Actividad 6 (Evaluación Filtro de KALMAN)

Instituto Tecnológico y de Estudios Superiores de Monterrey

TE3002B.501

Integración de robótica y sistemas Inteligente (Gpo 501)

Semestre: febrero - junio 2024

Alumno:

Jose Angel Ramírez Ramírez A01735529

Profesor:

Alfredo Garcia Suarez

```
function extended_kalman_filter()
```

Se declaran los parámetros para el instante 0, donde al conocer exactamente dónde está el robot, sabemos el valor de MuK y, por lo tanto, no hay covarianza. MuK_ant representa la estimación inicial del estado del robot. SigmaK_ant es la matriz de covarianza inicial que indica la incertidumbre en la estimación del estado (inicialmente nula porque conocemos exactamente el estado).

```
    MuK_ant = [0; 0; 0];
    SigmaK_ant = zeros(3, 3);
```

Se declaran nuestras variables de control que en este caso son la velocidad lineal (vk) y la velocidad angular (wk), también se declara nuestro tiempo de muestreo (dt). Qk es la matriz de ruido del proceso (incertidumbre en la predicción del estado). Rk es la matriz de ruido de la observación (incertidumbre en las mediciones del sensor). landmark_pos es la posición del punto de referencia o marcador.

```
    vk = 1;
    wk = 1;
    dt = 0.1;
```

```
    Qk = [0.50 0.01 0.01;
          0.01 0.50 0.01;
          0.01 0.01 0.20];
    Rk = [0.1  0;
          0    0.02];
    landmark_pos = [3; 4];
```

Se declaran las medidas observadas de nuestro robot por el sensor extereoceptivo en cada uno de los 3 instantes.

```
    Zik_array = [4.87, 0.8;
                 4.72, 0.72;
                 4.69, 0.65];
```

Almacenamiento de resultados para las estimaciones del estado (MuK_store) y las matrices de covarianza (SigmaK_store) para cada iteración.

```
    MuK_store = zeros(3, 3);
    SigmaK_store = zeros(3, 3, 3);
```

A continuación mediante un ciclo for se realizan las tres iteraciones del filtro de Kalman extendido, donde se estima la posición y la orientación del robot en cada paso de tiempo, corrigiendo la estimación con mediciones reales y actualizando la incertidumbre asociada.

```
    for iteracion = 1:3
```

- En cada iteración, se obtiene la medida observada para ese paso de tiempo desde `Zik_array`. Esta medida es proporcionada por el sensor y contiene información sobre la distancia y el ángulo al marcador (landmark).

```
        Zik = Zik_array(iteracion, :)'; % Se obtiene la medida observada
para la iteración actual.
```

- Aquí se predice el nuevo estado del robot utilizando el modelo de movimiento diferencial. Este modelo considera la velocidad lineal (`vk`), la velocidad angular (`wk`) y el tiempo de muestreo (`dt`).

```
        % Paso 2: Predicción del estado usando el modelo de movimiento del
robot.
        MuK_hat = [MuK_ant(1) + dt * vk * cos(MuK_ant(3));
                   MuK_ant(2) + dt * vk * sin(MuK_ant(3));
                   MuK_ant(3) + dt * wk];
```

- Se calcula el Jacobiano de la función de transición de estado, que es una aproximación lineal del modelo no lineal. Este Jacobiano es esencial para la propagación de la incertidumbre en el siguiente paso.

```
        % Paso 3: Calcular el modelo linealizado usando propagación de
incertidumbre (Jacobiano).
        Hk = [1 0 -dt * vk * sin(MuK_ant(3));
              0 1  dt * vk * cos(MuK_ant(3));
              0 0 1];
```

- Se actualiza la matriz de covarianza del estado predicho, que representa la incertidumbre en la predicción del estado. Esto se realiza propagando la incertidumbre anterior y sumando el ruido del proceso (`Qk`).

```
        % Paso 4: Propagación de la incertidumbre, predicción de la
covarianza del estado.
        SigmaK_hat = Hk * SigmaK_ant * Hk' + Qk;
```

- Se calcula la distancia (`dx`, `dy`) y el ángulo esperado (`Zik_hat`) al marcador desde la posición predicha del robot. Este modelo de observación predice lo que el sensor debería medir si la estimación del estado es correcta.

```
        % Paso 6: Calcular medidas con el modelo de observación.
        dx = landmark_pos(1) - MuK_hat(1);
        dy = landmark_pos(2) - MuK_hat(2);
        p = dx^2 + dy^2;

        % Distancia esperada al marcador.
        Zik_hat = [sqrt(dx^2 + dy^2);
                   atan2(dy, dx) - MuK_hat(3)];
```

- Se calcula el Jacobiano de la función de observación, que linealiza la relación no lineal entre el estado del robot y las observaciones del sensor.

```
        % Paso 7: Linealizar el modelo de observación (Jacobiano de la
función de observación).
        Gk = [-dx/(sqrt(p))  -dy/(sqrt(p))  0;
               dy/p          -dx/p         -1];
```

- Se actualiza la covarianza de la innovación, que representa la incertidumbre en la diferencia entre la medida observada y la predicha.

```
        % Paso 8: Incertidumbre de la medición (Covarianza de la innovación).
        Zk = Gk * SigmaK_hat * Gk' + Rk;
```

- Se calcula la ganancia de Kalman, que determina cuánto debe ajustarse la predicción del estado en función de la nueva observación.

```
    % Paso 9: Obtener la ganancia de Kalman.
    Kk = SigmaK_hat * Gk' / Zk;
```

- Se corrige la estimación del estado utilizando la ganancia de Kalman y la diferencia entre la observación real y la predicha. Esto mejora la precisión de la estimación.

```
    % Paso 10: Corrección del estado usando la observación real (Zik).
    MuK = MuK_hat + Kk * (Zik - Zik_hat);
```

- Se actualiza la matriz de covarianza del estado, reduciendo la incertidumbre en la estimación tras considerar la nueva observación.

```
    % Paso 11: Corrección de la covarianza del estado.
    I = eye(3);
    SigmaK = (I - Kk * Gk) * SigmaK_hat;
```

- Se almacenan las estimaciones del estado y las matrices de covarianza para cada iteración para su posterior análisis y graficación.

```
    % Almacenar resultados.
    MuK_store(:, iteracion) = MuK;
    SigmaK_store(:, :, iteracion) = SigmaK;
```

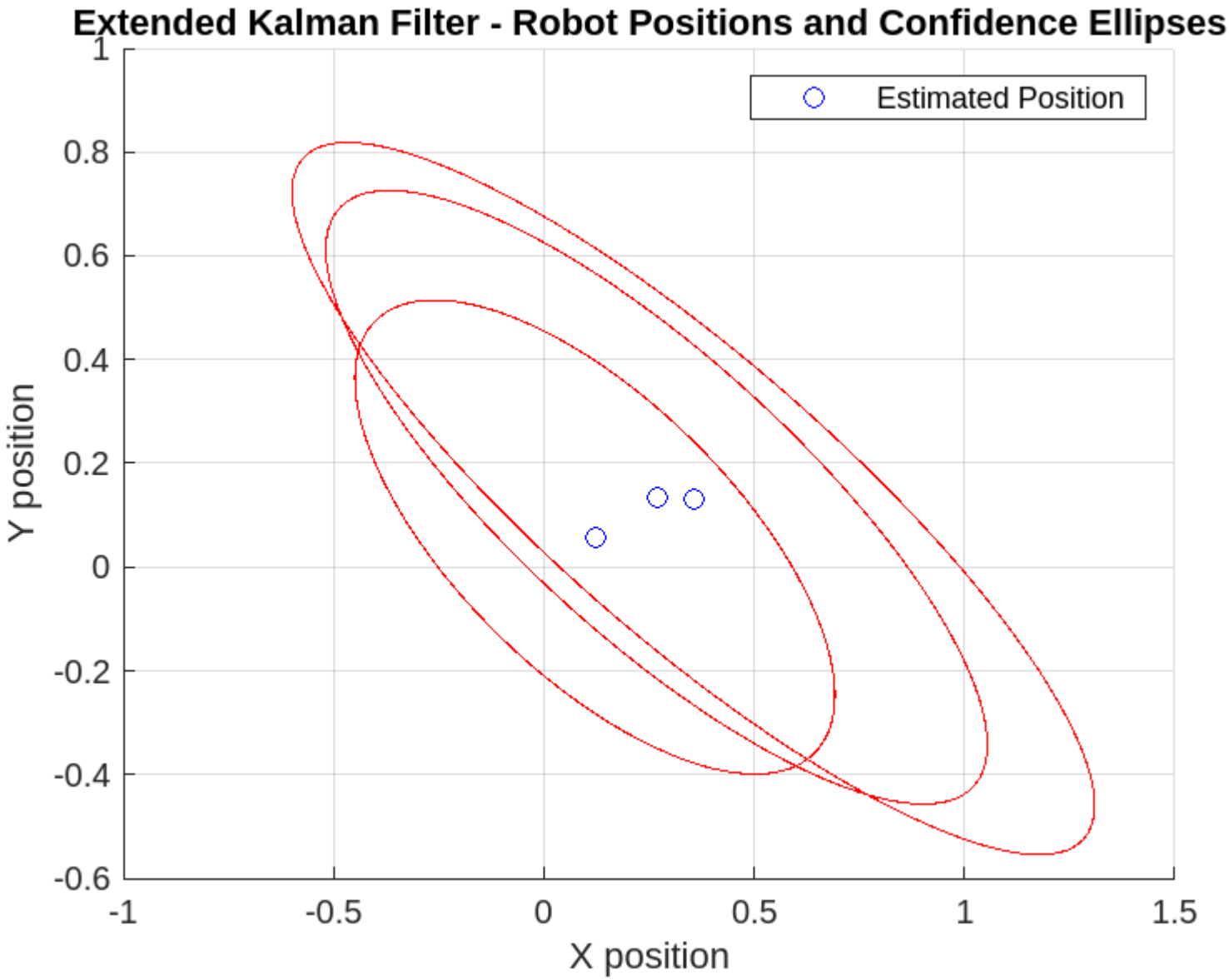
- Se actualizan las variables del estado y la covarianza para la próxima iteración, utilizando las estimaciones corregidas actuales.

```
    % Actualizar estado anterior para la próxima iteración.
    MuK_ant = MuK;
    SigmaK_ant = SigmaK;
end

% Graficar posiciones obtenidas y elipses de confianza.
figure;
hold on;
for iteracion = 1:3
    MuK = MuK_store(:, iteracion);
    SigmaK = SigmaK_store(:, :, iteracion);

    plot(MuK(1), MuK(2), 'bo');
    plot_covariance_ellipse(MuK, SigmaK);
end
xlabel('X position');
ylabel('Y position');
```

```
    title('Extended Kalman Filter - Robot Positions and Confidence
Ellipses');
    legend('Estimated Position');
    grid on;
    hold off;
end

function plot_covariance_ellipse(mu, sigma)
    % Obtener elipses de covarianza para graficar la incertidumbre en la
estimación.
    [eigvec, eigval] = eig(sigma(1:2, 1:2));
    theta = linspace(0, 2*pi, 100);
    circle = [cos(theta); sin(theta)];
    ellipse = eigvec * sqrt(eigval) * circle;
    ellipse = bsxfun(@plus, ellipse, mu(1:2));
    plot(ellipse(1, :), ellipse(2, :), 'r');
end
```

Resultado:

A continuacion se puede observar el resultado de la aplicacion del filtro de kalman, donde logicamente en cada iteracion la elipse de confidencia se incrementa, pero que al mismo tiepo logramos ver que el robot se mentiene dentro de la misma.


Extended Kalman Filter - Robot Positions and Confidence Ellipses
Estimated Position
Y position
X position
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-1
-0.5
0
0.5
1
1.5