Robótica móvil - Un enfoque probabilístico (86.48)

Universidad de Buenos Aires
Facultad de Ingeniería
Año 2020 - 2do cuatrimestre

TRABAJO PRÁCTICO N.º 2
TEMA: Modelos Probabilísticos y Filtros Discretos.
FECHA: 30 de octubre de 2020
GITHUB: `github.com/fnastasi/8648_Robotica_movil_TPs`

INTEGRANTE :
Nastasi, Franco Gabriel - #100002
`fnastasi@fi.uba.ar`

1. Muestreo de distribuciones de probabilidad

Existen diferentes formas de muestrear una distribución normal $\mathcal{N}(\mu,\sigma^2)$. Una de ellas es utilizando el teorema central del límite con una cantidad de muestras lo suficientemente alta. Si se considera $\mathcal{X}_i \sim \mathcal{U}[-\sigma,\sigma]$, es posible aproximar una distribución normal como:

$$Y = \frac{1}{2}\sum_{i=1}^{12} \mathcal{X}_i + \mu \sim \mathcal{N}(\mu,\sigma^2) \tag{1.1}$$

Entre otros métodos se encuentra el método del muestreo con rechazo que se basa en tomar una muestra de una distribución uniforme en un intervalo dado y tomar otra muestra de una distribución uniforme entre 0 y el máximo de la distribución que se quiere formar. Si la segunda muestra es mayor que el valor de la densidad de probabilidad evaluada en la primera, se descarta la muestra. De esta forma se prevalecen las muestras que tienen mayor probabilidad mientras que aumenta la relación de muestras descartadas con respecto al total de aquellas que tienen baja probabilidad de ocurrencia. El último método que se estudio fue el de Box-Muller, el cual indica que se puede muestrear un distribución normal estandar a partir de 2 distribuciones uniformes $U_1 \sim \mathcal{U}[0,1]$, $U_2 \sim \mathcal{U}[0,1]$ tal que

$$x = \cos(2\pi\mu_1)\sqrt{-2log\mu_2} \tag{1.2}$$

Cada uno de estos métodos se utilizaron para obtener una función que permita muestrear una distribución normal $\mathcal{N}(\mu,\sigma)$. A continuación se muestran con los histogramas resultantes junto con el histograma creado con la función *normrnd* de matlab para una cantidad de muestras $N = 10^5$, $\mu = 0$ y $\sigma = 1$.

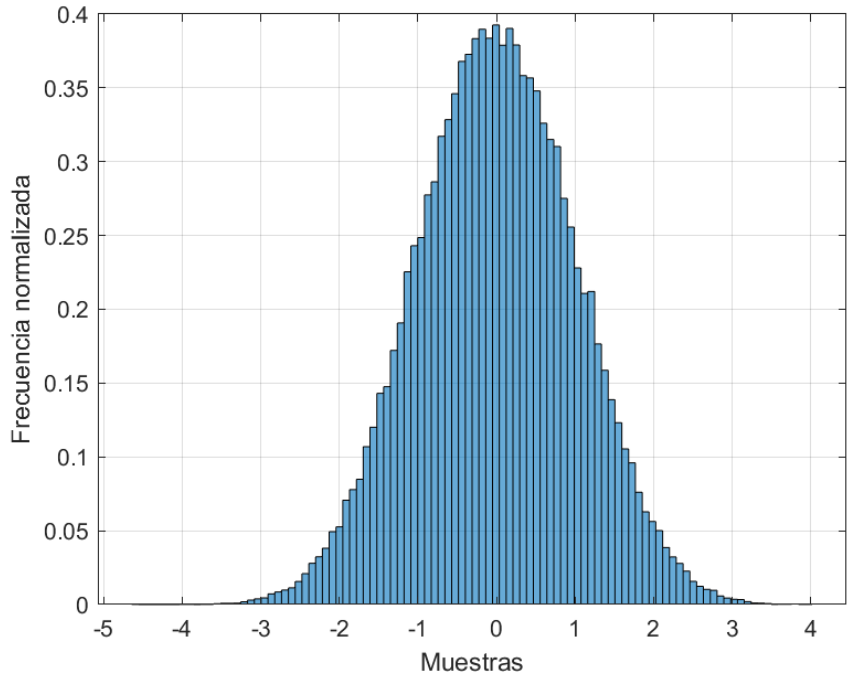

Figura 1.1: Histograma de una distribución normal a partir de la suma de variables uniformes.

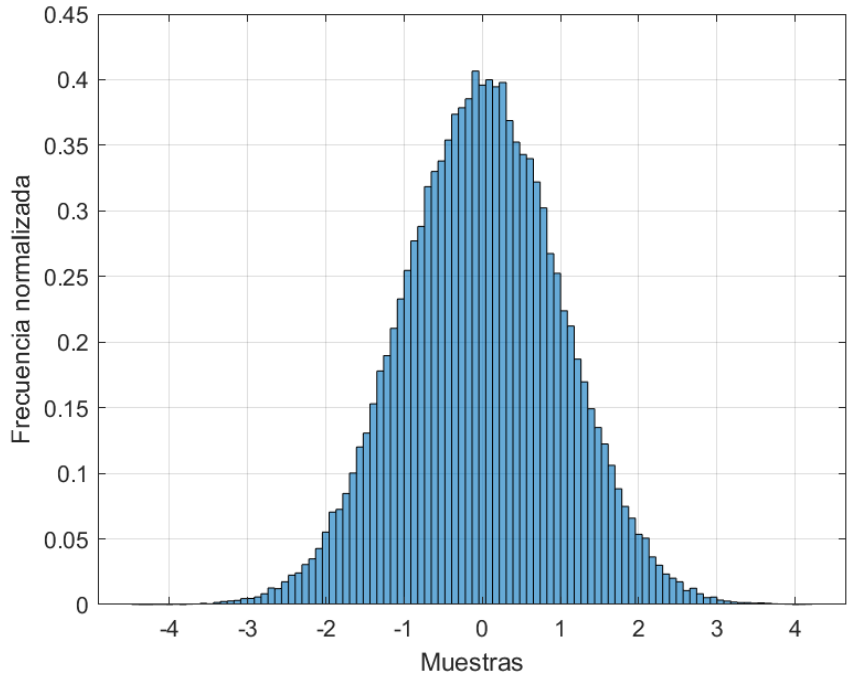

Figura 1.2: Histograma de una distribución normal a partir del método de muestreo con rechazo.

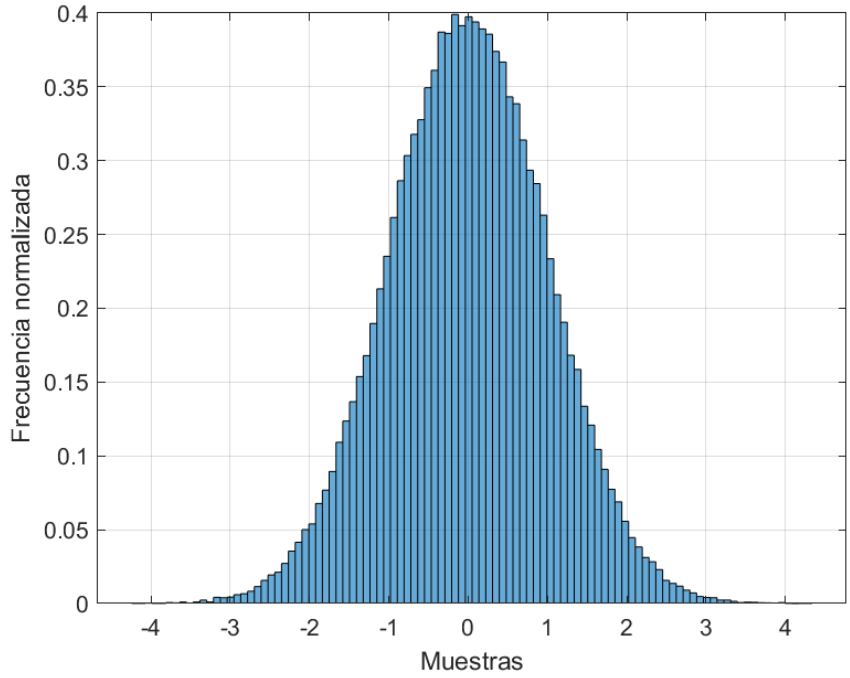

Figura 1.3: Histograma de una distribución normal a partir del método Box-Muller.

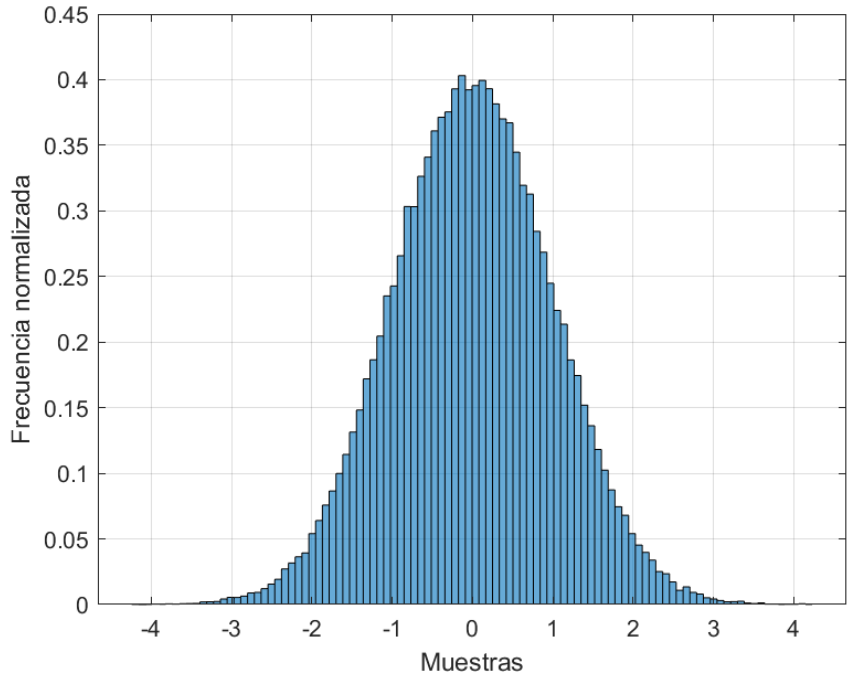

Figura 1.4: Histograma de una distribución normal utilizando la función *normrnd* de matlab.

También se estimó el tiempo que se tarda en tomar una muestra de cada método y se compararon entre sí. Para ello se obtuvo el tiempo en tomar cada muestra y se promedio en las 10^5 muestras. Los resultados se pueden observar en la siguiente tabla en donde se observa como el método de suma de distribuciones uniformes es ligeramente más rápido que los demás. Sin embargo no se observó una diferencia significativa excepto en el método de muestreo con rechazo donde se observo que el tiempo de ejecución del método era aproximadamente 4 mayor a los otros.

Método	Tiempo de ejecución [μs]
Suma de distribuciones uniformes	6.8992
Muestreo con rechazo	33.0934
Box-Muller	8.7162
Función normrnd	9.7254

Tabla 1.1: Comparación de los tiempos de ejecución de los distintos métodos

2. Modelo de movimiento basado en odometría

Se implementó un modelo de movimiento basado en odometría según las mediciones realizadas. Para ello se cuenta con la pose inicial x_t, las mediciones obtenidas u_t y los parámetros α que representan el peso de las incertidumbres asignadas a cada medición.
Utilizando las funciones de muestreo de una distribución normal desarrolladas anteriormente, se logró conocer como se comporta el robot ante las mediciones de odometría agregando error a las mediciones. Se utilizaron los siguientes valores.

$$\boldsymbol{x}_t = \begin{bmatrix} 2{,}0 \\ 4{,}0 \\ 0{,}0 \end{bmatrix} \qquad u_t = \begin{bmatrix} \frac{\pi}{2} \\ 0{,}0 \\ 1{,}0 \end{bmatrix} \qquad \alpha = \begin{bmatrix} 0{,}1 \\ 0{,}1 \\ 0{,}01 \\ 0{,}01 \end{bmatrix} \tag{2.1}$$

Dado los valores de pose inicial y odometría se esperaría que el robot se encuentre en $\boldsymbol{x}_{t+1} = \begin{bmatrix} 2{,}0 \\ 5{,}0 \\ 0{,}0 \end{bmatrix}$ si no

existiese error en las mediciones. Sin embargo al simular el movimiento para 5000 muestras, se obtuvo la siguiente distribución se donde se puede observar la forma similar a una *banana* que se suele obtener

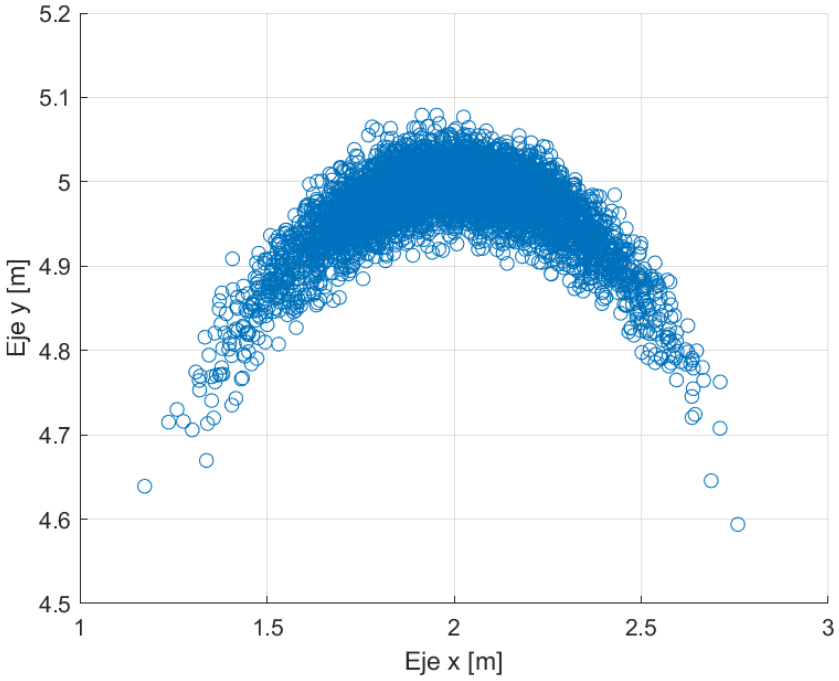

Figura 2.1: Posiciones en $\boldsymbol{x}_{t+1}$ obtenidas al introducir error en la odometría

Adicionalmente, se observó como resultaban las posiciones si se suponían otros parámetros de ruido del modelo. Se tomo $\alpha_M = 10 \times \alpha$ y $\alpha_m = \frac{1}{10}\alpha$. Los resultados se muestran a continuación en donde se observa que para α_m la distribución de puntos en donde se puede encontrar el robot está acotada a un espacio mucho menor dado que al reducir el valor de α el desvío estandar del error introducido es menor. Para el gráfico obtenido suponiendo α_M se observa como la distribución de puntos ocupa un lugar mucho mayor en el espacio y donde se obteniendo una forma circular alrededor de la pose inicial.

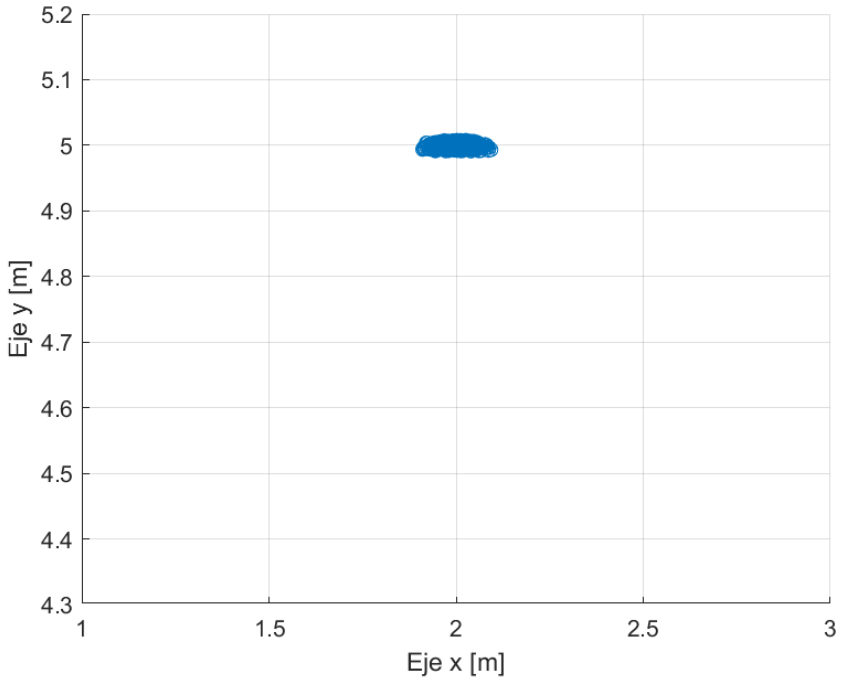

Figura 2.2: Posiciones en x_{t+1} obtenidas al introducir error en la odometría con α_m

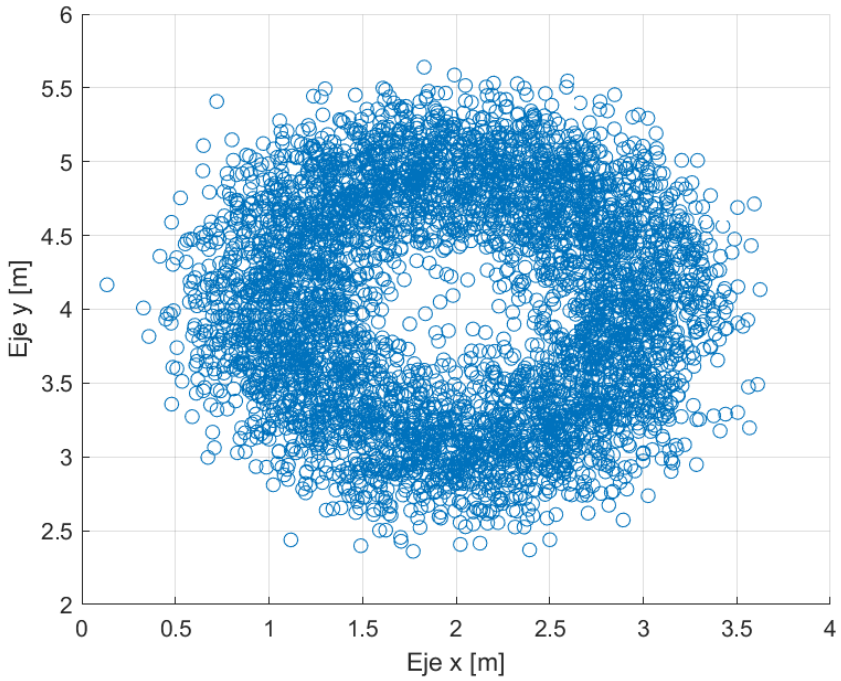

Figura 2.3: Posiciones en x_{t+1} obtenidas al introducir error en la odometría con α_M

3. Filtro discreto

A continuación se muestra el resultado de implementar un filtro de Bayes discreto para conocer la probabilidad de encontrar un robot en un conjunto de celdas. Para ello se conoce la probabilidad de encontrar al robot en una celda dada la información de donde se encontraba y la medición sobre la acción, esto es, si se midió que

el robot avanzó o retrocedió una celda. Dado que no se tiene información acerca del entorno, el filtro de Bayes se puede deducir de forma similar a si se tuviese esa información:

$$\begin{aligned} Bel(x_t) &= P(x_t|u_1,...,u_t) \\ &= \int P(x_t|u_1,...,u_t,x_{t-1})P(x_{t-1}|u_1,....,u_t)dx_{t-1} \\ &= \int P(x_t|u_t,x_{t-1})P(x_{t-1}|u_1,....,u_t)dx_{t-1} \\ &= \int P(x_t|u_t,x_{t-1})P(x_{t-1}|u_1,....,u_{t-1})dx_{t-1} \\ &= \int P(x_t|u_t,x_{t-1})Bel(x_{t-1})dx_{t-1} \end{aligned} \tag{3.1}$$

El filtro de Bayes discreto se logra considerando una sumatoria sobre todos el conjunto de celdas sobre las cuales el robot puede moverse y discretizando el tiempo. Esto es

$$Bel(x_k) = \sum_{x_{k-1}} P(x_k|u_k,x_{k-1})Bel(x_{k-1}) \tag{3.2}$$

En este caso se tiene que el robot puede desplazarse por un conjunto de 20 celdas indexadas a partir de 0 y solo puede avanzar o retroceder. Además se conoce $P(x_k|u_k,x_{k-1})$, por lo tanto es posible aplicar el filtro de Bayes discreto para conocer el *belief* del robot en cada una de las celdas luego de que se realicen una serie de movimientos.
Se aplicó el algoritmo para el caso en que el robot comienza en la celda numero 10 y se realizan 9 movimiento en sentido de avance y 3 en retroceso. Esto es, se comienza conociendo con completa certeza la posición del robot por lo que el *belief* inicial del robot se observa en la figura 3.1. Luego de realizado el movimiento, se observa como se disemina la probabilidad de encontrarse en cada celda aunque se mantiene una mayor probabilidad en la celda 16 que es donde se esperaría encontrar al robot si no existiesen errores en el control. En la figura 3.1 se comparan el *belief* inicial y final mientras que en la figura 3.2 se muestra solamente el *belief* final para una mejor apreciación.

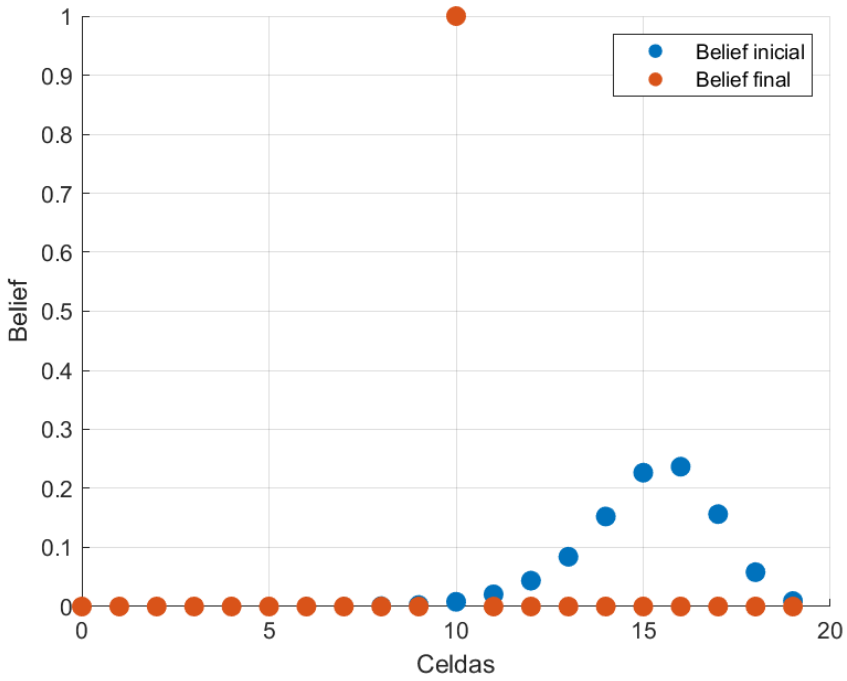

Figura 3.1: *Belief* inicial y final luego de producirse 9 avances consecutivos y 3 retrocesos consecutivos

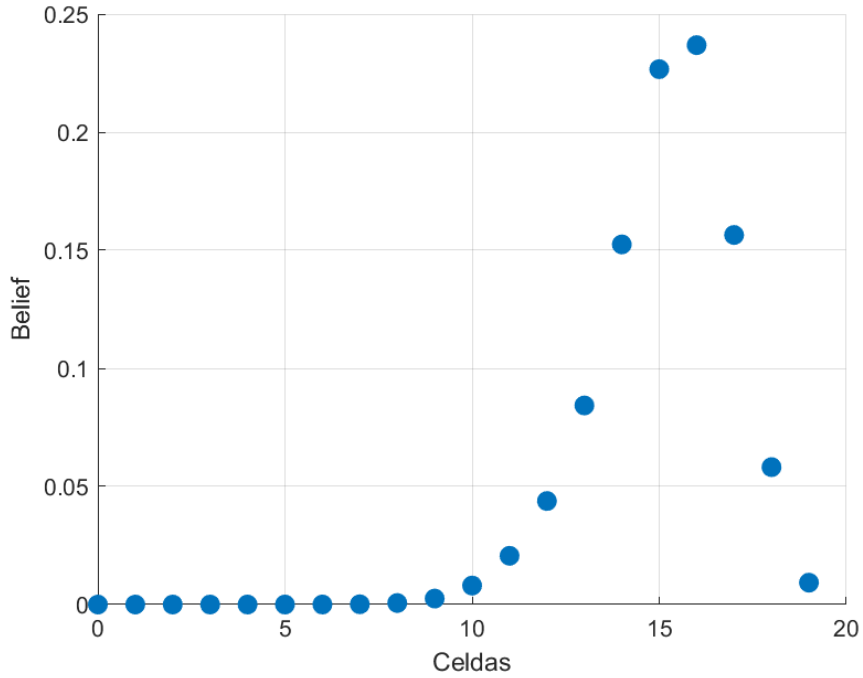

Figura 3.2: *Belief* final luego de producirse 9 avances consecutivos y 3 retrocesos consecutivos

4. Conclusiones

Se logró realizar 2 simulaciones de importancia. Por un lado, se simuló como se introduce error a partir de muestrear distribuciones gaussianas cuando se quiere aplicar odometría para conocer la posición del robot luego de la acción de control. Esto llevó a que se obtengan gráficos en donde la distribución de puntos tiene una forma coherente con lo esperado según se modele una mayor o menor incertidumbre. Por otro lado se logró aplicar un filtro de Bayes discreto en un sistema simple e unidimensional para conocer el belief resultante del movimiento del robot. Ambas simulaciones ayudaron a asimilar los conceptos teóricos vistos.