Centro de Investigación y de Estudios
Avanzados del Instituto Politécnico Nacional
Unidad Guadalajara

Tarea 6. Control de formaciones basado en desplazamiento

Presentado por

Jesús Alejandro Díaz Hernández

Presentado para el curso de

Tópicos avanzados de control 2

Curso impartido por: Héctor Manuel Becerra Fermín
Profesor

Guadalajara, Jalisco 24 de junio del 2024

Pregunta 1.-

Considerando el grafo de la figura 8.b del artículo "Multi-vehicle consensus with a time varying reference state" pero sin líder

Inciso a)

La matriz Laplaciana correspondiente

$$\begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

Inciso b)

Los valores y vectores propios son:

$$\begin{bmatrix} 1 \\ 3 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & -0.4 & -0.5 & 0 \\ 0 & 0.8 & -0.5 & 0 \\ 0 & -0.4 & -0.5 & 0 \\ 1 & 0 & -0.5 & -1 \end{bmatrix}$$

Inciso c)

Como el vector propio izquierdo asociado al eigenvalor nulo y normalizado es

$$\begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \\ 0 \end{bmatrix}$$

las condiciones iniciales en x elegidas al azar son:

$$\begin{bmatrix} 2 \\ 1 \\ 4 \\ 10 \end{bmatrix}$$

y en y también elegidas al azar son:

$$\begin{bmatrix} 5 \\ 6 \\ 10 \\ 0 \end{bmatrix}$$

Los valores de consenso en x son 2.33 y en y son 7.

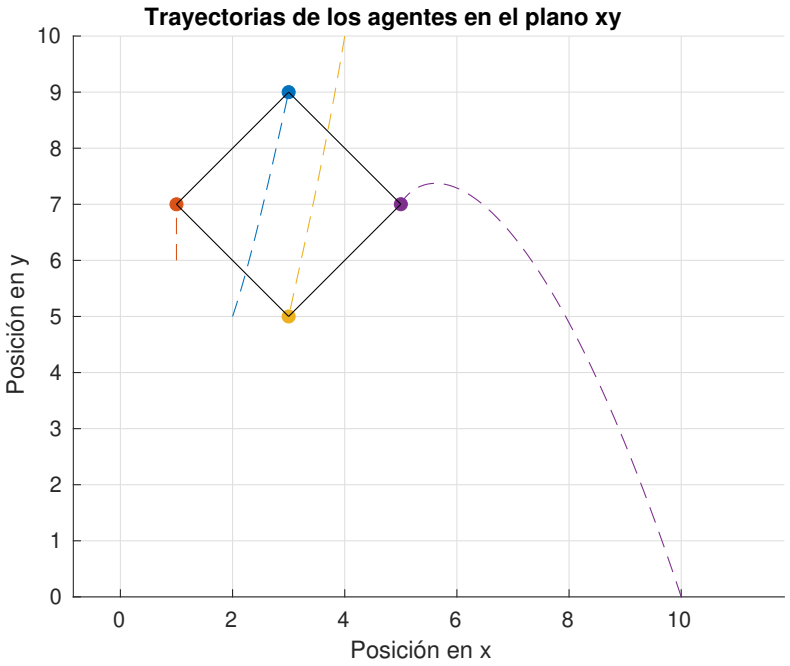
Inciso d)

Considerando los valores del artículo para especificar la formación $\delta_1 = [0, 2]^T$,$\delta_2 = [-2, 0]^T$,$\delta_3 = [0, -2]^T$ y $\delta_4 = [2, 0]^T$ y un control por formación dado como

$$u_i = k_p \sum w_{ij}(p_j - p_i - p_j^* + p_i^*)$$

se obtiene el resultado mostrado en el siguiente inciso

Inciso e)



Inciso f)

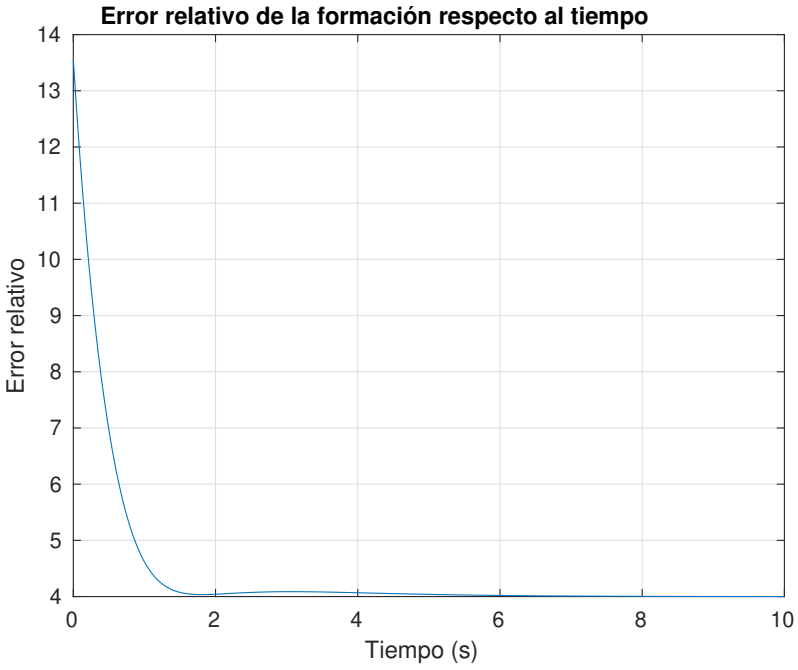
Como el sistema está considerado como

$$(L \bigotimes I)(\xi - \delta)$$

El error de consenso de la formación es igual a la suma del error de consenso del sistema calculado como en el inciso c), más el desplazamiento causado por el eigen vector q multiplicado por la formación δ, tanto en la coordenada x como la coordenada y.

Inciso g)

El error relativo es el mostrado en la siguiente figura:



Pregunta 2.-

Inciso a)

Implementado el siguiente control

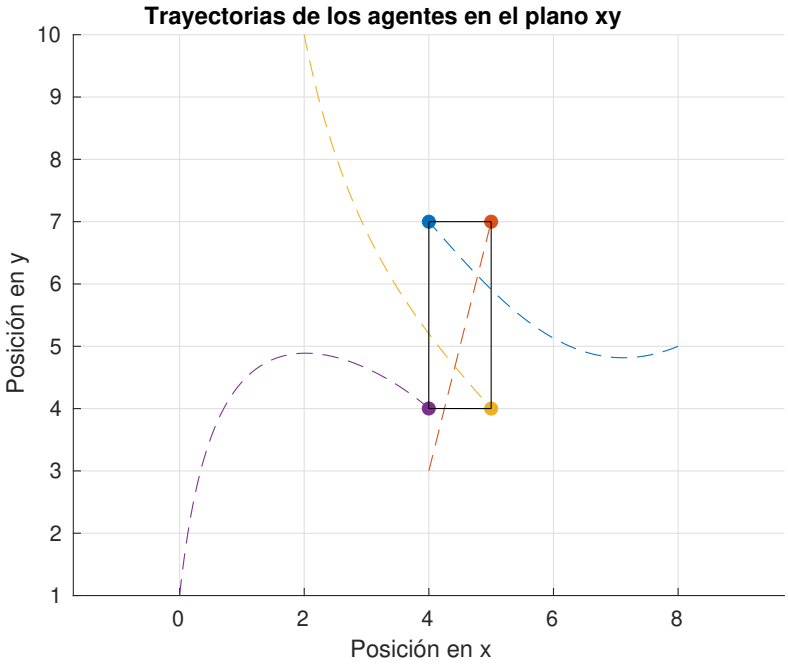
$$u_i = k_p \sum w_{ij}(p_j - p_i - \delta^*_{ji})$$

Donde especificamos las diferencias relativas en δ^*_{ji} para lograr una formación rectangular como:

$$\delta^*_{21} = [1, 0]^T$$
$$\delta^*_{13} = [-1, -3]^T$$
$$\delta^*_{23} = [0, 3]^T$$
$$\delta^*_{34} = [1, 0]^T$$

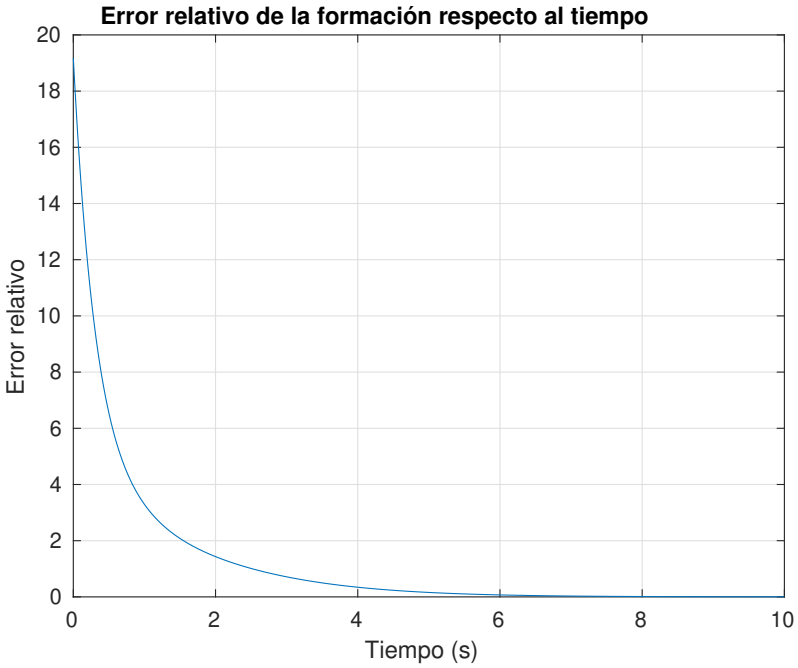
Se consigue la siguiente formación partiendo de las condiciones iniciales escogidas al azar como $x_x = [8, 4, 2, 0]^T$ y $x_y = [5, 3, 10, 1]^T$

Inciso b)



Inciso c)

El error relativo con respecto al tiempo es:



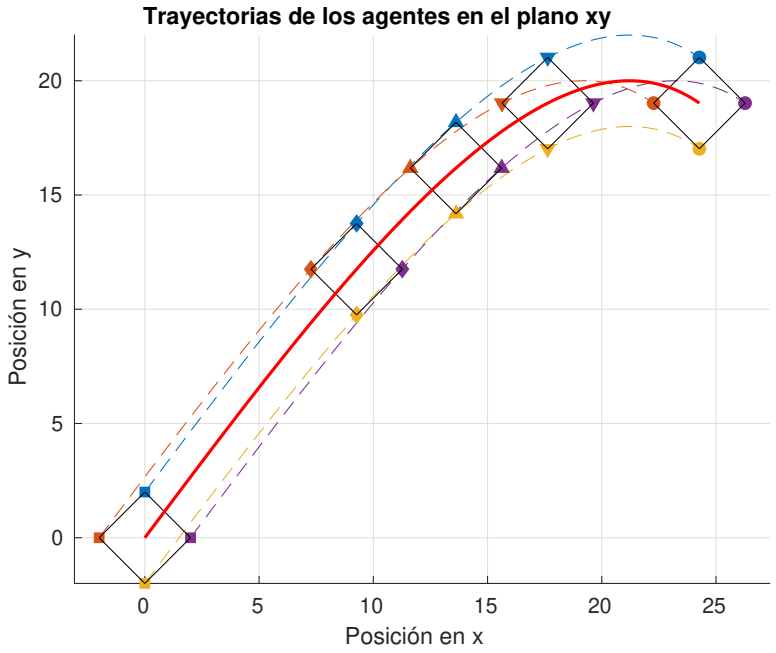
Inciso d)

Definir la formación con desplazamientos relativos, es un poco más complicado debido a que hay considerar la formación y calcular las diferencias manualmente, mientras que con vectores de desplazamiento individuales solo hay que considerar las posiciones que se desean.

Pregunta 3.-

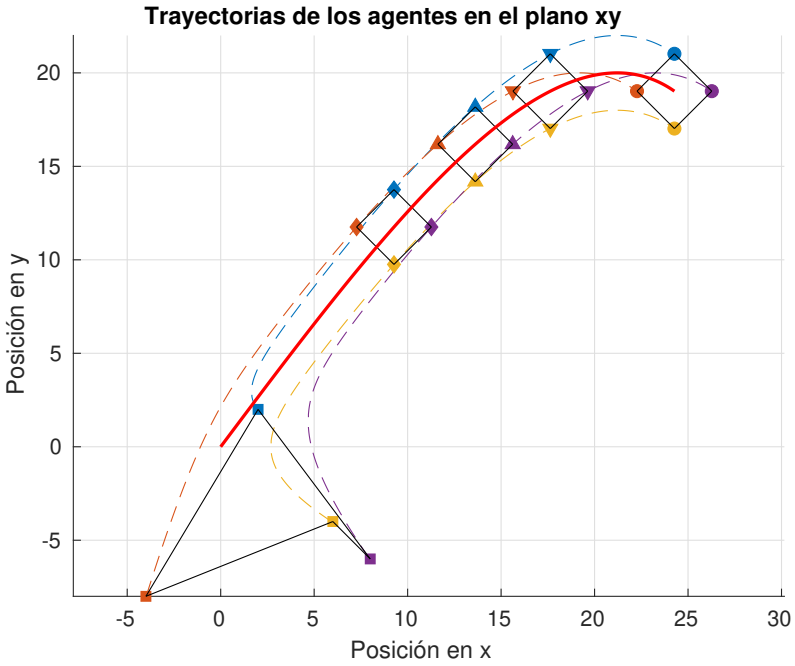
Inciso a)

Reproduciendo la figura 9. considerando el grafo de la figura 8.b, obtenemos:



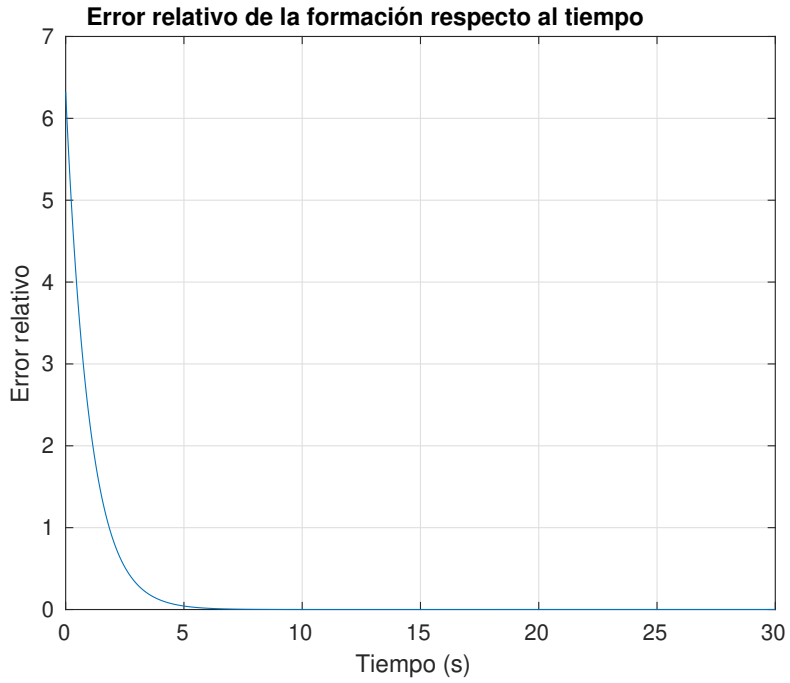
Inciso b)

Ahora consideremos los valores iniciales escogidos aleatoriamente como $x_x = [2, -4, 6, 8]^T$ Y $x_y = [2, -4, -8, -6]^T$ obtenemos los siguientes resultados:



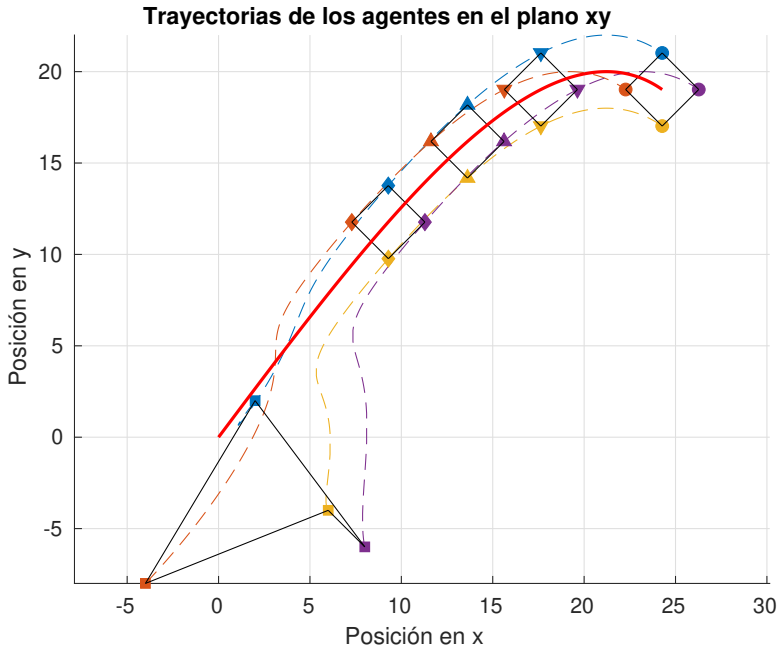
Inciso c)

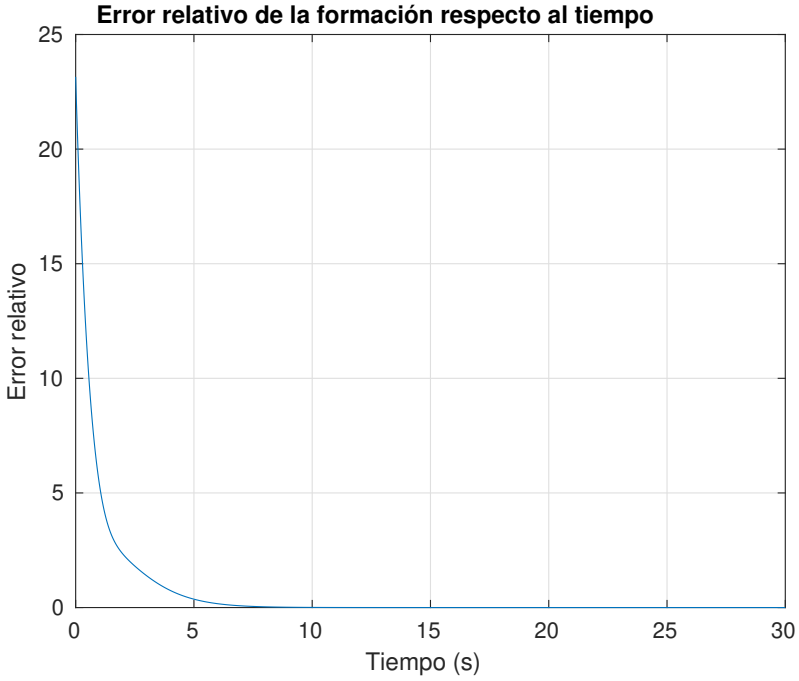
El error relativo es:



Inciso d)

Ahora, considerando el grafo de la figura 8.a y con las mismas condiciones iniciales del inciso anterior, obtenemos los siguientes resultados:


Trayectorias de los agentes en el plano xy
Posición en y
Posición en x
20
15
10
5
0
-5
-5
0
5
10
15
20
25
30


Error relativo de la formación respecto al tiempo
Error relativo
Tiempo (s)
25
20
15
10
5
0
0
5
10
15
20
25
30

Inciso e)

El grafo con menos conexiones es ligeramente más lenta, se esperaría que fuera más pronunciado, pero en realidad no logra ser significativo.

Pregunta 4.-

En conclusión, el control de formación de múltiples agentes utilizando posiciones globales resulta ser más sencillo en comparación con el uso de posiciones relativas. No obstante, la complejidad aumenta significativamente cuando se requiere que los agentes, además de mantener una formación específica, sigan una trayectoria en formación. A pesar de esta mayor dificultad, se logró implementar con éxito esta tarea utilizando la ecuación 11 del artículo de referencia. Cabe destacar que el control propuesto demuestra ser efectivo incluso cuando los agentes no parten de la formación deseada, mostrando robustez y adaptabilidad en diversas condiciones iniciales

Anexo (código usado)

pregunta 1

```
clc
clearvars
close all


q=[1/3 1/3 1/3 0];

L=[1 -1  0 0;
  -1  2 -1 0;
   0 -1  1 0;
   0  0 -1 1];

[V,D]=eig(L);

%La matriz diagonal de unos
I=eye(2);

%Producto Kronecker
KL=kron(L,I);

% Definir las condiciones inciales para las coordenadas por
    separado
x_x = [2; 1; 4; 10]; %Esto es en x
x_y = [5; 6; 10; 0]; %Esto es en y

% Inicializar el vector
x = zeros(2 * length(x_x), 1);

% Combinarlos para la simlacion
for i = 1:length(x_x)
```

```
    x(2*i-1) = x_x(i);
    x(2*i) = x_y(i);
end
x0=x;


%Datos de la simulacion
%periodo de muestreo
Dt=0.01;
tiempo=10; %segundos
iteraciones=tiempo/Dt;

%Simulacion
for k=1:iteraciones

    %Aproximacion de Euler con Laplaciano
    x(:,k+1)=x(:,k)+Dt*(-KL*x(:,k));


end




t=linspace(0,tiempo,iteraciones+1);

%
% Subplot para el consenso en el eje x
subplot(1, 2, 1);
hold on;
for i = 1:2:size(x, 1) % Consenso en x
     plot(t, x(i, :));
end
hold off;
xlabel('Tiempo (s)');
ylabel('\xi_{x}');
title('Consenso en el eje x');
grid on;

% Subplot para el consenso en el eje y
subplot(1, 2, 2);
hold on;
for i = 2:2:size(x, 1) % Consenso en y
    plot(t, x(i, :));
end
hold off;
xlabel('Tiempo (s)');
ylabel('\xi_{y}');
title('Consenso en el eje y');
grid on;


%% Esta area es para el punto d)

% Formacion deseada (especificada en el mapa)
delta1=[0;2];
delta2=[-2;0];
```

```
delta3=[0;-2];
delta4=[2;0];
delta = [delta1;delta2;delta3;delta4]; % deltas concatenadas
P=zeros(2,1);

% Initialize the state matrix to store the simulation results
x_forma = zeros(size(x, 1), iteraciones + 1);
x_forma(:, 1) = x0;


% Inicializar la matriz para almacenar el error relativo
error_relativo = zeros(1, iteraciones + 1);

% Calcular el error inicial
error_relativo(1) = norm(KL * x0 - delta);

% Simulacion
for k = 1:iteraciones
    %Aproximacion de Euler con Laplaciano
    x_forma(:, k+1) = x_forma(:, k) + Dt * (-KL * (x_forma(:, k) -
        delta));

    % Calcular el error en cada iteracion
    error_relativo(k + 1) = norm(KL * x_forma(:, k + 1) - delta);
end

% Tiempo para ploteos
t = linspace(0, tiempo, iteraciones + 1);

% Graficar el error relativo respecto al tiempo
% t = 0:Dt:tiempo;
figure;
plot(t, error_relativo);
xlabel('Tiempo (s)');
ylabel('Error relativo');
title('Error relativo de la formacion respecto al tiempo');
grid on;

% Errores de consenso
figure;
subplot(1, 2, 1);
hold on;
for i = 1:2:size(x_forma, 1) % Error de consenso en x
     plot(t, x_forma(i, :)-delta(i));
end
hold off;
xlabel('Tiempo (s)');
ylabel('e_{x}');
title('Error de consenso en el eje x');
grid on;
subplot(1, 2, 2);
hold on;
for i = 2:2:size(x_forma, 1) % Error de consenso en y
    plot(t, x_forma(i, :)-delta(i));
end
hold off;
xlabel('Tiempo (s)');
```

```
ylabel('e_{y}');
title('Error de consenso en el eje y');
grid on;



% Graficar las posiciones de los agentes en el plano xy con sus
    posiciones finales
figure;
hold on;
num_agents = size(x0, 1) / 2;
colors = lines(num_agents);
for i = 1:num_agents
    plot(x_forma(2*i-1, :) + P(1), x_forma(2*i, :) + P(2), '
        DisplayName', ['Agente ' num2str(i)], 'Color', colors(i, :)
        , 'LineStyle', '--');
    plot(x_forma(2*i-1, end) + P(1), x_forma(2*i, end) + P(2), 'o',
         'Color', colors(i, :), 'MarkerFaceColor', colors(i, :));
end

% Agregar lineas entre los agentes especificados
plot([x_forma(1,end) + P(1) x_forma(7,end) + P(1)], [x_forma(2,end)
     + P(2) x_forma(8,end) + P(2)], 'k-'); % Linea entre Agente 1 y
     Agente 4
plot([x_forma(1,end) + P(1) x_forma(3,end) + P(1)], [x_forma(2,end)
     + P(2) x_forma(4,end) + P(2)], 'k-'); % Linea entre Agente 1 y
     Agente 2
plot([x_forma(7,end) + P(1) x_forma(5,end) + P(1)], [x_forma(8,end)
     + P(2) x_forma(6,end) + P(2)], 'k-'); % Linea entre Agente 4 y
     Agente 3
plot([x_forma(3,end) + P(1) x_forma(5,end) + P(1)], [x_forma(4,end)
     + P(2) x_forma(6,end) + P(2)], 'k-'); % Linea entre Agente 2 y
     Agente 3

hold off;
xlabel('Posicion en x');
ylabel('Posicion en y');
title('Trayectorias de los agentes en el plano xy');
axis equal;
grid on;
```

pregunta 2

```
    clc
clearvars
close all


q=[1/3 1/3 1/3 0];

L=[1 -1  0 0;
  -1  2 -1 0;
   0 -1  1 0;
```

```
   0  0 -1  1];

[V,D]=eig(L);

%La matriz diagonal de unos
I=eye(2);

%Producto Kronecker
KL=kron(L,I);

% Definir las condiciones inciales para las coordenadas por
    separado
x_x = [8; 4; 2; 0]; %Esto es en x
x_y = [5; 3; 10; 1]; %Esto es en y

% Inicializar el vector
x = zeros(2 * length(x_x), 1);

% Combinarlos para la simlacion
for i = 1:length(x_x)
    x(2*i-1) = x_x(i);
    x(2*i) = x_y(i);
end
x0=x;

% Formacion deseada (especificada relativamente)
 delta=[ 1; % d21 x
         0; % d21 y
        -1; % d12 - d32 x
        -3; % d12 - d32 y
         0; % d23 x
         3; % d23 y
         1; % d34 x
         0]; % d34 y
P=zeros(2,1);


%Datos de la simulacion
%periodo de muestreo
Dt=0.01;
tiempo=10; %segundos
iteraciones=tiempo/Dt;

% Inicializar la matriz de estados
x_forma = zeros(size(x, 1), iteraciones + 1);
x_forma(:, 1) = x0;

% Inicializar la matriz para almacenar el error relativo
error_relativo = zeros(1, iteraciones + 1);

% Calcular el error inicial
error_relativo(1) = norm(KL * x0 + delta);

% Simulacion
for k = 1:iteraciones
    %Aproximacion de Euler con Laplaciano
```

```
    x_forma(:, k+1) = x_forma(:, k) - Dt * (KL * x_forma(:, k) +
        delta);
    % Calcular el error en cada iteracion
    error_relativo(k + 1) = norm(KL * x_forma(:, k + 1) + delta);
end

% Graficar el error relativo respecto al tiempo
t = 0:Dt:tiempo;
figure;
plot(t, error_relativo);
xlabel('Tiempo (s)');
ylabel('Error relativo');
title('Error relativo de la formacion respecto al tiempo');
grid on;



% Graficar las posiciones de los agentes en el plano xy con sus
    posiciones finales
figure;
hold on;
num_agents = size(x0, 1) / 2;
colors = lines(num_agents);
for i = 1:num_agents
    plot(x_forma(2*i-1, :) + P(1), x_forma(2*i, :) + P(2), '
        DisplayName', ['Agente ' num2str(i)], 'Color', colors(i, :)
        , 'LineStyle', '--');
    plot(x_forma(2*i-1, end) + P(1), x_forma(2*i, end) + P(2), 'o',
         'Color', colors(i, :), 'MarkerFaceColor', colors(i, :));
end

% Agregar lineas entre los agentes especificados
plot([x_forma(1,end) + P(1) x_forma(7,end) + P(1)], [x_forma(2,end)
     + P(2) x_forma(8,end) + P(2)], 'k-'); % Linea entre Agente 1 y
     Agente 4
plot([x_forma(1,end) + P(1) x_forma(3,end) + P(1)], [x_forma(2,end)
     + P(2) x_forma(4,end) + P(2)], 'k-'); % Linea entre Agente 1 y
     Agente 2
plot([x_forma(7,end) + P(1) x_forma(5,end) + P(1)], [x_forma(8,end)
     + P(2) x_forma(6,end) + P(2)], 'k-'); % Linea entre Agente 4 y
     Agente 3
plot([x_forma(3,end) + P(1) x_forma(5,end) + P(1)], [x_forma(4,end)
     + P(2) x_forma(6,end) + P(2)], 'k-'); % Linea entre Agente 2 y
     Agente 3

hold off;
xlabel('Posicion en x');
ylabel('Posicion en y');
title('Trayectorias de los agentes en el plano xy');
axis equal;
grid on;
```

pregunta3

```
clc
clearvars
close all


%Para primer punto
% A = [ 0  1  0  0
%       1  0  1  0
%       0  1  0  0
%       0  0  1  0 ];
%
% Lg_b = [ 1 -1  0  0
%         -1  2 -1  0
%          0 -1  1  0
%          0  0 -1  1 ];
% Para el segundo punto
A=[0 0 0 0
   1 0 0 0
   0 1 0 0
   0 0 1 0];

Lg_b=[0 0  0 0
     -1 1  0 0
      0 -1 1 0
      0 0 -1 1];

eig(Lg_b)
I = eye(2);
I2 = zeros(8,8);
I2(1,1) = 1;
I2(2,2) = 1;
I3 = eye(8,8);
I3(1,1) = 1/2;
I3(2,2) = 1/2;
I3(3,3) = 1/2;
I3(4,4) = 1/2;

% Ganancias
gamma = 1;
g = 1;
alpha = 1;

% Condiciones iniciales de los agentes en x e y
x_init = [ 2; -4;  6;  8 ];  % Condiciones iniciales en x
y_init = [ 2;  -8; -4;  -6 ];  % Condiciones iniciales en y

% Numero de agentes
num_agents = length(x_init);

% Combinar condiciones iniciales en un solo vector
x0 = zeros(2 * num_agents, 1);
for i = 1:num_agents
    x0(2*i - 1) = x_init(i);
    x0(2*i) = y_init(i);
end

% Tiempo de simulacion
```

```
t0 = 0; % tiempo inicial
tf = 30; % tiempo final
h = 0.001; % paso de tiempo

% Inicializar las variables
t = t0:h:tf;
n = length(t);

% Desplazamientos deseados en x e y
d_x = [ 0; -2;  0;  2 ];  % Desplazamientos deseados en x
d_y = [ 2;  0; -2;  0 ];  % Desplazamientos deseados en y

% Inicializar los desplazamientos deseados y sus derivadas
d = zeros(2 * num_agents, 1);

% Valores iniciales para los desplazamientos deseados
for i = 1:num_agents
    d(2*i - 1) = d_x(i);
    d(2*i) = d_y(i);
end

x = x0;

% Inicializar la matriz para almacenar el error relativo
error_relativo = zeros(1, n+1 );

% Calcular el error inicial
error_relativo(1) = norm(kron(Lg_b, I) * (x0 - d));

% Inicializar a la referencia y su derivada
ref = zeros(8,1);
dref = zeros(8,1);
dref(1) = 30 * (pi/100);
dref(2) = 20 * (pi/50);

x_dot = zeros(8,1);

% Tiempos especificos para almacenar las posiciones
t_intervals = [0, 10, 15, 20];
interval_indices = round(t_intervals / h) + 1; % Convertir tiempos
    a indices

% Inicializar las posiciones en tiempos especificos
positions_at_intervals = cell(length(t_intervals), 1);

for i = 1:n
    % Calculo de la dinamica del sistema con el desplazamiento
        relativo
    x(:, i + 1) = x(:, i) + h * (I3 * alpha * I2 * (dref - gamma *
        (x(:, i) - d - ref)) + g * I3 * (kron(A, I) * x_dot - gamma
         * (kron(Lg_b, I) * (x(:, i) - d))));
    x_dot = (I3 * alpha * I2 * (dref - gamma * (x(:, i) - d - ref))
         + g * I3 * (kron(A, I) * x_dot - gamma * (kron(Lg_b, I) *
        (x(:, i) - d))));

    % Calcular el error en cada iteracion
```

```
    error_relativo(i + 1) = norm(kron(Lg_b, I) * (x(:, i + 1) - d))
        ;

    % Dinamica de la referencia
    ref(1) = 30 * sin(pi * t(i) / 100);
    ref(2) = 20 * sin(pi * t(i) / 50);

    % Guardar la referencia en la trayectoria
    ref_traj(1, i+1) = ref(1);
    ref_traj(2, i+1) = ref(2);

    % Derivadas de la referencia
    dref(1) = 30 * (pi / 100) * cos(pi * t(i) / 100);
    dref(2) = 20 * (pi / 50) * cos(pi * t(i) / 50);

    % Almacenar posiciones en tiempos especificos
    if ismember(i, interval_indices)
        positions_at_intervals{find(interval_indices == i)} = x(:,
            i);
    end
end

% Graficar el error relativo respecto al tiempo
t = linspace(t0, tf, n+1);
figure;
plot(t, error_relativo);
xlabel('Tiempo (s)');
ylabel('Error relativo');
title('Error relativo de la formacion respecto al tiempo');
grid on;

% Graficar las posiciones de los agentes en el plano xy con sus
    posiciones finales
figure;
hold on;
num_agents = size(x0, 1) / 2;
colors = lines(num_agents);

% Graficar trayectorias
for i = 1:num_agents
    plot(x(2*i-1, :), x(2*i, :), 'DisplayName', ['Agente ' num2str(
        i)], 'Color', colors(i, :), 'LineStyle', '--');
    plot(x(2*i-1, end), x(2*i, end), 'o', 'Color', colors(i, :), '
        MarkerFaceColor', colors(i, :));
end

% Graficar las posiciones a los tiempos especificos
markers = {'s', 'd', '^', 'v'}; % Marcadores para tiempos
    especificos
for j = 1:length(t_intervals)
    pos = positions_at_intervals{j};
    for i = 1:num_agents
        plot(pos(2*i-1), pos(2*i), markers{j}, 'Color', colors(i,
            :), 'MarkerFaceColor', colors(i, :), 'DisplayName', ['
            Agente ' num2str(i) ' @ ' num2str(t_intervals(j)) 's'])
            ;
    end
```

```
156
157     % Agregar lineas entre los agentes en tiempos especificos
158     plot([pos(1) pos(7)], [pos(2) pos(8)], 'k-'); % Linea entre
            Agente 1 y Agente 4
159     plot([pos(1) pos(3)], [pos(2) pos(4)], 'k-'); % Linea entre
            Agente 1 y Agente 2
160     plot([pos(7) pos(5)], [pos(8) pos(6)], 'k-'); % Linea entre
            Agente 4 y Agente 3
161     plot([pos(3) pos(5)], [pos(4) pos(6)], 'k-'); % Linea entre
            Agente 2 y Agente 3
162 end
163
164 % Agregar lineas entre los agentes especificados en la posicion
      final
165 plot([x(1,end) x(7,end)], [x(2,end) x(8,end)], 'k-'); % Linea entre
       Agente 1 y Agente 4
166 plot([x(1,end) x(3,end)], [x(2,end) x(4,end)], 'k-'); % Linea entre
       Agente 1 y Agente 2
167 plot([x(7,end) x(5,end)], [x(8,end) x(6,end)], 'k-'); % Linea entre
       Agente 4 y Agente 3
168 plot([x(3,end) x(5,end)], [x(4,end) x(6,end)], 'k-'); % Linea entre
       Agente 2 y Agente 3
169
170 % Graficar la trayectoria de la referencia
171 plot(ref_traj(1, :), ref_traj(2, :), 'r-', 'DisplayName', '
      Referencia', 'LineWidth', 1.5);
172
173 hold off;
174 xlabel('Posicion en x');
175 ylabel('Posicion en y');
176 title('Trayectorias de los agentes en el plano xy');
177 legend show;
178 axis equal;
179 grid on;
```