PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
Departamento de Ingeniería Mecatrónica

1MTR53 Robótica e Inteligencia Artificial
2024-1

Estimación en Sistemas Robóticos

1 Conceptos Previos

Con el fin de determinar la posición en todo momento de un robot, se puede utilizar información de la velocidad del mismo para realizar una estimación. Entonces, analizando un robot que viaja en un solo eje, de la ecuación diferencial ordinaria (EDO) que relaciona la velocidad y posición en el tiempo se obtiene:

$$\frac{dx(t)}{dt} = v(t) \rightarrow x(t) = \int_0^t v(t)\, dt. \tag{1}$$

No obstante, esta expresión es útil conociendo la velocidad en el caso continuo, es decir, cuando se sabe la velocidad de forma analítica. Así, debido a que los sensores en una aplicación real toman mediciones de la señal de forma discreta (de forma puntual cada cierto tiempo), es necesario realizar una aproximación de la ecuación (1) que permita tomar en consideración estas restricciones, lo que se denomina como **Integración Numérica**.
A continuación, se presentarán dos métodos ampliamente utilizados en robots móviles debido a que pueden ser implementados en un controlador con un costo computacional relativamente bajo. No obstante, se pueden encontrar diversos métodos numéricos avanzados en la literatura como lo es el método de Runge-Kutta.

1.1 Método de Euler

Una forma intuitiva de aproximar una integral es analizándola como el área total bajo la curva de la función. De este modo, considerando muestreos discretos de la señal, se aproxima el área total como la suma del área de los rectángulos formados con bases iguales a la diferencia temporal entre muestreos $\delta_k = t_k - t_{k-1}$ y alturas iguales a cada muestreo de la señal v_k (ver Figura 1).

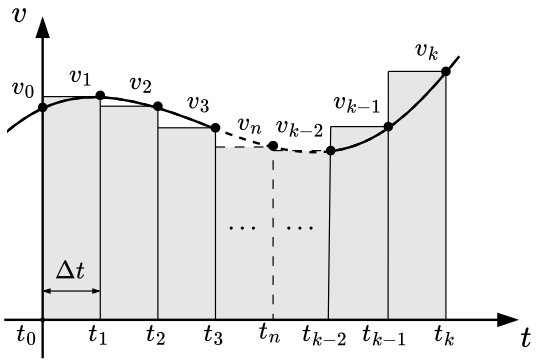

Figura 1: Representación gráfica del método de Euler.

A partir de ello, se obtiene la siguiente expresión:

$$x_k = \sum_{n=1}^{k} v_n \delta_n$$
$$x_k = \sum_{n=1}^{k-1} v_n \delta_n + v_k \delta_k$$
$$x_k = x_{k-1} + v_k \delta_k. \tag{2}$$

Si consideramos que cada muestra se mide con un periodo de muestreo Δt constante ($\delta_k = t_k - t_{k-1} = \Delta t$), se tiene

$$x_k = x_{k-1} + v_k \Delta t. \tag{3}$$

1.2 Método bilineal

Previamente, se mostró la aproximación del área utilizando rectángulos en cada punto muestreado, no obstante es posible mejorar este ajuste utilizando trapecios con bases igual a dos puntos de muestreo contiguos v_k y v_{k-1}, y altura igual a la diferencia temporal entre muestreos δ_k (ver Figura 2).

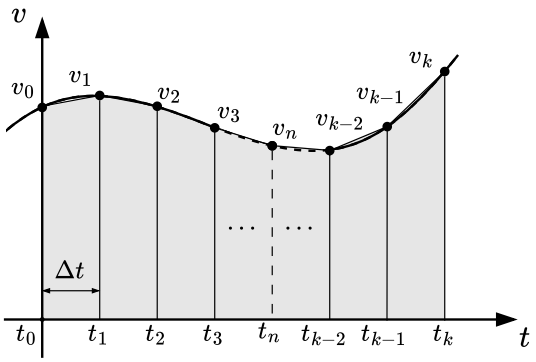

Figura 2: Representación gráfica del método bilineal.

A partir de ello, se obtiene la siguiente expresión:

$$x_k = \sum_{n=1}^{k} (v_n + v_{n-1}) \frac{\delta_n}{2}$$
$$x_k = \sum_{n=1}^{k-1} (v_n + v_{n-1}) \frac{\delta_n}{2} + (v_k + v_{k-1}) \frac{\delta_k}{2}$$
$$x_k = x_{k-1} + (v_k + v_{k-1}) \frac{\delta_k}{2}. \tag{4}$$

Si consideramos que cada muestra se mide con un periodo de muestreo Δt constante ($\delta_k = t_k - t_{k-1} = \Delta t$), se tiene

$$x_k = x_{k-1} + (v_k + v_{k-1}) \frac{\Delta t}{2}. \tag{5}$$

2 Estimación y Localización Relativa

Con el fin de determinar la localización de los sistemas robóticos se requiere utilizar la información de los sensores a disposición para desarrollar algoritmos de estimación de posición y orientación. En este sentido, algunos sensores que pueden ser utilizados para esta tarea son: encoders incrementales y absolutos, unidades de medición inercial, cámaras, sensores GPS, etc.
Particularmente, en un sistema robótico se define los siguientes vectores de posición $\boldsymbol{x} = [x_1 \ \cdots \ x_{n_x}]^T$ $(n_x \in \mathbb{Z}^+)$ y velocidad $\boldsymbol{\nu} = [\dot{x}_1 \ \cdots \ \dot{x}_{n_x}]^T$, de donde $\boldsymbol{\nu} = \dot{\boldsymbol{x}}$. Además, si se cuenta con información de las velocidades de los actuadores $\boldsymbol{\omega} = [\omega_1 \ \cdots \ \omega_{n_u}]^T$ $(n_u \in \mathbb{Z}^+)$, aplicando la transformación con la matriz Jacobiana de velocidad $\mathbf{J}(\boldsymbol{x})$ se tiene:

$$\boldsymbol{\nu} = \mathbf{J}(\boldsymbol{x})\boldsymbol{\omega}. \tag{6}$$

Para un robot que trabaja en un plano bidimensional, como lo son las configuraciones diferencial y omnidirecional, el vector de posiciones es $\boldsymbol{x} = [x \ y \ \theta]^T$, y el vector de velocidades es $\boldsymbol{\nu} = [\dot{x} \ \dot{y} \ \dot{\theta}]^T$. **Si la matrix Jacobiana es constante** $(\mathbf{J}(\boldsymbol{x}) = \hat{\mathbf{J}})$ y el muestreo se realiza con un periodo constante Δt, la aproximación por el método de Euler y bilineal se muestran en (7) y (8), respectivamente.

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \Delta t \, \hat{\mathbf{J}} \boldsymbol{\omega}_k \tag{7}$$

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \frac{\Delta t}{2} \hat{\mathbf{J}} (\boldsymbol{\omega}_k + \boldsymbol{\omega}_{k-1}). \tag{8}$$

En general, para un Δt pequeño, ambos métodos retornan resultados similares. A continuación, se presentan los algoritmos como pseudocódigo que pueden ser implementados en un controlador para la estimación de posición y orientación para el caso presentado anteriormente:

Algorithm 1 Linear Estimation by Euler's Method

Require: $\hat{\mathbf{J}} \in \mathbb{R}^{n_x \times n_u}$, $\Delta t \in \mathbb{R}$
$\boldsymbol{x}_{k-1} \leftarrow [0 \ \cdots \ 0]^T \in \mathbb{R}^{n_x}$
$t_{k-1} \leftarrow 0$
loop
  $\boldsymbol{\omega}_k \leftarrow$ ReadSensors() $\in \mathbb{R}^{n_u}$
  $t_k \leftarrow$ GetTime()
  $\boldsymbol{x}_k \leftarrow \boldsymbol{x}_{k-1} + (t_k - t_{k-1}) \, \hat{\mathbf{J}} \boldsymbol{\omega}_k$
  $\boldsymbol{x}_{k-1} \leftarrow \boldsymbol{x}_k$
  $t_{k-1} \leftarrow t_k$
  Wait until GetTime() $- t_k \geq \Delta t$
end loop

Algorithm 2 Linear Estimation by Bilinial Method

Require: $\hat{\mathbf{J}} \in \mathbb{R}^{n_x \times n_u}$, $\Delta t \in \mathbb{R}$
$\boldsymbol{x}_{k-1} \leftarrow [0 \ \cdots \ 0]^T \in \mathbb{R}^{n_x}$
$\boldsymbol{w}_{k-1} \leftarrow [0 \ \cdots \ 0]^T \in \mathbb{R}^{n_u}$
$t_{k-1} \leftarrow 0$
loop
  $\boldsymbol{\omega}_k \leftarrow$ ReadSensors() $\in \mathbb{R}^{n_u}$
  $t_k \leftarrow$ GetTime()
  $\boldsymbol{x}_k \leftarrow \boldsymbol{x}_{k-1} + \frac{1}{2}(t_k - t_{k-1}) \, \hat{\mathbf{J}} (\boldsymbol{\omega}_k + \boldsymbol{\omega}_{k-1})$
  $\boldsymbol{x}_{k-1} \leftarrow \boldsymbol{x}_k$
  $\boldsymbol{\omega}_{k-1} \leftarrow \boldsymbol{\omega}_k$
  $t_{k-1} \leftarrow t_k$
  Wait until GetTime() $- t_k >= \Delta t$
end loop

En forma general, para una transformación no lineal entre las velocidades de entrada y las velocidades absolutas, es decir

$$\boldsymbol{\nu} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\omega}) \tag{9}$$

las aproximaciones para ambos métodos, respectivamente, son:

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \Delta t \, \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{\omega}_k) \tag{10}$$

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \frac{\Delta t}{2} \left(\boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{\omega}_k) + \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{\omega}_{k-1}) \right). \tag{11}$$

A continuación, se presentan los algoritmos como pseudocódigo que pueden ser implementados en un controlador para la estimación de posición y orientación en el caso general:

Algorithm 3 Non-linear Estimation by Euler's Method

Require: $\hat{\mathbf{J}} \in \mathbb{R}^{n_x \times n_u}$, $\Delta t \in \mathbb{R}$

$\boldsymbol{x}_{k-1} \leftarrow [0 \ \cdots \ 0]^T \in \mathbb{R}^{n_x}$
$t_{k-1} \leftarrow 0$
loop
  $\boldsymbol{\omega}_k \leftarrow$ ReadSensors() $\in \mathbb{R}^{n_u}$
  $t_k \leftarrow$ GetTime()
  $\boldsymbol{x}_k \leftarrow \boldsymbol{x}_{k-1} + (t_k - t_{k-1})\ \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{\omega}_k)$
  $\boldsymbol{x}_{k-1} \leftarrow \boldsymbol{x}_k$
  $t_{k-1} \leftarrow t_k$
  Wait until GetTime() $- t_k \geq \Delta t$
end loop

Algorithm 4 Non-linear Estimation by Bilinial Method

Require: $\hat{\mathbf{J}} \in \mathbb{R}^{n_x \times n_u}$, $\Delta t \in \mathbb{R}$

$\boldsymbol{x}_{k-1} \leftarrow [0 \ \cdots \ 0]^T \in \mathbb{R}^{n_x}$
$\boldsymbol{f}_{k-1} \leftarrow [0 \ \cdots \ 0]^T \in \mathbb{R}^{n_x}$
$t_{k-1} \leftarrow 0$
loop
  $\boldsymbol{\omega}_k \leftarrow$ ReadSensors() $\in \mathbb{R}^{n_u}$
  $t_k \leftarrow$ GetTime()
  $f_k \leftarrow \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{\omega}_k)$
  $\boldsymbol{x}_k \leftarrow \boldsymbol{x}_{k-1} + \frac{1}{2}(t_k - t_{k-1})\ (\boldsymbol{f}_k + \boldsymbol{f}_{k-1})$
  $\boldsymbol{x}_{k-1} \leftarrow \boldsymbol{x}_k$
  $\boldsymbol{f}_{k-1} \leftarrow \boldsymbol{f}_k$
  $t_{k-1} \leftarrow t_k$
  Wait until GetTime() $- t_k >= \Delta t$
end loop

Finalmente, cabe resaltar que si la transformación es altamente no lineal el algoritmo estimará de forma imprecisa la posición y orientación del robot desde un corto periodo de tiempo. En este caso, si se requiere mejorar la precisión se puede reducir el periodo de muestreo de la señal, lo que puede incrementar el costo computacional y requerimientos de los sensores y controlador.