Taller 3 punto 2

2. *Randomly generate an undirected graph of 25 nodes. Set the dimension of the state* $x \in \mathbb{R}^3$. *For each agent create a* 3×3 *matrix* M_i *and vector* b_i *with entries from an interval* $[0.1, 1]$. *Solve the optimization problem:*

$$\min \quad \sum_{i=1}^{m} f_i(x)$$

where

$$f_i(x) = ||M_i x - b_i||^2$$

which is a distributed optimization problem. Solve using Matlab/Python and analyze the convergence from a computational viewpoint (no theoretic). Show a step by step figure of each x_i (only two coordinates are needed).

Optimización Distribuida Basada en Consenso

En este experimento, se implementa un esquema de optimización distribuida basado en consenso en una red de **n = 25** nodos. Cada nodo mantiene un estado en un espacio tridimensional y busca minimizar una función de costo local mediante un algoritmo de actualización iterativa.

Para definir la conectividad entre los nodos, se genera una matriz de adyacencia aleatoria simétrica que representa un grafo no dirigido. Esta matriz es normalizada para garantizar que cada nodo tenga una contribución balanceada en la actualización de su estado.

Cada nodo posee una función de costo local definida por una matriz Mi y un vector bi que se generan aleatoriamente con valores en el intervalo [0.1, 1]. En cada iteración, los nodos actualizan sus estados combinando un promedio ponderado de los estados de sus vecinos y un término de descenso de gradiente basado en la función de costo local.

Tras 200 iteraciones, el estado de cada nodo converge hacia un valor cercano al consenso global, el cual se calcula como el promedio de los estados finales de todos los nodos. Se presentan gráficos que ilustran la evolución de los estados y la convergencia hacia el punto de consenso.

Los resultados muestran que la optimización distribuida basada en consenso es efectiva para minimizar las funciones de costo individuales de cada nodo, permitiendo alcanzar una solución común sin necesidad de una autoridad centralizada.

```
clc; clear; close all;
rng(42); % Para reproducibilidad

% Parámetros
n = 25; % Número de nodos
d = 3;  % Dimensión de x (en R^3)
alpha = 0.01; % Tasa de aprendizaje
max_iter = 200; % Número de iteraciones

% Crear un grafo aleatorio no dirigido (matriz de adyacencia simétrica)
```

```
A = randi([0, 1], n, n)
```

```
A = 25×25
     0     1     1     1     0     0     1     0     1     0     0     1     0 ···
     1     0     1     1     1     1     0     1     0     1     0     1     1
     1     1     1     0     0     1     0     0     0     0     1     1     1
     1     1     1     0     1     0     0     1     1     1     0     1     1
     0     0     1     0     1     1     1     0     1     1     0     1     1
     0     1     1     1     0     0     0     0     0     1     1     1     1
     0     0     0     1     0     0     1     0     0     1     1     0     1
     1     0     0     0     1     0     1     1     1     1     1     0     0
     1     1     0     0     0     0     0     1     0     0     1     0     1
     1     1     0     0     0     1     1     0     0     0     0     1     0
     ⋮
```

```
A = triu(A, 1);
A = A + A'; % Hacerlo simétrico (no dirigido)
A = A - diag(diag(A)); % Asegurar que no haya auto-conexiones

% Normalizar A para que sea doblemente estocástica
A = A ./ (sum(A, 2))
```

```
A = 25×25
         0    0.0714    0.0714    0.0714         0         0    0.0714         0 ···
    0.0625         0    0.0625    0.0625    0.0625    0.0625         0    0.0625
    0.0769    0.0769         0         0         0    0.0769         0         0
    0.0714    0.0714         0         0    0.0714         0         0    0.0714
         0    0.0833         0    0.0833         0    0.0833    0.0833         0
         0    0.0909    0.0909         0    0.0909         0         0         0
    0.0833         0         0         0    0.0833         0         0         0
         0    0.1000         0    0.1000         0         0         0         0
    0.0714         0         0    0.0714    0.0714         0         0    0.0714
         0    0.0667         0    0.0667    0.0667    0.0667    0.0667    0.0667
     ⋮
```

```
%% Inicializar estados x_i en R^3
x = rand(d, n, max_iter); % Estados de los nodos en cada iteración

% Crear matrices M_i y vectores b_i para cada agente
M = zeros(d, d, n); % Matrices M_i
b = zeros(d, n); % Vectores b_i
for i = 1:n
    M(:, :, i) = 0.1 + 0.9 * rand(d, d); % Entradas en [0.1, 1]
    b(:, i) = 0.1 + 0.9 * rand(d, 1); % Entradas en [0.1, 1]
end

% Algoritmo de optimización distribuida
for t = 1:max_iter-1
    for i = 1:n
        % Obtener los vecinos del nodo i
        neighbors = find(A(i, :) > 0);

        % Calcular el promedio ponderado de los estados de los vecinos
        weighted_sum = zeros(d, 1);
```

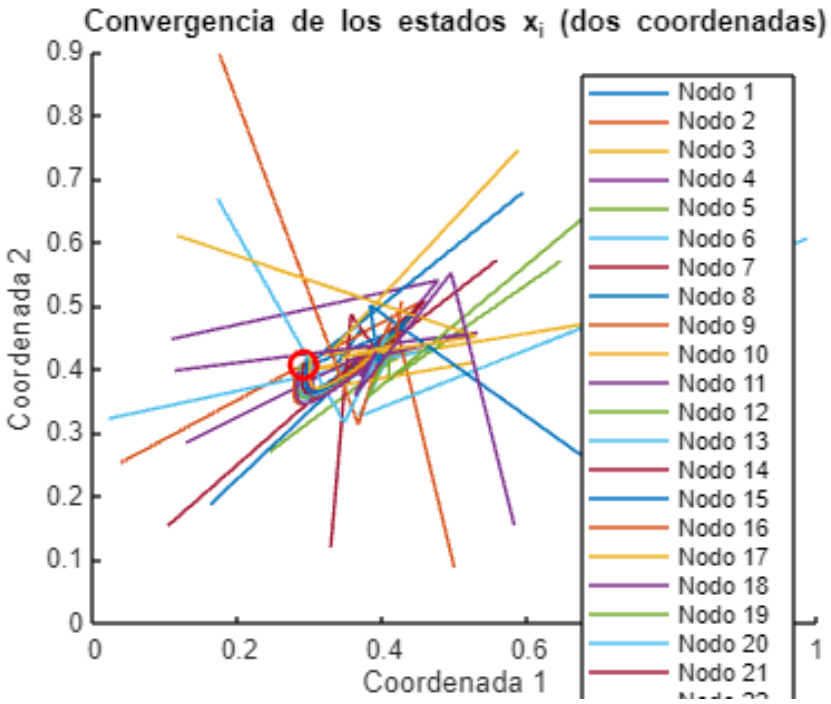
```
        for j = neighbors
            weighted_sum = weighted_sum + A(i, j) * x(:, j, t);
        end

        % Calcular el gradiente de la función de costo local
        grad = 2 * M(:, :, i)' * (M(:, :, i) * x(:, i, t) - b(:, i));

        % Actualizar el estado del nodo i
        x(:, i, t+1) = weighted_sum - alpha * grad;
    end
end

% Calcular el valor de consenso (promedio de los estados finales)
consensus_value = mean(x(:, :, end), 2); % Promedio sobre todos los nodos

% Mostrar el valor de consenso
disp('El valor de consenso es:');
```

```
El valor de consenso es:
```

```
disp(consensus_value);
```

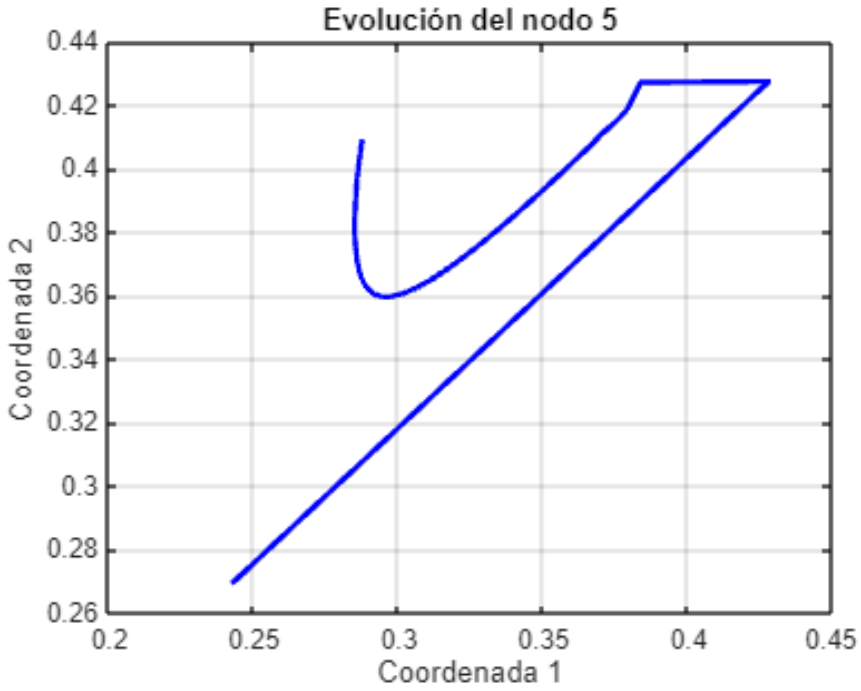
```
    0.2920
    0.4100
    0.2778
```

```
% Graficar la convergencia de los estados (solo dos coordenadas)
figure;
hold on;
for i = 1:n
    plot(squeeze(x(1, i, :)), squeeze(x(2, i, :)), 'DisplayName', ['Nodo '
num2str(i)]);
end

% Dibujar el punto de consenso
plot(consensus_value(1), consensus_value(2), 'ro', 'MarkerSize', 10, 'LineWidth',
2, 'DisplayName', 'Consenso');

xlabel('Coordenada 1');
ylabel('Coordenada 2');
legend show;
title('Convergencia de los estados x_i (dos coordenadas)');
hold off;
```

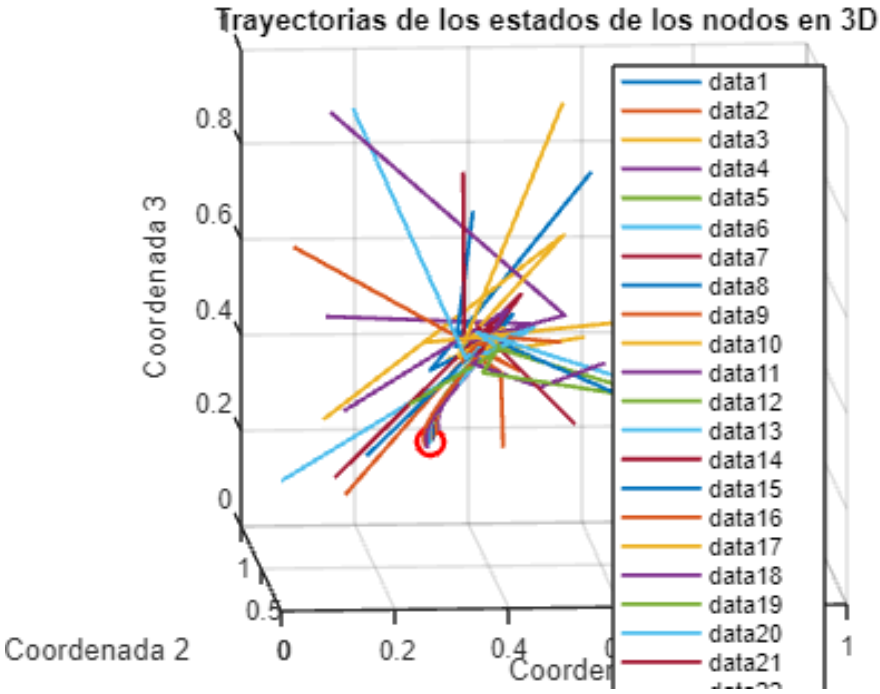


Se gráfica para un solo nodo y ver su trayectoria más facilmente.

```
% Graficar la evolución de un nodo específico (por ejemplo, el nodo 1)
nodo_especifico = 5; % Seleccionar el nodo 1
figure;
plot(squeeze(x(1, nodo_especifico, :)), squeeze(x(2, nodo_especifico, :)), 'b-',
'LineWidth', 2);
xlabel('Coordenada 1');
ylabel('Coordenada 2');
title(['Evolución del nodo ' num2str(nodo_especifico)]);
grid on;
```




```
% Graficar la evolución de los estados de los nodos en 3D

figure;
hold on;
grid on;
xlabel('Coordenada 1');
ylabel('Coordenada 2');
zlabel('Coordenada 3');
title('Trayectorias de los estados de los nodos en 3D');

for i = 1:n
    plot3(squeeze(x(1, i, :)), squeeze(x(2, i, :)), squeeze(x(3, i, :)),
'LineWidth', 1.5);
end

% Dibujar el punto de consenso
plot3(consensus_value(1), consensus_value(2), consensus_value(3), 'ro',
'MarkerSize', 10, 'LineWidth', 2, 'DisplayName', 'Consenso');

legend('show');
view(3); % Establecer la vista en 3D
hold off;
```


Trayectorias de los estados de los nodos en 3D
Coordenada 3
0.8
0.6
0.4
0.2
0
1
0.5
Coordenada 2
0
0.2
0.4
1
data1
data2
data3
data4
data5
data6
data7
data8
data9
data10
data11
data12
data13
data14
data15
data16
data17
data18
data19
data20
data21