedia Libre. https://es.wikipedia.org/wiki/Algoritmo_de_murci%C3%A9lago

2. Dik, A. (2023). Algoritmos de optimización de la población: Algoritmo de murciélago (Bat algorithm - BA). MQL5 Community. https://www.mql5.com/es/articles/11915

3. Crear una gráfica de líneas en 2D - MATLAB & Simulink - MathWorks España. (n.d.). https://es.mathworks.com/help/matlab/creating_plots/using-high-level-plotting-functions.html