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

# Tarea 10. Control de formación basado en ángulos (bearings)

Presentado por

**Jesús Alejandro Díaz Hernández**

Presentado para el curso de

Tópicos avanzados de control 2

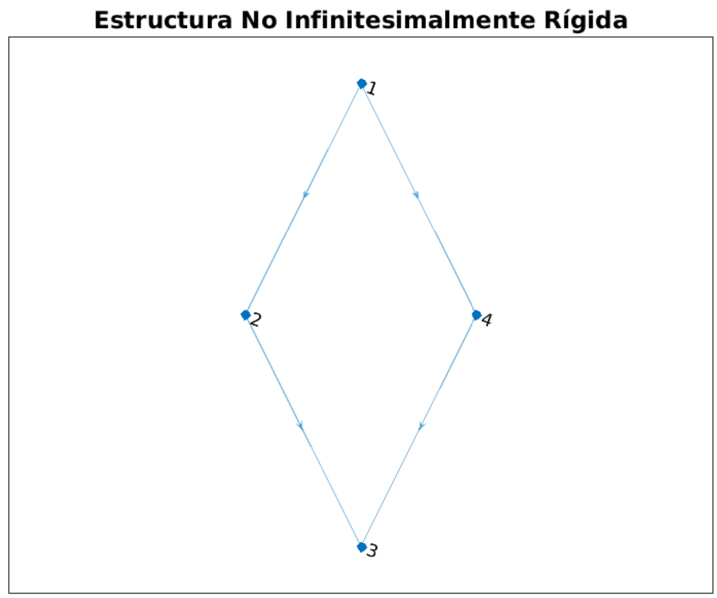
Curso impartido por: Héctor Manuel Becerra Fermín
Profesor

Guadalajara, Jalisco

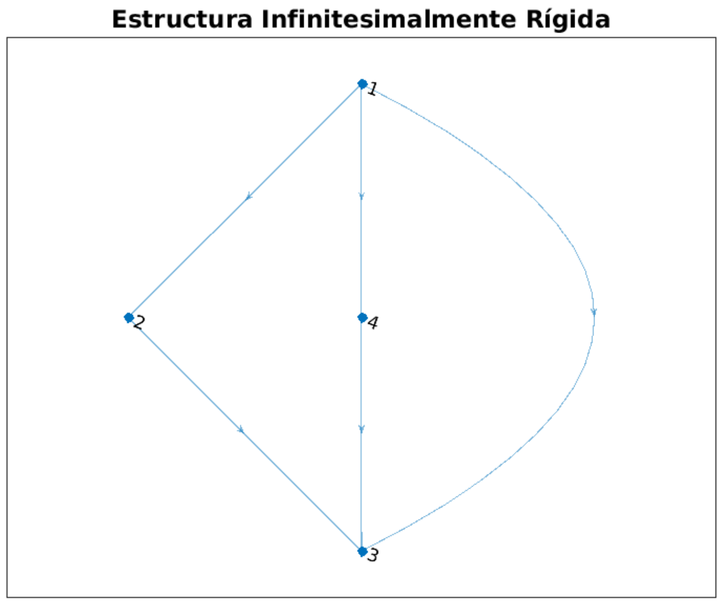
5 de agosto del 2024

## Pregunta 1.-

Por simplicidad usaremos los mismos grafos que en la tera anterior que son el grafo no infinitesimalmente rígido:




El infinitesimalmente rígido




# Pregunta 2.-

Las matrices laplacianas, pero de ángulos serían dadas como:

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

y para la infinitesimalmente rígida:

$$\begin{bmatrix} 1.5 & 0.5 & 0 & 0 & -0.5 & -0.5 & -1 & 0 \\ -0.5 & 1.5 & 0 & -1 & -0.5 & -0.5 & 0 & 0 \\ 0 & 0 & 1.5 & -0.5 & -1 & 0 & -0.5 & 0.5 \\ 0 & -1 & -0.5 & 1.5 & 0 & 0 & 0.5 & -0.5 \\ -0.5 & -0.5 & -1 & 0 & 1.5 & 0.5 & 0 & 0 \\ -0.5 & -0.5 & 0 & 0 & 0.5 & 1.5 & 0 & -1 \\ 1 & 0 & -0.5 & 0-5 & 0 & 0 & 1.5 & -0.5 \\ 0 & 0 & 0.5 & -0.5 & 0 & -1 & -0.5 & 1.5 \end{bmatrix}$$

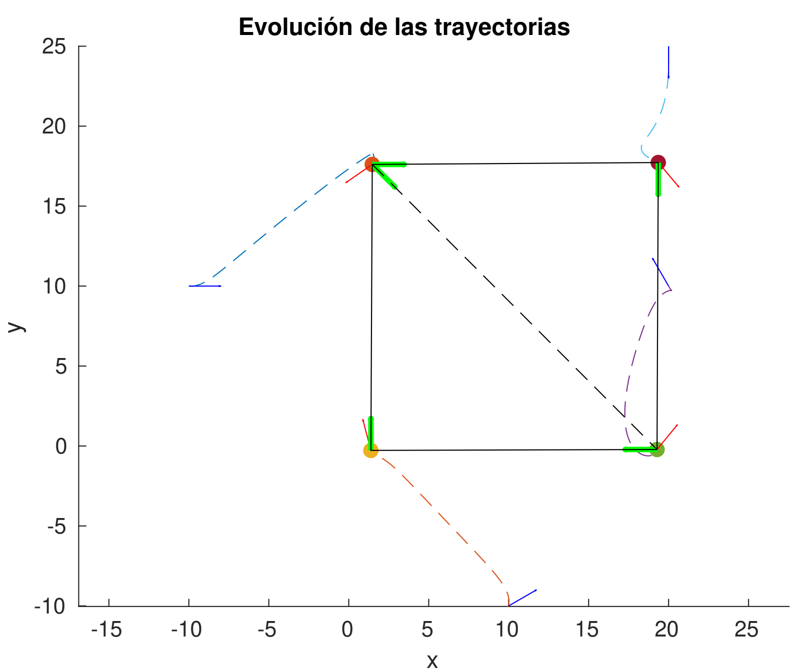
## Pregunta 3.-

Los rangos de la matriz asociada al grafo no infinitesimalmente rígido es igual a 4 y para la infinitesimalmente rígida es igual a 5

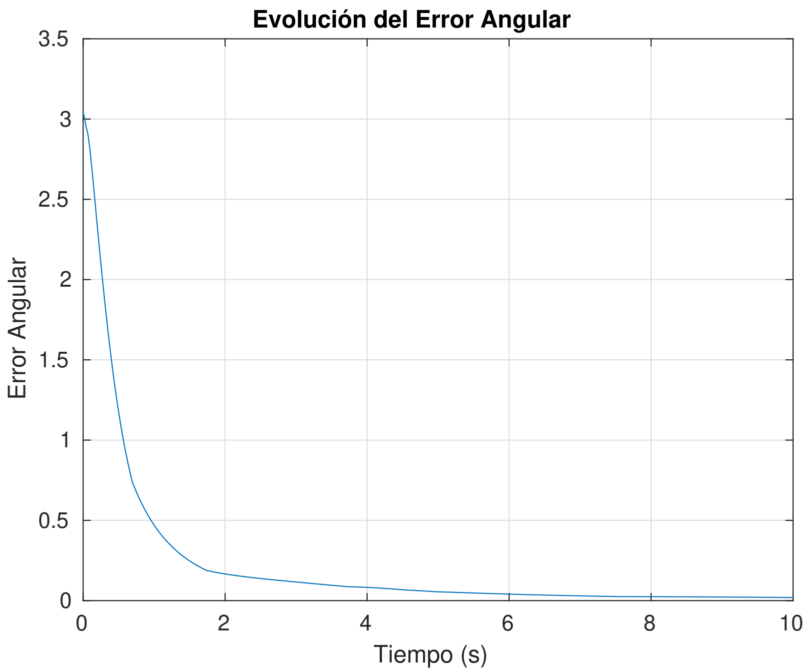
## Pregunta 4.-

Implementando el control basado en ángulos para robots no holonomos como en Zhao&Zelazo obtenemos los siguientes resultados

### Inciso a)

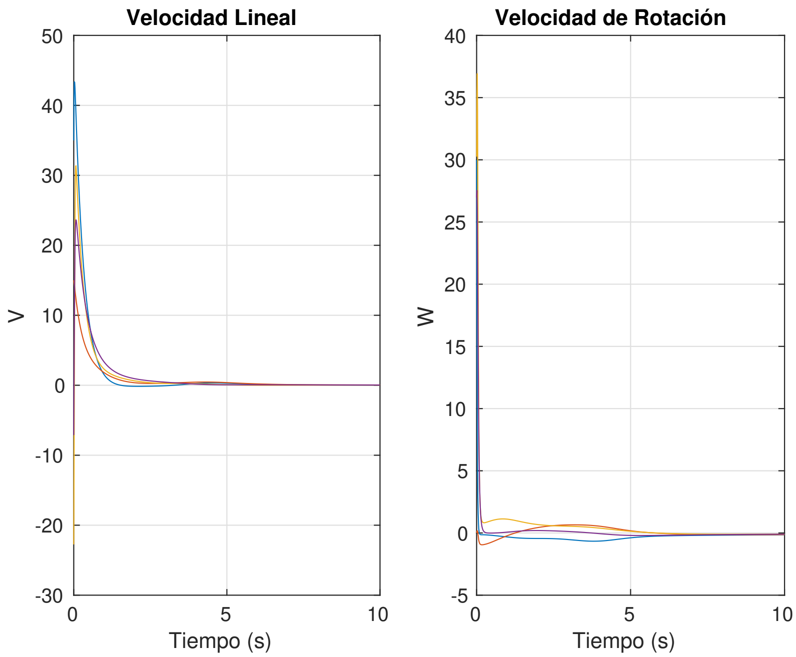


## Inciso b)



Evolución del Error Angular
Error Angular
Tiempo (s)
3.5
3
2.5
2
1.5
1
0.5
0
0
2
4
6
8
10

### Inciso c)




# Pregunta 5.-

En esta tarea se realizó un análisis práctico sobre formaciones basadas en orientación. Primero se examinó el laplaciano de orientación para determinar si un grafo dado es rígido o no en términos de orientación. Luego, se implementó un control para agentes modelados como uniciclo, utilizando la matriz de proyección (de una orientación deseada) para cada agente. Las gráficas presentadas demuestran teóricamente lo estudiado en clase.

Este esquema de control es particularmente útil cuando se requiere mantener la estructura de la formación, pero es necesario escalarla, por ejemplo, para pasar por una abertura o un hueco. En estos casos, es fácil creer que la formación se pierde, pero en realidad, simplemente se está reescalando para adaptarse al entorno. Esto permite una flexibilidad en la navegación y operación en espacios restringidos mientras se mantiene la orientación deseada de los agentes.

# Anexo (Código usado)

```
clearvars;
close all;
```

```
clc;

% N mero de agentes
num_agentes = 4;

% Posiciones iniciales y orientaciones de los agentes (aleatorias)
posiciones = [-10 20 20 10; 10 25 10 -10];
orientaciones = deg2rad([0 270 120 30]);
ang_in = deg2rad([0 270 120 30]);

% Vectores de bearing deseados para formar un cuadrado
g_ast = [1 0; 0 -1; -1 0; 0 1; sqrt(2)/2 -sqrt(2)/2]';

% Matriz de adyacencia para definir las conexiones
A = [1 2; 2 3; 3 4; 4 1; 1 3];

% Par metros de la simulaci n
dt = 0.01;    % Paso de integraci n
T = 10;       % Horizonte de tiempo
t = 0:dt:T;   % Vector de tiempo
N = length(t);% N mero de iteraciones

% Historial de posiciones y orientaciones
trayectorias = zeros(2, num_agentes, N);
orientaciones_hist = zeros(num_agentes, N);

% Historial de las entradas de control
vel_lineal_hist = zeros(num_agentes, N);
vel_rotacion_hist = zeros(num_agentes, N);

% Ganancias de control
kp = 1;

% Funci n de matriz de proyecci n ortogonal
proyeccion = @(x) eye(2) - (x * x') / (x' * x);



% Bucle principal de simulaci n
for k = 1:N
    % Guardar las posiciones y orientaciones actuales
    trayectorias(:, :, k) = posiciones;
    orientaciones_hist(:, k) = orientaciones;

    % Actualizaci n de control para cada agente
    for i = 1:num_agentes
        v = [cos(orientaciones(i)); sin(orientaciones(i))];
        u_v = zeros(2, 1);
        u_w = 0;

        % Control basado en  ngulos
        for j = 1:size(A, 1)
            if A(j, 1) == i || A(j, 2) == i
                idx1 = A(j, 1);
                idx2 = A(j, 2);

                if idx1 == i
```

```
                    idx_neigh = idx2;
                else
                    idx_neigh = idx1;
                end

                % Calcular el bearing actual
                ebearing = posiciones(:, idx_neigh) - posiciones(:,
                     i);
                bearing = ebearing / norm(ebearing);

                % Aplicar la ley de control usando la matriz de
                    proyecci n ortogonal
                Px = proyeccion(g_ast(:, j));
                u_v = u_v + Px * (posiciones(:, idx_neigh) -
                    posiciones(:, i));
            end
        end

        % Control de velocidad lineal y angular
        v_i = v' * u_v;
        w_i = -sin(orientaciones(i)) * u_v(1) + cos(orientaciones(i
            )) * u_v(2);

        % Guardar las entradas de control
        vel_lineal_hist(i, k) = v_i;
        vel_rotacion_hist(i, k) = w_i;

        % Actualizar posiciones y orientaciones
        posiciones(:, i) = posiciones(:, i) + dt * v * v_i;
        orientaciones(i) = orientaciones(i) + dt * w_i;
    end
end

% Graficar la evoluci n de la formaci n
figure;
hold on;
xlabel('x');
ylabel('y');
title('Evoluci n de las trayectorias');

% Dibujar trayectorias y orientaciones de los agentes
for i = 1:num_agentes
    plot(squeeze(trayectorias(1, i, :)), squeeze(trayectorias(2, i,
         :)), '--');
    scatter(trayectorias(1, i, N), trayectorias(2, i, N), 50, '
        filled');
    quiver(trayectorias(1, i, N), trayectorias(2, i, N), cos(
        orientaciones_hist(i, N)), sin(orientaciones_hist(i, N)),
        2, 'r');
    quiver(trayectorias(1, i, 1), trayectorias(2, i, 1), cos(ang_in
        (i)), sin(ang_in(i)), 2, 'b');
    quiver(trayectorias(1, i, N), trayectorias(2, i, N), g_ast(1, i
        ), g_ast(2, i), 2, 'g', 'LineWidth', 2.5);
end

quiver(trayectorias(1, 1, N), trayectorias(2, 1, N), g_ast(1, 5),
    g_ast(2, 5), 2, 'g', 'LineWidth', 2.5);
```

```
% Dibujar las conexiones finales
plot(trayectorias(1, [1 2 3 4 1], N), trayectorias(2, [1 2 3 4 1], N), 'k');
plot(trayectorias(1, [1 3], N), trayectorias(2, [1 3], N), 'k--'); % Diagonal
axis equal;

% Graficar el error de los   ngulos
figure;
error_angular = zeros(1, N);
for k = 1:N
    error_sum = 0;
    for j = 1:size(A, 1)
        idx1 = A(j, 1);
        idx2 = A(j, 2);
        ebearing = trayectorias(:, idx2, k) - trayectorias(:, idx1, k);
        bearing = ebearing / norm(ebearing);
        error_sum = error_sum + norm(g_ast(:, j) - bearing); % Ajuste de acumulaci n de errores
    end
    error_angular(k) = error_sum;
end
plot(t, error_angular);
xlabel('Tiempo (s)');
ylabel('Error Angular');
title('Evoluci n del Error Angular');
grid on;

% Graficar la evoluci n de las entradas de control
figure;
subplot(1, 2, 1);
plot(t, vel_lineal_hist');
xlabel('Tiempo (s)');
ylabel('V');
title('Velocidad Lineal');
grid on;

subplot(1, 2, 2);
plot(t, vel_rotacion_hist');
xlabel('Tiempo (s)');
ylabel('W');
title('Velocidad de Rotaci n');
grid on;
```