

Modelando el recorrido de una partícula voladora a través de obstáculos usando procesos de decisión de Markov

Diego Fernando Fonseca Valero

Resumen

En este trabajo queremos mostrar el proceso de toma de decisión de un avión que atraviesa una retícula con obstáculos, además, el avión esta sometido a un viento aleatorio cuya distribución de probabilidad es conocida en cada punto de la retícula. Para tal fin, se modela este problema como un proceso de decisión de Markov.

El problema: Considere un objeto volador que se mueve en una retícula de 200x200 bajo la influencia del viento. La meta del objeto volador es llegar a la región de la retícula con coordenadas $[155, 157] \times [155, 157]$ en el menor tiempo posible. En cada posición en el interior de la retícula el objeto tiene 4 acciones posibles $A = \{u, d, l, r\}$ con probabilidades

de transición dadas por:

$$\mathbb{Q}((z, w)|(x, y), u) = \begin{cases} 0, 3 & \text{si } z = x, w = y + 1 \\ 0, 4 & \text{si } z = x, w = y + 2 \\ 0, 2 & \text{si } z = x - 1, w = y + 2 \\ 0, 1 & \text{si } z = x - 1, w = y + 1 \end{cases}$$

$$\mathbb{Q}((z, w)|(x, y), d) = \begin{cases} 0, 3 & \text{si } z = x, w = y \\ 0, 3 & \text{si } z = x, w = y - 1 \\ 0, 2 & \text{si } z = x - 1, w = y \\ 0, 2 & \text{si } z = x - 1, w = y - 1 \end{cases}$$

$$\mathbb{Q}((z, w)|(x, y), l) = \begin{cases} 0, 3 & \text{si } z = x - 1, w = y + 1 \\ 0, 2 & \text{si } z = x - 1, w = y \\ 0, 3 & \text{si } z = x - 2, w = y \\ 0, 2 & \text{si } z = x - 2, w = y + 1 \end{cases}$$

$$\mathbb{Q}((z, w)|(x, y), r) = \begin{cases} 0, 3 & \text{si } z = x + 1, w = y \\ 0, 4 & \text{si } z = x + 1, w = y + 1 \\ 0, 2 & \text{si } z = x, w = y \\ 0, 1 & \text{si } z = x, w = y + 1 \end{cases}$$

Para las posiciones en el borde de la retícula definimos las probabilidades de transición de forma coherente, eso las definimos mas adelante. Adicionalmente, el objeto quiere evitar algunas regiones de la retícula, no necesariamente conexas, y que alcanza un 5 % del total del área de la retícula. Nuestros objetivos en este trabajo son los siguientes:

- ① Modelar el problema anterior como un problema descontado de horizonte infinito, especificando claramente cada uno de los elementos del problema de decisión de Markov.
- ② Emplear el algoritmo de Value Iteration para encontrar un control óptimo. Escojer dos puntos iniciales distintos y para cada uno de ellos simular varias trayectorias bajo la política óptima encontrada.
- ③ Formular el problema de control como un problema de optimización lineal y resolverlo, especificando la distribución inicial usada. Construir una política óptima a partir de la solución del problema de optimización lineal y compararla con la encontrada en el algoritmo de Value Iteration. Además, queremos responder las preguntas ¿Se pue-

de encontrar una política determinística? ¿Qué pasa cuando cambia la distribución inicial?

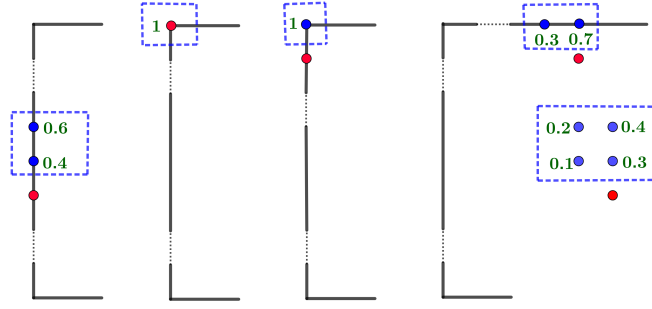
- ④ Usando cualquiera de los métodos anteriores queremos hacer un análisis de sensibilidad de la política óptima encontrada cuando el factor de descuento se acerca a 1.

Solución del problema: Un avión parte de un punto ubicado en una región que esta parametrizada por una retícula grande de tamaño 200×200 , en dicha región existen unos obstáculos que en área cubren el 5 % del territorio, el avión debe llegar a una área dentro de la retícula grande la cual esta parametrizada por la retícula con coordenadas $[155, 157] \times [155, 157]$, debe llegar a dicha región en el menor tiempo posible evitando los obstáculos sin salir de la retícula grande y teniendo en cuenta el efecto del viento el cual esta determinado por la distribución de probabilidad \mathbb{Q} , la cual influye en cada una de las acciones que el avión puede tomar, u subir, d bajar, l izquierda y r derecha. Esta situación se enmarca dentro de un contexto mas amplio en donde el objeto que se traslada de un punto de partida a una región de llegada no necesariamente es un avión, puede ser cualquier partícula, en adelante nos centraremos en la situación del avión ya que representa completamente la situación general.

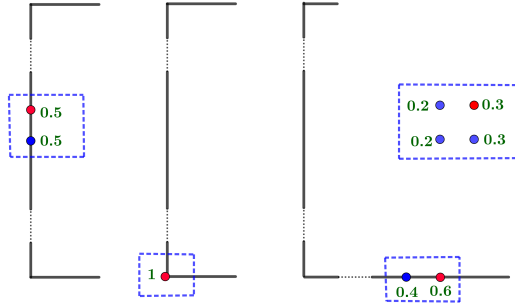
En esta configuración es importante establecer las probabilidades de transición \mathbb{Q} , estas dentro de la retícula de 200×200 fueron dadas en el enunciado, lo que definiremos a continuación son las probabilidades en las fronteras de la retícula, considerando $s' = (z, w)$ y $s = (x, y)$ se tiene:

$$\mathbb{Q}(s'|s, a) = \left\{ \begin{array}{ll} 0,4 & \text{si } a = u, s \in \{0\} \times [0, 198], z = x, w = y + 1 \\ 0,6 & \text{si } a = u, s \in \{0\} \times [0, 198], z = x, w = y + 2 \\ 0,7 & \text{si } a = u, s \in [1, 200] \times \{199\}, z = x, w = y + 1 \\ 0,3 & \text{si } a = u, s \in [1, 200] \times \{199\}, z = x - 1, w = y + 1 \\ 0,7 & \text{si } a = u, s \in [1, 200] \times \{200\}, z = x, w = y \\ 0,3 & \text{si } a = u, s \in [1, 200] \times \{200\}, z = x - 1, w = y \\ 1 & \text{si } a = u, s = (0, 200), z = x, w = y \\ 0,5 & \text{si } a = d, s \in \{0\} \times [1, 200], z = x, w = y \\ 0,5 & \text{si } a = d, s \in \{0\} \times [1, 200], z = x, w = y - 1 \\ 0,6 & \text{si } a = d, s \in [1, 200] \times \{0\}, z = x, w = y \\ 0,4 & \text{si } a = d, s \in [1, 200] \times \{0\}, z = x - 1, w = y \\ 1 & \text{si } a = d, s = (0, 0), z = x, w = y \\ 0,5 & \text{si } a = l, s \in \{0\} \times [0, 199], z = x, w = y \\ 0,5 & \text{si } a = l, s \in \{0\} \times [0, 199], z = x, w = y + 1 \\ 0,5 & \text{si } a = l, s \in \{1\} \times [0, 199], z = x - 1, w = y \\ 0,5 & \text{si } a = l, s \in \{1\} \times [0, 199], z = x - 1, w = y + 1 \\ 0,5 & \text{si } a = l, s \in [2, 200] \times \{200\}, z = x - 2, w = y \\ 0,5 & \text{si } a = l, s \in [2, 200] \times \{200\}, z = x - 1, w = y \\ 1 & \text{si } a = l, s = (1, 200), z = x - 1, w = y \\ 1 & \text{si } a = l, s = (0, 200), z = x, w = y \\ 0,5 & \text{si } a = r, s \in \{200\} \times [0, 199], z = x, w = y \\ 0,5 & \text{si } a = r, s \in \{200\} \times [0, 199], z = x, w = y + 1 \\ 0,3 & \text{si } a = r, s \in [0, 199] \times \{200\}, z = x, w = y \\ 0,7 & \text{si } a = r, s \in [0, 199] \times \{200\}, z = x + 1, w = y \\ 1 & \text{si } a = r, s = (200, 200), z = x, w = y \end{array} \right.$$

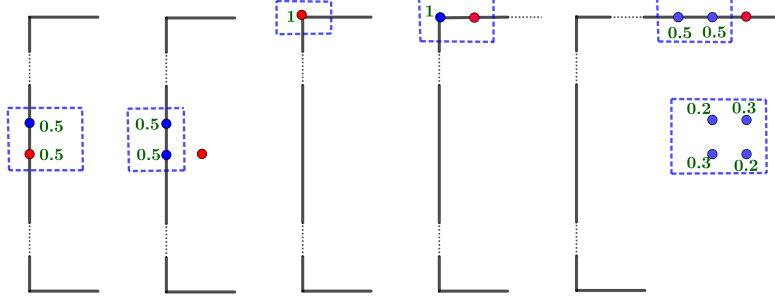
Las probabilidades de transición son ilustradas gráficamente en la Figura 1.



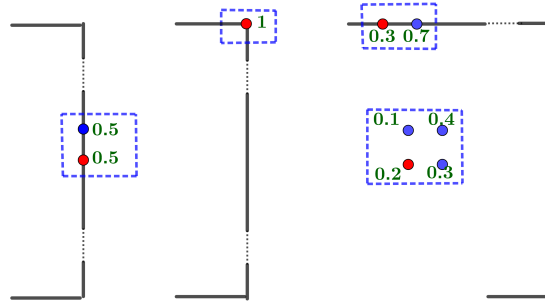
(a)



(b)



(c)



(d)

Figura 1: Definición de las probabilidades de transición $\mathbb{Q}(s'|s, a)$ en donde el punto rojo es s y los puntos encerrados por el recuadro de líneas discontinuas azul son los puntos s' para los cuales $\mathbb{Q}(s'|s, a) > 0$, en estas imágenes se muestran todas las posibles configuraciones de s y s' que se pueden dar dentro de la retícula de 200×200 , además tenemos en (a) $a = u$, (b) $a = d$, (c) $a = l$ y (d) $a = r$. En verde están las probabilidades.

- ① El problema de guiar un avión desde un punto de partida a una zona determinada sujeto a condiciones adversas como obstáculos y vientos se puede caracterizar como un Proceso de Decisión de Markov descontado de horizonte infinito con la siguiente caracterización:

Tiempo: $T = \{0, 1, 2, 3, \dots, N\}$ con $N = \infty$.

Espacio de estados: $S := [0, 200] \times [0, 200]$ posibles ubicaciones en donde puede estar el avión.

Acciones: $A = \{u, d, l, r\}$, el avión solo puede u subir, d bajar, l girar a la izquierda ó r girar a la derecha..

Acciones admisibles: $A(s) = \{u, d, l, r\}$, sin importar la ubicación del avión esta siempre tiene estas posibilidades de decisión.

Transición: La transición del proceso es

$$S_{n+1} = f(S_n, a, \xi_n)$$

donde $a \in A$ y los posibles valores de $f(s, a, \xi_n)$ con $s = (x, y)$ son los elementos del conjunto

$$s + \mathcal{P}_a^s := \{s + p_a^s \mid p_a^s \in \mathcal{P}_a^s\}$$

donde

$$\mathcal{P}_a^s = \begin{cases} \{(0, 1), (0, 2), (-1, 2), (-1, 0)\} & \text{si } a = u, s \in [1, 200] \times [0, 198] \\ \{(0, 1), (0, 2)\} & \text{si } a = u, s \in \{0\} \times [0, 198] \\ \{(0, 1), (-1, 1)\} & \text{si } a = u, s \in [1, 200] \times \{199\} \\ \{(0, 0), (-1, 0)\} & \text{si } a = u, s \in [1, 200] \times \{200\} \\ \{(0, 0)\} & \text{si } a = u, s = (0, 200) \\ \{(0, 0), (0, -1), (-1, -1), (-1, 0)\} & \text{si } a = d, s \in [1, 200] \times [1, 200] \\ \{(0, 0), (0, -1)\} & \text{si } a = d, s \in \{0\} \times [1, 200] \\ \{(0, 0), (-1, 0)\} & \text{si } a = d, s \in [1, 200] \times \{0\} \\ \{(0, 0)\} & \text{si } a = d, s = (0, 0) \\ \{(-1, 0), (-1, 1), (-2, 1), (-2, 0)\} & \text{si } a = l, s \in [2, 200] \times [1, 199] \\ \{(0, 0), (0, 1)\} & \text{si } a = l, s \in \{0\} \times [0, 199] \\ \{(-1, 0), (-1, 1)\} & \text{si } a = l, s \in \{1\} \times [0, 199] \\ \{(-2, 0), (-1, 0)\} & \text{si } a = l, s \in [2, 200] \times \{200\} \\ \{(-1, 0)\} & \text{si } a = l, s = (1, 200) \\ \{(0, 0)\} & \text{si } a = l, s = (0, 200) \\ \{(0, 0), (0, 1), (1, 1), (1, 0)\} & \text{si } a = r, s \in [0, 199] \times [0, 199] \\ \{(0, 0), (0, 1)\} & \text{si } a = r, s \in \{200\} \times [0, 199] \\ \{(0, 0), (1, 0)\} & \text{si } a = r, s \in [0, 199] \times \{200\} \\ \{(0, 0)\} & \text{si } a = r, s = (200, 200) \end{cases}$$

Las probabilidades mediante las cuales s alcanza cada valor $s + p_a^s$ son determinadas por \mathbb{Q} , es decir, si $s' = (z, w) = s + p_a^s$ siendo \mathbb{P} la distribución de ξ_n para toda n , entonces

$$\mathbb{P}(f(s, a, \xi_n) = s') = \mathbb{Q}(s'|s, a)$$

donde \mathbb{Q} fue definida para el interior de la retícula en el enunciado del ejercicio y para las fronteras en el inicio de la solución de este problema.

Probabilidad de transición: Esta probabilidad \mathbb{Q} ya fue definida para el interior de la retícula en el enunciado de este problema y para la frontera en el inicio de la solución, concretamente, la Figura 1 muestra de manera gráfica las probabilidades \mathbb{Q} para cada acción.

Recompensa: Sean \mathcal{O} y \mathcal{M} conjuntos de puntos en la retícula $([0, 200] \times [0, 200]) \cap (\mathbb{Z} \times \mathbb{Z})$ que conforman los **obstáculos** y la **meta** (o región objetivo) respectivamente. Los obstáculos que consideraremos en este problema son los representados en la Figura 2.

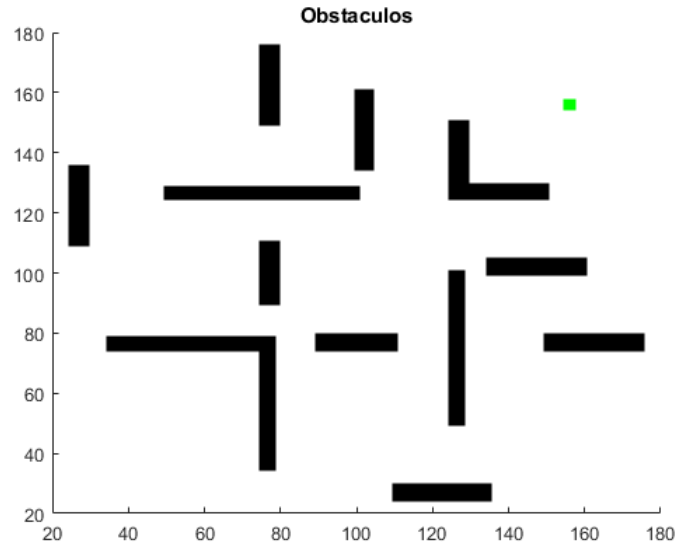


Figura 2: En negro los obstáculos que debe sortear el avión y en verde la región objetivo a donde debe llegar el avión.

A partir de lo inmediatamente anterior definimos la función h para cada $s \in$

$([0, 200] \times [0, 200]) \cap (\mathbb{Z} \times \mathbb{Z})$ por

$$h(s) := \begin{cases} -100,000,000 & \text{si } s \in \mathcal{O} \\ 10,000 & \text{si } s \in \mathcal{M} \\ 1 & \text{en otro caso.} \end{cases}$$

A partir de esta función definimos la **función recompensa** \mathbf{r} para cada $s \in ([0, 200] \times [0, 200]) \cap (\mathbb{Z} \times \mathbb{Z})$ y acción a por

$$\mathbf{r}(s, a) := \frac{4}{|P_a^s|} \sum_{p_a^s \in P_a^s} h(s + p_a^s).$$

Esta forma de establecer la recompensa permite que el avión no ingrese a ninguno de los obstáculos ya que si se esta en una ubicación s cerca de algún obstáculo y uno de los puntos que sugiere una hipotética a acción esta dentro del obstáculo entonces $\mathbf{r}(a, s)$ tendrá un valor muy bajo.

El factor $\frac{4}{|P_a^s|}$ permite que si un estado esta cerca o en una frontera de la retícula la recompensa siga valiendo lo mismo que en los lugares donde no hay frontera cerca, recordemos que $|P_a^s| = 4$ si no se esta cerca a fronteras, $|P_a^s| = 2$ si se esta cerca o en las fronteras y $|P_a^s| = 1$ si esta cerca o en las esquinas, la noción de cercanía depende de cada acción a . En la Figura 3 se ilustra el valor de la recompensa para cada acción.

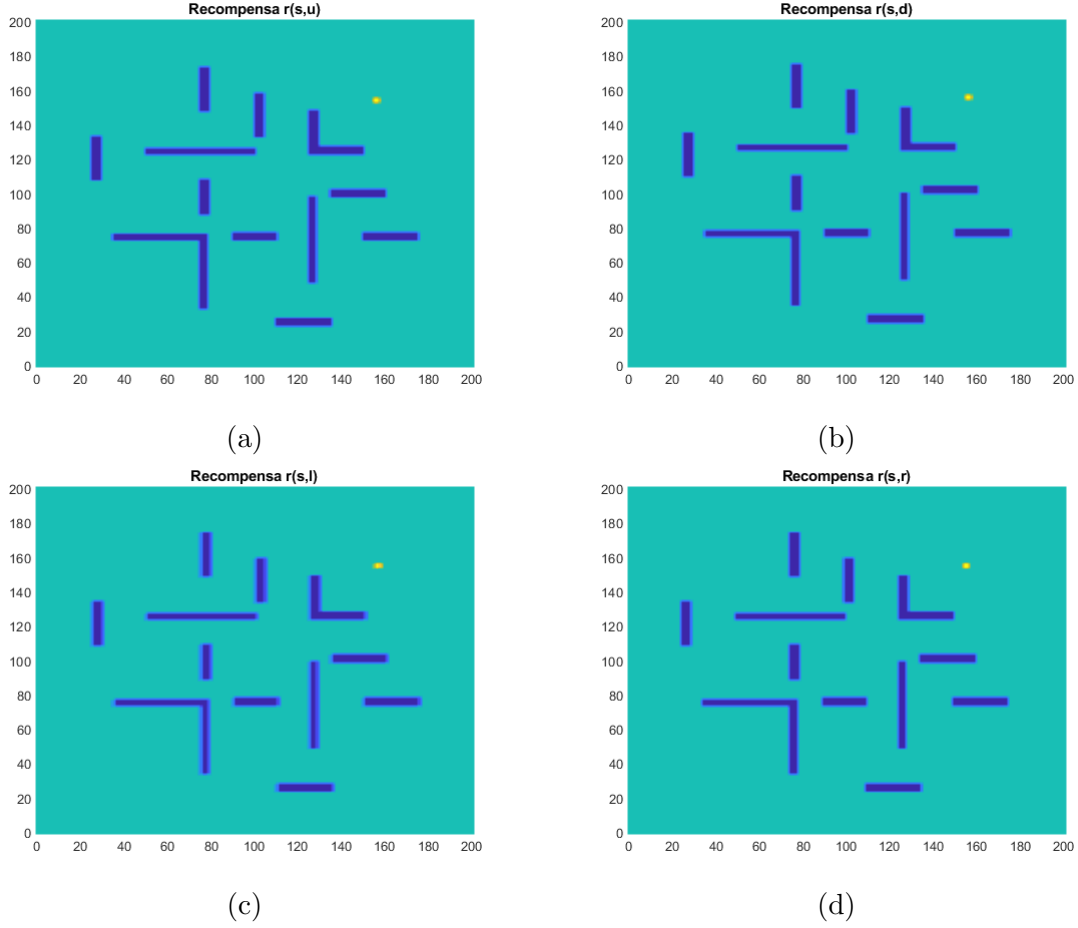


Figura 3: El valor que la recompensa r otorga al tomar la acción a estando en el estado s , si el punto s es azul oscuro entonces la recompensa es lo más baja posible, si el color es amarillo entonces la recompensa es lo mas alto posible. En esta figura se tiene en (a) $a = u$, en (b) $a = d$, en (c) $a = l$, en (d) $a = r$.

- ② Para esta parte se seleccionó el método de **Value iteration** (iteración por valores) pues este no requiere solucionar sistemas de ecuaciones como si ocurre con Policy iteration, sin embargo, como se evidencia en el código de nuestro algoritmo de Value iteration dicho método no es menos complejo de programar.

En la Figura 4 consideramos $\lambda = 0,99$ y $\varepsilon = 10^{-7}$, los colores ■, ■, ■ y ■ representan subir, bajar, izquierda y derecha respectivamente, por otro lado, el color ■ representa un empate entre subir e ir a izquierda, ■ representa un empate entre subir e ir a la derecha, ■ representa un empate entre bajar e ir a izquierda y ■ un empate entre

bajar e ir a la derecha. Cuando nos referimos a la palabra 'empate' significa que se pueden tomar cualquiera de las dos acciones empatadas. En la Figura 4 (a) se considera la política con los empates, en (b) se considera sin empates, en esta solo aparecen los colores ■, ■, ■ y ■, esta es la política que por defecto entrega nuestro algoritmo de Value Iteration, esta es elegida considerando un orden en las acciones, dicho orden es $u < d < l < r$, de modo que si se presenta un empate entre dos acciones se elige la menor acción. Esta última política es la que emplearemos para las simulaciones.

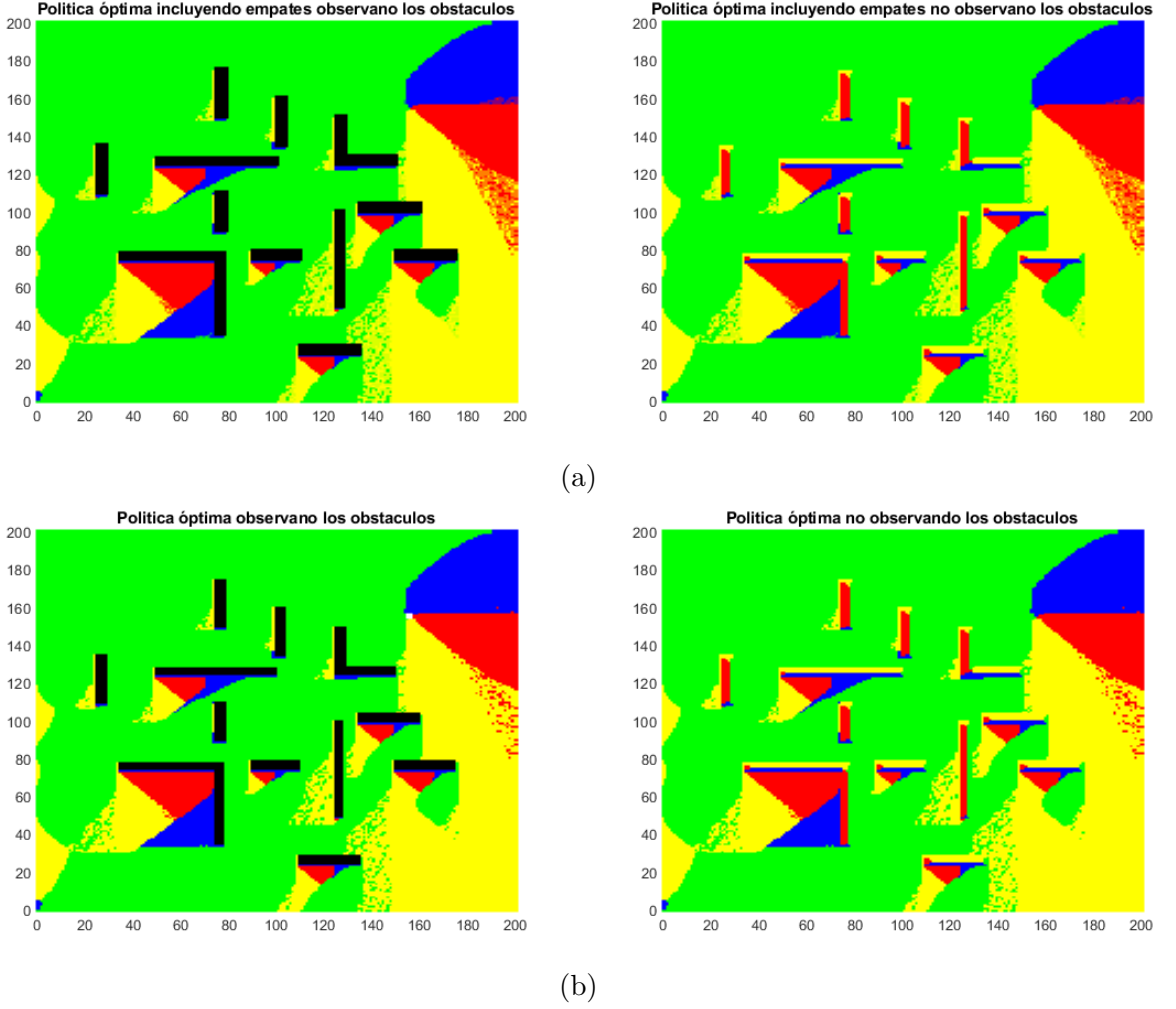


Figura 4: Política óptima obtenida de Value Iteration para $\lambda = 0,99$ y $\varepsilon = 10^{-7}$. En (a) se consideran todas las posibles políticas óptimas que se obtienen al considerar los empates, en (b) la política óptima por default que entrega nuestro algoritmo de value iteration.

Los empates se producen ya que en el desarrollo del algoritmo de Value iteration, concretamente en el paso 4 (así se identifico dicho paso en la clase), el argumento máximo

$$\mu_{\varepsilon}(s) = \operatorname{argmin}_{a \in A} \left\{ \mathbf{r}(s, a) + \lambda \sum_{s' \in S} \mathbb{Q}(s'|s, a) v^{n+1}(s') \right\}$$

no es único, experimentalmente se comprobó que para algunos estados existen dos acciones que alcanzan dicho máximo, además se evidenció que no se dan empates entre acciones opuestas, por ejemplo subir-bajar, o derecha-izquierda.

Teniendo en cuenta la política obtenida por nuestro algoritmo de Value iteration, en la Figura 5 se presentan 100 simulaciones de recorridos del avión partiendo desde diferentes puntos de la retícula 200×200 y regido por la política de la Figura 4 (b).

De esta figura se observa que eventualmente todos los recorridos terminan en la meta, sin embargo se observa un comportamiento interesante en las Figuras 5 (a), (b) (d), algunos recorridos pasan cerca de la meta pero deciden subir mas hasta casi cerca a la frontera para luego bajar y encaminarsen hacia la meta, este fenómeno es coherente con la política de la Figura 4 (b) pero no deja de ser curioso, una explicación para dicho fenómeno es la forma como se establecieron las probabilidades \mathbb{Q} dentro del retículo de 200×200 , se podría pensar que \mathbb{Q} modela la aleatoriedad de un viento que va de sur-oriente a nor-occidente pero dicha apreciación no es correcta ya que $\mathbb{Q}(s'|s, r)$ esta demasiado cargada hacia el nor-oriente, esto hace que cerca a la meta se presente dicho comportamiento.

Las imágenes en la Figura 5 fueron generadas por el Algoritmo 3 el cual intenta presentar mas de una simulación de recorridos del avión partiendo desde un punto inicial fijo y para una política dada. Este presenta una imagen de la retícula con los obstáculos y una silueta en escala de rojos en donde los puntos con mayor intensidad de rojo es en donde mas recorridos simulados pasaron. Este algoritmo hace uso del Algoritmo 2 el cual crea la simulación de un recorrido del avión partiendo desde un punto inicial y para una política dada.

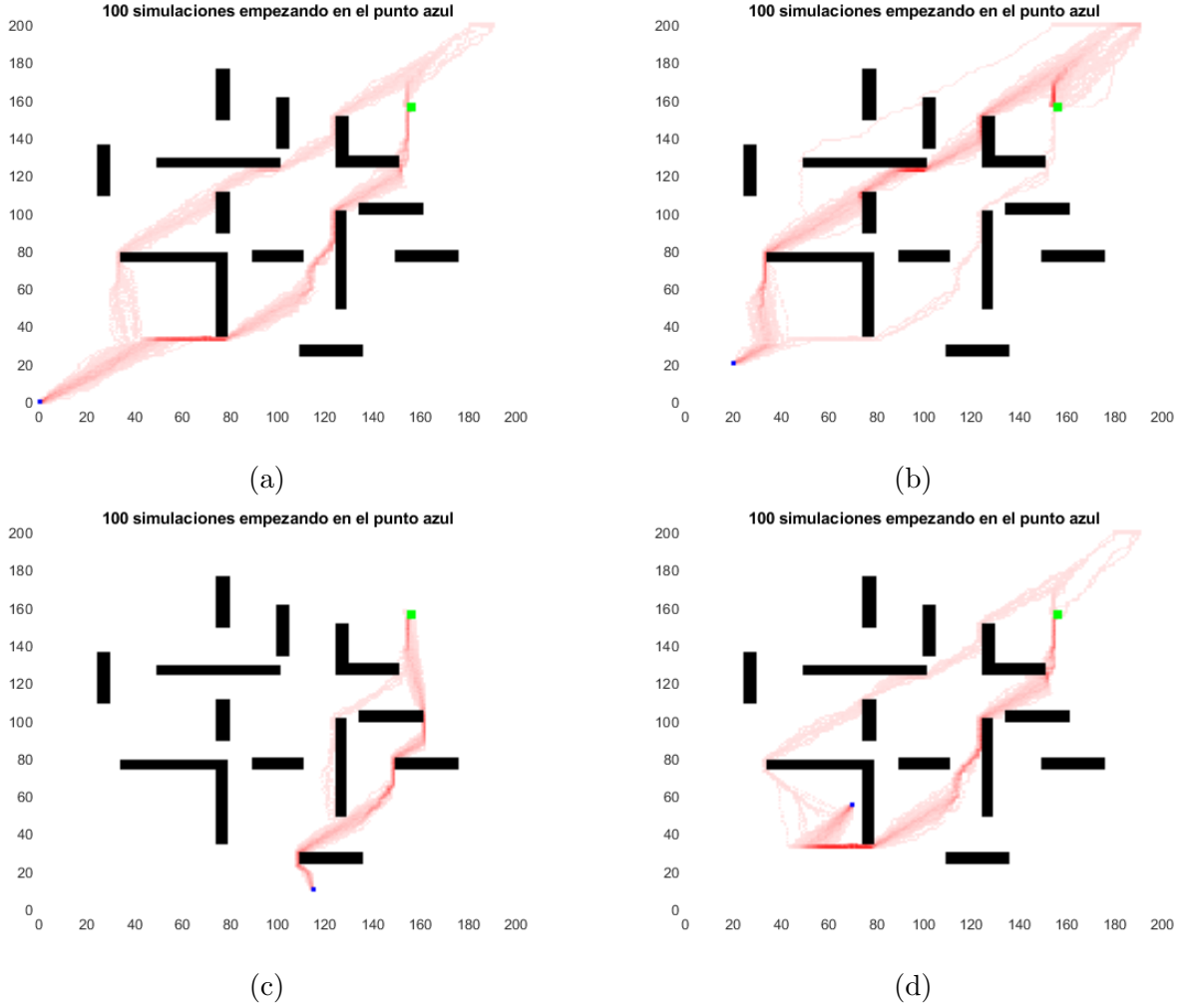


Figura 5: Simulación de 100 recorridos del avión regido por la política de la Figura 5 y partiendo del punto (a) $(0,0)$ (b) $(20,20)$ (c) $(115,10)$ y (d) $(70,55)$. En azul el punto de partida y en verde la región de llegada.

El algoritmo de Value Iteration propuesto en este documento es el Algoritmo 1 en el Apéndice, en este algoritmo se evita el uso de todo tipo de bucle como **for**, en ese sentido se vectorizó todo el código, esto hizo que el código fuera muy largo, pero la ganancia estuvo en el tiempo computacional, por ejemplo, para encontrar las políticas de la Figura 4 nuestro algoritmo tardó aproximadamente **37 segundos**, experimentalmente se evidencia que dicho tiempo aumenta a medida que λ se acerca a 1 y ε se mantiene fijo, esto es de esperar ya que el número de iteraciones necesarias

aumenta, por ejemplo, para $\lambda = 0,999$ y $\varepsilon = 10^{-7}$ tardo aproximadamente 324 segundos.

- ③ Consideramos v como un vector de longitud $|S| = 40401$ y α un vector de entradas positivas de tamaño $|S|$ y tal que $\alpha_1 + \dots + \alpha_n = 1$, asumimos v vector vertical y α horizontal. Con estas consideraciones tenemos que el problema lineal asociado al problema de decisión de Markov descontado es dado por

$$(P) \quad \begin{cases} \min_{v \in \mathbb{R}^{|S|}} & \alpha v \\ \text{sujeto a} & Av \geq \mathbf{r}. \end{cases} \quad (1)$$

Para describir la matriz A y el vector vertical \mathbf{r} debemos definir otras matrices y enumerar S como $S = \{s_1, s_2, \dots, s_{|S|}\}$, a partir de esto se tiene para cada $a \in \{u, d, l, r\}$ las matrices

$$\mathcal{Q}_a := \begin{bmatrix} \mathbb{Q}(s_1|s_1, a) & \mathbb{Q}(s_2|s_1, a) & \cdots & \mathbb{Q}(s_{|S|}|s_1, a) \\ \mathbb{Q}(s_1|s_2, a) & \mathbb{Q}(s_2|s_2, a) & \cdots & \mathbb{Q}(s_{|S|}|s_2, a) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{Q}(s_1|s_{|S|}, a) & \mathbb{Q}(s_2|s_{|S|}, a) & \cdots & \mathbb{Q}(s_{|S|}|s_{|S|}, a) \end{bmatrix}.$$

Esta es una matriz de tamaño $|S| \times |S|$ y de la forma como esta establecida \mathbb{Q} se sigue que \mathcal{Q}_a es una matriz **sparse** ya que cada fila a lo mas tiene 4 entradas diferentes de cero. Siguiendo, denotamos por $I_{|S|}$ la matriz identidad de tamaño $|S| \times |S|$, por lo tanto, la matriz A es dada por

$$A = \begin{bmatrix} I_{|S|} \\ I_{|S|} \\ I_{|S|} \\ I_{|S|} \end{bmatrix} - \lambda \begin{bmatrix} \mathcal{Q}_u \\ \mathcal{Q}_d \\ \mathcal{Q}_l \\ \mathcal{Q}_r \end{bmatrix}. \quad (2)$$

De esta construcción se sigue que A es una matriz de tamaño $4|S| \times |S|$. Adicionalmente, se tiene que \mathbf{r} es de la forma

$$\mathbf{r}^t = [\mathbf{r}(s_1, u), \dots, \mathbf{r}(s_{|S|}, u), \mathbf{r}(s_1, d), \dots, \mathbf{r}(s_{|S|}, d), \mathbf{r}(s_1, l), \dots, \mathbf{r}(s_{|S|}, l), \mathbf{r}(s_1, r), \dots, \mathbf{r}(s_{|S|}, r)]$$

donde t indica *transpuesto*.

Nuestro interés se enfoca en el problema dual ya que este permite construir una política, en ese sentido, el problema dual es dado por:

$$(D) \quad \begin{cases} \min_{\mu \in \mathbb{R}^{4|S|}} & \mathbf{r}^t \mu \\ \text{sujeto a} & A^t \mu = \alpha^t \\ & \mu \geq 0. \end{cases} \quad (3)$$

donde μ debe ser visto como

$$\mu^t = [\mu(s_1, u), \dots, \mu(s_{|S|}, u), \mu(s_1, d), \dots, \mu(s_{|S|}, d), \mu(s_1, l), \dots, \mu(s_{|S|}, l), \mu(s_1, r), \dots, \mu(s_{|S|}, r)].$$

Recordemos que si μ^* es solución de (3) entonces π_{μ^*} la política inducida por μ^* es dada por

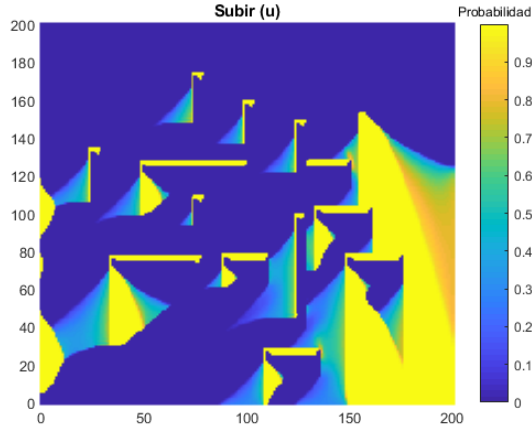
$$\pi_{\mu^*}(a|s) := \frac{\mu^*(s, a)}{\sum_{s' \in S} \mu^*(s', a)}.$$

Esta política no necesariamente es determinista.

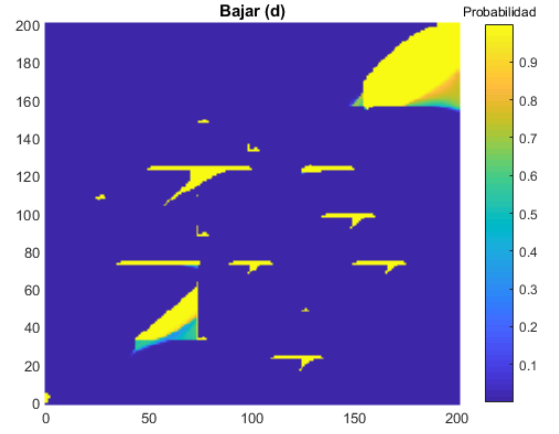
Con estas consideraciones en mente procedemos a solucionar el problema dual (3), para tal fin empleamos el Algoritmo 4, en este empleamos el solver **LinProg** con el método de **Punto Interior**, además, la matriz A es creada con el comando **sparse**. En la Figura 6 se expone la política no determinística asociada a la solución del problema dual (3) para $\lambda = 0,99$ y $\alpha = \frac{1}{|S|}\mathbb{1}_{|S|}$ donde $\mathbb{1}_{|S|}$ es un vector de longitud $|S|$ donde todas su componentes son 1.

Para comparar la política del punto anterior consignada en la Figura 4 (b) y la política de este punto consignada en la Figura 6 realizaremos simulaciones, en ese sentido, la Figura 7 expone dichas simulaciones, en total se realizaron 100 simulaciones para las políticas determinadas por ambos métodos, en dichas simulaciones se observa que la estela dejada por los recorridos de la política obtenida por Value Iteration es menos gruesa que la generada por los recorridos de la política generada por Programación Lineal, es más, algunos recorridos inducidos por la política generada por programación lineal pasan por lugares en la retícula que no son visitados por los recorridos generados por la política relacionada a Value Iteration, además cerca a la meta el comportamiento de los recorridos inducidos por Programación Lineal son un poco mas caóticos. En conclusión, dado que el objetivo es minimizar el tiempo de llegada que a su vez se traduce en recorridos de menor longitud se puede afirmar que Value Iteration produce políticas que se ajustan mas a dicho propósito, no obstante, se debe hacer la salvedad que se esta comparando una política determinística con una no determinística, puede que dicha comparación no sea equitativa o justa.

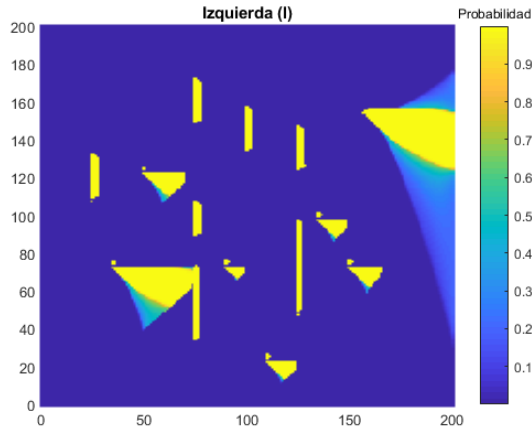
Las simulaciones en las imágenes (b) y (d) de la Figura fueron generadas mediante las líneas de código expuestas en el Algoritmo 6, este algoritmo es idéntico en código y funcionalidad al Algoritmo 3, la única diferencia es que el Algoritmo 6 emplea el Algoritmo 5 que crea la simulación de un recorrido del avión partiendo desde un punto inicial y para una política no determinista dada por el método de Linear Programming.



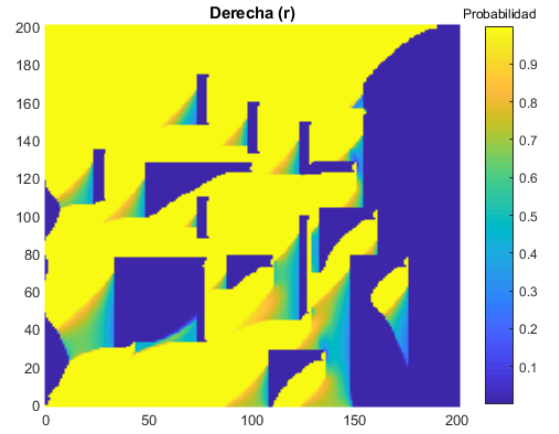
(a)



(b)



(c)



(d)

Figura 6: Política no determinística asociada a la solución del problema dual (3) para $\lambda = 0,99$ y $\alpha = \frac{1}{|S|}\mathbb{1}_{|S|}$. Las imágenes representan las probabilidades de tomar la acción a estando en el estado s para (a) $a = u$ (b) $a = d$ (c) $a = l$ y (d) $a = r$. En azul la menor probabilidad y en amarillo la mayor.

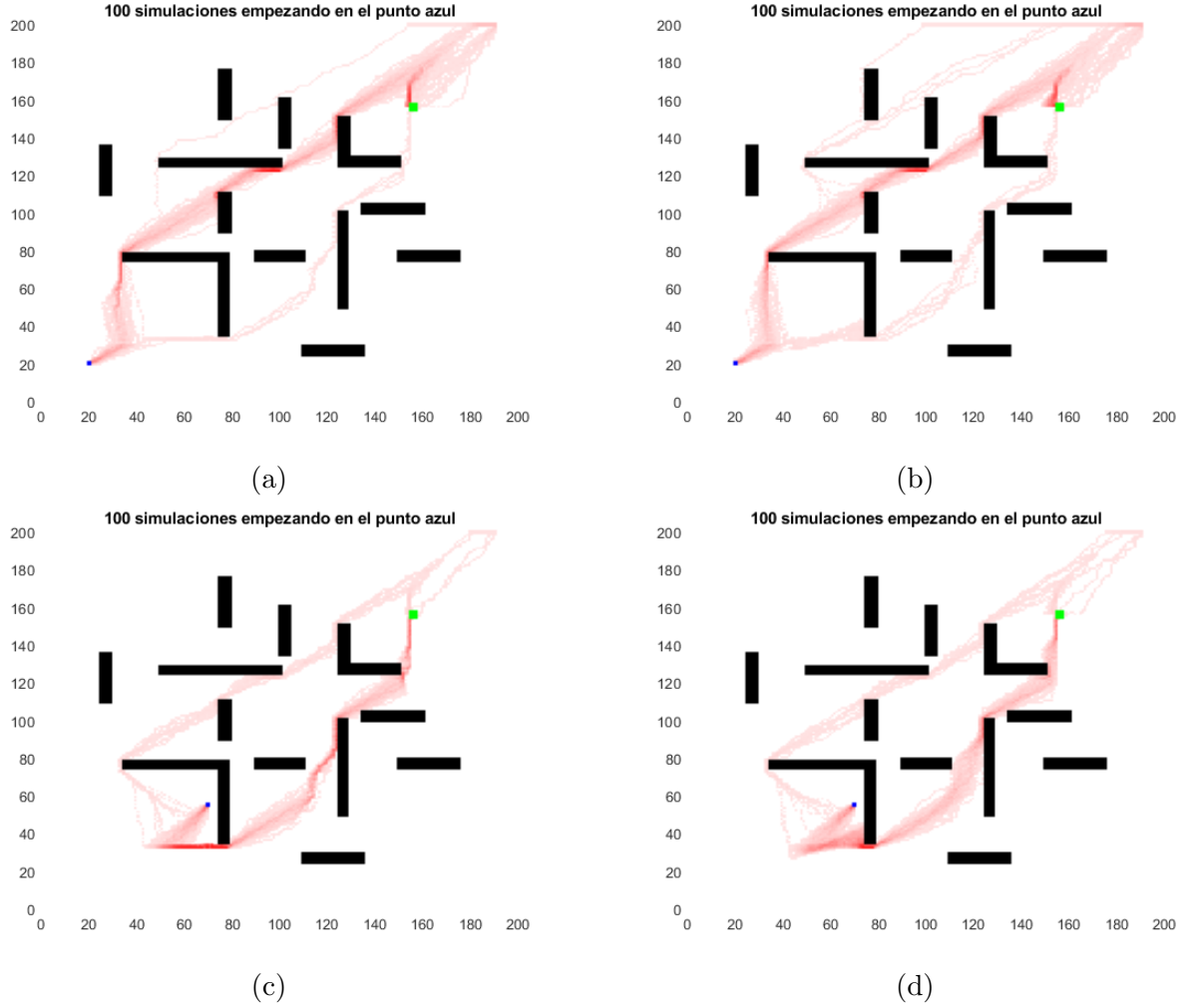


Figura 7: Simulación de 100 recorridos del avión regido por la política de la Figura 5 y la de la Figura 6 partiendo del punto (a) (20, 20) con Value Iteration (b) (20, 20) mediante Programación Lineal (c) (70, 55) con Value Iteration y (d) (70, 55) mediante Programación Lineal. En azul el punto de partida y en verde la región de llegada.

¿Puede encontrar una política determinística? Para este caso no fue posible encontrar una política determinística debido a las limitantes computacionales, el solver de MatLab utilizado fue `LinProg`, este permite emplear dos algoritmos que son `Punto Interior` y `Dual-Simplex`, este último es una versión mejorada del tradicional método de Simplex, es sabido que el método de Simplex entrega puntos óptimos que se encuentran en los vértices del polítopo convexo que determina la restricción del

problema dual (3), estas soluciones son las únicas que nos permiten recrear políticas determinísticas, pero dicho método requiere de tener un vértice para poder empezar a ejecutarse, ya que la matriz del problema es de gran escala y fue almacenada de manera sparse el algoritmo no pudo encontrar puntos factibles ubicados en los vertices.

¿Qué pasa cuando cambia la distribución inicial?


La respuesta es que la política óptima (que en nuestro caso es no determinista) no cambia, para verificarlo se realizo un experimento cambiando α , el nuevo α se define a partir del vector

$$\kappa := [\mathbb{1}_{50(201)}, 11\mathbb{1}_{50(201)}, \mathbb{1}_{101(201)}]$$

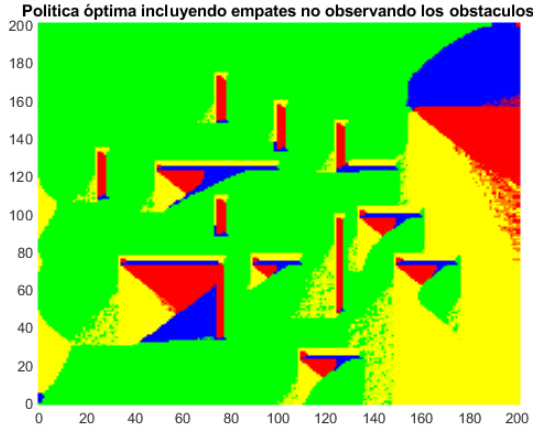
donde $\mathbb{1}_m$ es un vector de unos de longitud m . A partir de este vector κ se consideró $\alpha = \frac{\kappa}{\|\kappa\|}$.

La política óptima para este α y $\lambda = 0,99$ es igual a la obtenida en la Figura 6, se observo detalle por detalle y no se evidencia cambio alguno.

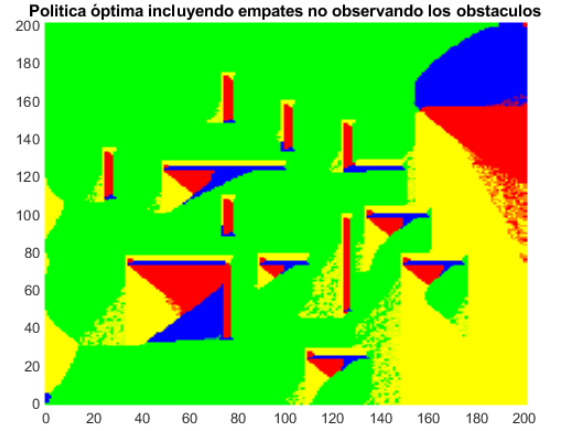
- ④ Para esta parte del trabajo elegimos realizar dicho análisis empleando el método de **Value Iteration**, la razón de su elección es que este método permitió obtener políticas deterministas, esto es importante ya que en el caso de políticas generadas por Programación Lineal estas no son cómodas de representar gráficamente debido al hecho de no ser determinista y se requieren de 4 imágenes para visualizarlas, una por cada acción.

En la Figura 8 se observa que la política aparentemente no tiene cambios tan evidentes, esto ocurre aproximadamente a partir de $\lambda \geq 0,986 =: \lambda^*$, en dicha figura se consideran las políticas exponiendo los empates, ya que si dicho análisis se realiza con la política entregada por default por nuestro algoritmo de Value Iteration no se podrá evidenciar dicho patrón. Note que observando minuciosamente las políticas para $\lambda \geq 0,986$ a simple vista comparten las regiones donde se presentan empates, es decir, las imágenes en la Figura 8 se puede interpretar como un conjunto de políticas debido a los empates, es decir, para cada λ su imagen correspondiente en la Figura 8 es su correspondiente conjunto de políticas, a partir de lo observado una primera impresión podría ser que que dichos conjuntos no son disjuntos, uno de los detalles que permite intuirlo es que el color rojo  en el estado $s = (200, 200)$ desaparece a partir de $\lambda \geq 0,986$ y además se preserva la forma, lo que permite intuir la existencia de una política de Blackwell, sin embargo, esto solo esta basado en una apreciación visual, puede que en realidad el verdadero λ^* que caracteriza las políticas de Blackwell sea mas cercano a uno que el propuesto hasta el momento, es más, si realizamos un acercamiento comparando estado por estado siempre encontraremos una diferencia.

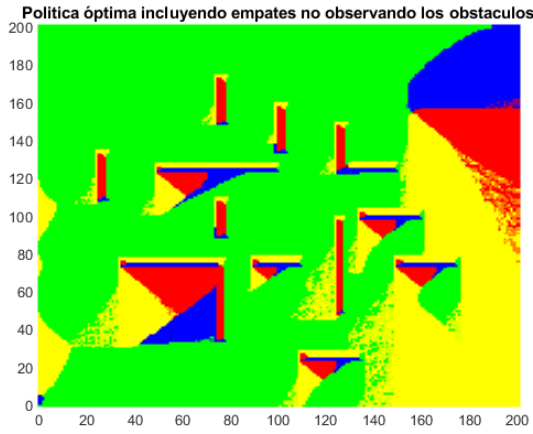
La imposibilidad de determinar la política de Blackwell es una situación que era muy probable ya que la política entregada por el algoritmo de Value Iteration no necesariamente es la de Blackwell, en realidad no se conoce un resultado teórico que permita garantizar que el algoritmo entregue dicha política tan especial.



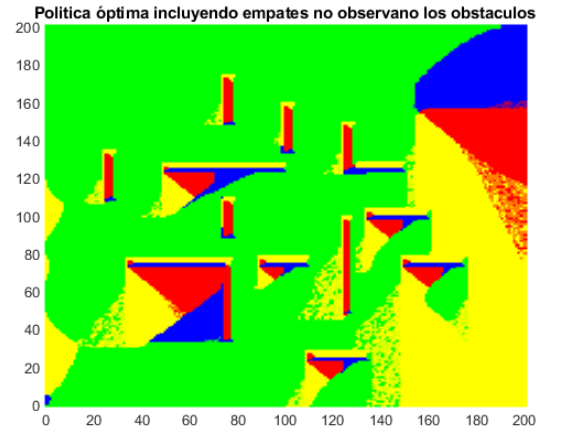
(a)



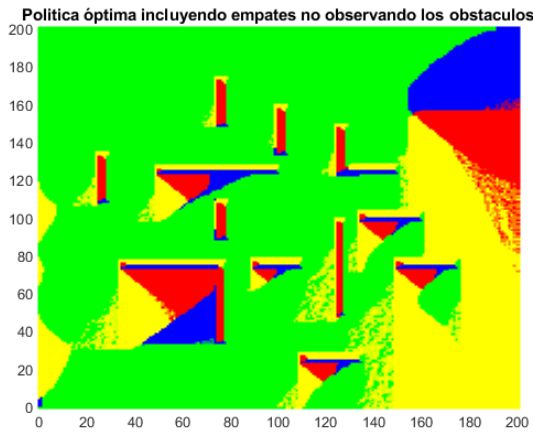
(b)



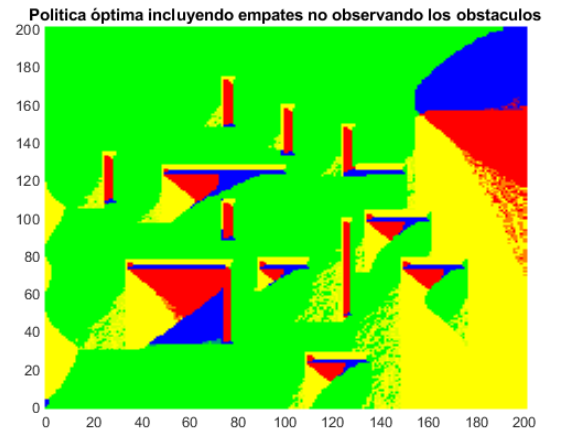
(c)



(d)



(e)



(f)

Figura 8: Políticas obtenidas mediante Value Iteration (con empates) con ε y (a) $\lambda = 0,984$ (b) $\lambda = 0,985$ (c) $\lambda = 0,986$ (d) $\lambda = 0,99$ (e) $\lambda = 0,995$ y (f) $\lambda = 0,999$.



Apéndice: Códigos

Algoritmo 1: Algoritmo de Value Iteration.

```
1 function [politica, V, AllPolice] = ValueIterationAvion(lambda, epsilon)
2 tic
3 %Estados S en coordenadas X y Y
4 X = repmat(0:1:200, 1, 201);
5 Y = reshape(repmat(0:1:200, 201, 1), [1, 40401]);
6 S = [X; Y];
7 %Coordenadas de los Obstaculos
8 YObsta = [repmat(110:1:135, 1, 5), repmat(25:1:29, 1, 26), repmat(90:1:110, 1, 5),
9           repmat(75:1:79, 1, 21), repmat(125:1:128, 1, 51), repmat(50:1:100, 1, 4), repmat
10            (150:1:175, 1, 5), repmat(75:1:79, 1, 26), repmat(135:1:160, 1, 5), repmat
11            (100:1:104, 1, 26), repmat(35:1:78, 1, 4), repmat(75:1:78, 1, 40), repmat
12            (125:1:129, 1, 26), repmat(130:1:150, 1, 5)];
13 XObsta = 200 - [reshape(repmat(171:1:175, 26, 1), [1, 130]), reshape(repmat
14                      (65:1:90, 5, 1), [1, 130]), reshape(repmat(121:1:125, 21, 1), [1, 105]), reshape(
15                      repmat(90:1:110, 5, 1), [1, 105]), reshape(repmat(100:1:150, 4, 1), [1, 204]),
16                      reshape(repmat(72:1:75, 51, 1), [1, 204]), reshape(repmat(121:1:125, 26, 1),
17                      [1, 130]), reshape(repmat(25:1:50, 5, 1), [1, 130]), reshape(repmat
18                      (96:1:100, 26, 1), [1, 130]), reshape(repmat(40:1:65, 5, 1), [1, 130]), reshape(
19                      repmat(122:1:125, 44, 1), [1, 176]), reshape(repmat(126:1:165, 4, 1), [1, 160]),
20                      reshape(repmat(50:1:75, 5, 1), [1, 130]), reshape(repmat(71:1:75, 21, 1),
21                      [1, 105])];
22 Obsta = [XObsta; YObsta];
23 PosObst = (ismember(S', Obsta', 'rows'))'; %Posición de los puntos de los
24          obstaculos
25 %Coordenadas de la Meta
26 CoordMeta = [repmat(155:1:157, 1, 3); reshape(repmat(155:1:157, 3, 1), [1, 9])];
27 PosMeta = (ismember(S', CoordMeta', 'rows'))'; %Posiciones de los puntos de
28          la meta
29 [m, L] = size(S); %Note que m=2.
30 %Definimos la recompensa
31 RenObs = PosObst * (-100000000); %Recompensa de estar en los obstaculos
32 RenEstad = ones(1, L); %Recompensa de estar en cualquier estado
33 RenMeta = PosMeta * (10000); %Recompensa de estar en la meta
34 RencomAux = RenObs + RenEstad + RenMeta; %Esta es la función h del documento.
35 RencomAuxMat = reshape(RencomAux, [201, 201])'; %Esta es la función h del
36          documento en forma matricial.
37 %Para u
38 RencomU1Mat = [[2*RencomAuxMat(2:1:200, 1); 4; 4], [RencomAuxMat
39              (2:1:200, 2:1:201); 2*RencomAuxMat(201, 2:1:201); 2*RencomAuxMat
40              (201, 2:1:201)]];
41 RencomU2Mat = [[2*RencomAuxMat(3:1:201, 1), RencomAuxMat(3:1:201, 2:1:201)
42              ]; zeros(1, 201); zeros(1, 201)];
```

```

29 RencomU3Mat=[zeros(201,1),[RencomAuxMat(3:1:201,1:1:200);zeros(1,200);
    zeros(1,200)]];
30 RencomU4Mat=[zeros(201,1),[RencomAuxMat(2:1:200,1:1:200);2*
    RencomAuxMat(201,1:1:200);2*RencomAuxMat(201,1:1:200)]];
31 RencomUMat=RencomU1Mat+RencomU2Mat+RencomU3Mat+RencomU4Mat;
32 RecomU=reshape(RencomUMat',[1,201*201]);
33 %Para d
34 RencomD1Mat=[[4;2*RencomAuxMat(2:1:201,1)],[2*RencomAuxMat(1,2:1:201);
    RencomAuxMat(2:1:201,2:1:201)]];
35 RencomD2Mat=[zeros(1,201);[2*RencomAuxMat(1:1:200,1),RencomAuxMat
    (1:1:200,2:1:201)]];
36 RencomD3Mat=[zeros(201,1),[zeros(1,200);RencomAuxMat(1:1:200,1:1:200)
    ]];
37 RencomD4Mat=[zeros(201,1),[2*RencomAuxMat(1,1:1:200);RencomAuxMat
    (2:1:201,1:1:200)]];
38 RencomDMat=RencomD1Mat+RencomD2Mat+RencomD3Mat+RencomD4Mat;
39 RecomD=reshape(RencomDMat',[1,201*201]);
40 %Para L
41 RencomL1Mat=[[2*RencomAuxMat(1:1:200,1);4],[2*RencomAuxMat(1:1:200,1)
    ;4]], [RencomAuxMat(1:1:200,2:1:200);2*RencomAuxMat(201,2:1:200)]];
42 RencomL2Mat=[[2*RencomAuxMat(2:1:201,1),RencomAuxMat(2:1:201,2:1:201)
    ];zeros(1,201)];
43 RencomL3Mat=[zeros(201,1),zeros(201,1),[RencomAuxMat(2:1:201,1:1:199);
    zeros(1,199)];
44 RencomL4Mat=[zeros(201,1),zeros(201,1),[RencomAuxMat(1:1:200,1:1:199)
    ;2*RencomAuxMat(201,1:1:199)]];
45 RencomLMat=RencomL1Mat+RencomL2Mat+RencomL3Mat+RencomL4Mat;
46 RecomL=reshape(RencomLMat',[1,201*201]);
47 %Para R
48 RencomR1Mat=[[RencomAuxMat(1:1:200,1:1:200);2*RencomAuxMat
    (201,1:1:200)],[2*RencomAuxMat(1:1:200,201);4]];
49 RencomR2Mat=[[RencomAuxMat(2:1:201,1:1:200),2*RencomAuxMat
    (2:1:201,201)];zeros(1,201)];
50 RencomR3Mat=[[RencomAuxMat(2:1:201,2:1:201);zeros(1,200)],zeros(201,1)
    ];
51 RencomR4Mat=[[RencomAuxMat(1:1:200,2:1:201);2*RencomAuxMat
    (201,2:1:201)],zeros(201,1)];
52 RencomRMat=RencomR1Mat+RencomR2Mat+RencomR3Mat+RencomR4Mat;
53 RecomR=reshape(RencomRMat',[1,201*201]);
54 %Matriz de recompensas, la filas representa acciones y las columnas
55 %los estados
56 Rencom=[RecomU;RecomD;RecomL;RecomR];
57
58 V=ones(1,L); %Función de valor inicial en forma de vector v^{n+1}
59 Vante=(2+epsilon*(1-lambda)/(2*lambda))*ones(1,L); %Funcion de valor
    anterior v^n}
60 Vmat=reshape(V,[201,201])'; %El vector V en forma de matriz.
61 while max(abs(V-Vante))> epsilon*(1-lambda)/(2*lambda)
62     %Extraemos los estados que tiene probabilidad no cero esto para cada
63     %acción y cada estado

```

```

64 %Para u (upper)
65 Probau1=0.3*ones(201,201);Probau1(:,1)=0.4*ones(1,201)';Probau1(200,:)=
    =0.7*ones(1,201);Probau1(201,:)=0.7*ones(1,201);Probau1
    (200:1:201,1)=1;
66 Vu1mat=zeros(201,201);Vu1mat(201,:)=Vmat(201,:);Vu1mat(1:1:200,:)=Vmat
    (2:1:201,:);
67 Vu1p=Vu1mat.*Probau1;
68 Vu1=reshape(Vu1p',[1,201*201]);
69 %Para u2
70 Probau2=0.4*ones(201,201);Probau2(:,1)=0.6*ones(1,201)';Probau2(200,:)=
    =zeros(1,201);Probau2(201,:)=zeros(1,201);
71 Vu2mat=zeros(201,201);Vu2mat(201,:)=Vmat(201,:);Vu2mat(200,:)=Vmat
    (201,:);Vu2mat(1:1:199,:)=Vmat(3:1:201,:);
72 Vu2p=Vu2mat.*Probau2;
73 Vu2=reshape(Vu2p',[1,201*201]);
74 %Para u3
75 Probau3=0.2*ones(201,201);Probau3(:,1)=zeros(1,201);Probau3(201,:)=
    =zeros(1,201);Probau3(200,:)=zeros(1,201);
76 Vu3mat=zeros(201,201);Vu3mat(201,:)=Vmat(201,:);Vu3mat(200,:)=Vmat
    (201,:);Vu3mat(:,1)=Vu3mat(201,:);Vu3mat(1:1:199,2:1:201)=Vmat
    (3:1:201,1:1:200);
77 Vu3p=Vu3mat.*Probau3;
78 Vu3=reshape(Vu3p',[1,201*201]);
79 %Para u4
80 Probau4=0.1*ones(201,201);Probau4(201,:)=0.3*ones(1,201);Probau4
    (200,:)=0.3*ones(1,201);Probau4(:,1)=zeros(1,201)';
81 Vu4mat=zeros(201,201);Vu4mat(201,2:1:201)=Vmat(201,1:1:200);Vu4mat
    (:,1)=Vmat(201,:);Vu4mat(1:1:200,2:1:201)=Vmat(2:1:201,1:1:200);
82 Vu4p=Vu4mat.*Probau4;
83 Vu4=reshape(Vu4p',[1,201*201]);
84 %Definimos vu como
85 Vu=Vu1+Vu2+Vu3+Vu4;
86 %Para d (down)
87 Probad1=0.3*ones(201,201);Probad1(:,1)=0.5*ones(201,1);Probad1(1,:)=
    =0.6*ones(1,201);Probad1(1,1)=1;
88 Vd1mat=Vmat;
89 Vd1p=Vd1mat.*Probad1;
90 Vd1=reshape(Vd1p',[1,201*201]);
91 %Para d2
92 Probad2=0.3*ones(201,201);Probad2(:,1)=0.5*ones(201,1);Probad2(1,:)=
    =zeros(1,201);
93 Vd2mat=zeros(201,201);Vd2mat(1,:)=Vmat(1,:);Vd2mat(2:1:201,:)=Vmat
    (1:1:200,:);
94 Vd2p=Vd2mat.*Probad2;
95 Vd2=reshape(Vd2p',[1,201*201]);
96 %Para d3
97 Probad3=0.2*ones(201,201);Probad3(:,1)=zeros(1,201);Probad3(1,:)=zeros
    (1,201);
98 Vd3mat=zeros(201,201);Vd3mat(1,:)=Vmat(1,:);Vd3mat(:,1)=Vmat(1,:);
    Vd3mat(2:1:201,2:1:201)=Vmat(1:1:200,1:1:200);

```

```

99     Vd3p=Vd3mat.*Probad3;
100     Vd3=reshape(Vd3p',[1,201*201]);
101     %Para d4
102     Probad4=0.2*ones(201,201);Probad4(1,:)=0.4*ones(1,201);Probad4(:,1)=
        zeros(1,201)';
103     Vd4mat=zeros(201,201);Vd4mat(1,:)=Vmat(1,:);Vd4mat(:,1)=Vmat(1,:)';
        Vd4mat(:,2:1:201)=Vmat(:,1:1:200);
104     Vd4p=Vd4mat.*Probad4;
105     Vd4=reshape(Vd4p',[1,201*201]);
106     %Definimos vd como
107     Vd=Vd1+Vd2+Vd3+Vd4;
108     %Para L (upper)
109     ProbaL1=0.2*ones(201,201);ProbaL1(:,1)=0.5*ones(201,1);ProbaL1(:,2)
        =0.5*ones(201,1);ProbaL1(201,:)=0.5*ones(1,201);ProbaL1(201,1:1:2)
        =1;
110     VL1mat=zeros(201,201);VL1mat(:,1)=Vmat(:,1);VL1mat(:,2:1:201)=Vmat
        (:,1:1:200);
111     VL1p=VL1mat.*ProbaL1;
112     VL1=reshape(VL1p',[1,201*201]);
113     %Para L2
114     ProbaL2=0.3*ones(201,201);ProbaL2(:,1)=0.5*ones(201,1);ProbaL2(:,2)
        =0.5*ones(201,1);ProbaL2(201,:)=zeros(1,201);
115     VL2mat=zeros(201,201);VL2mat(1:1:200,2:1:201)=Vmat(2:1:201,1:1:200);
        VL2mat(1:1:200,1)=Vmat(2:1:201,1);VL2mat(201,:)=Vmat(201,:);
116     VL2p=VL2mat.*ProbaL2;
117     VL2=reshape(VL2p',[1,201*201]);
118     %Para L3
119     ProbaL3=0.2*ones(201,201);ProbaL3(201,:)=zeros(1,201);ProbaL3(:,1)=
        zeros(201,1);ProbaL3(:,2)=zeros(201,1);
120     VL3mat=zeros(201,201);VL3mat(:,1)=Vmat(:,1);VL3mat(:,2)=Vmat(:,201);
        VL3mat(201,:)=VL3mat(:,201)';VL3mat(1:1:200,3:1:201)=Vmat
        (2:1:201,1:1:199);
121     VL3p=VL3mat.*ProbaL3;
122     VL3=reshape(VL3p',[1,201*201]);
123     %Para L4
124     ProbaL4=0.3*ones(201,201);ProbaL4(201,:)=0.5*ones(1,201);ProbaL4(:,1)=
        zeros(201,1);ProbaL4(:,2)=zeros(201,1);
125     VL4mat=zeros(201,201);VL4mat(:,1)=Vmat(:,1);VL4mat(:,2)=Vmat(:,2);
        VL4mat(:,3:1:201)=Vmat(:,1:1:199);
126     VL4p=VL4mat.*ProbaL4;
127     VL4=reshape(VL4p',[1,201*201]);
128     %Definimos vL como
129     VL=VL1+VL2+VL3+VL4;
130     %Para R (right)
131     ProbaR1=0.2*ones(201,201); ProbaR1(:,201)=0.5*ones(201,1);ProbaR1
        (201,:)=0.3*ones(1,201);ProbaR1(201,201)=1;
132     VR1mat=Vmat;
133     VR1p=VR1mat.*ProbaR1;
134     VR1=reshape(VR1p',[1,201*201]);
135     %Para R2

```



```

136     ProbaR2=0.1*ones(201,201);ProbaR2(:,201)=0.5*ones(201,1);ProbaR2
        (201,:)=zeros(1,201);
137     VR2mat=zeros(201,201);VR2mat(201,:)=Vmat(201,:);VR2mat(1:1:200,:)=Vmat
        (2:1:201,:);
138     VR2p=VR2mat.*ProbaR2;
139     VR2=reshape(VR2p',[1,201*201]);
140     %Para R3
141     ProbaR3=0.4*ones(201,201);ProbaR3(:,201)=zeros(1,201);ProbaR3(201,:)=
        zeros(1,201);
142     VR3mat=zeros(201,201);VR3mat(201,:)=Vmat(201,:);VR3mat(:,201)=Vmat
        (201,:);VR3mat(1:1:200,1:1:200)=Vmat(2:1:201,2:1:201);
143     VR3p=VR3mat.*ProbaR3;
144     VR3=reshape(VR3p',[1,201*201]);
145     %Para d4
146     ProbaR4=0.3*ones(201,201);ProbaR4(201,:)=0.7*ones(1,201);ProbaR4
        (:,201)=zeros(1,201)';
147     VR4mat=zeros(201,201);VR4mat(201,:)=Vmat(201,:);VR4mat(:,201)=Vmat
        (:,201);VR4mat(:,1:1:200)=Vmat(:,2:1:201);
148     VR4p=VR4mat.*ProbaR4;
149     VR4=reshape(VR4p',[1,201*201]);
150     %Definimos vR como
151     VR=VR1+VR2+VR3+VR4;
152
153     QV=[Vu;Vd;VL;VR];
154     Vante=V;
155     RencQV=Rencom+lambda*QV; %Esta es la matriz del paso 2 del algoritmo.
156     [V,posMax]=max(RencQV);
157     Vmat=reshape(V,[201,201])'; %V en forma de matriz
158 end
159
160 %Extraemos la politica que basicamente es realizar otra iteración del While
161 %anterior
162 %De nuevo extraemos los estados que tiene probabilidad no cero esto para
    cada
163 %acción y cada estado
164 %Para u (upper)
165     Probau1=0.3*ones(201,201);Probau1(:,1)=0.4*ones(1,201)';Probau1(200,:)
        =0.7*ones(1,201);Probau1(201,:)=0.7*ones(1,201);Probau1
        (200:1:201,1)=1;
166     Vu1mat=zeros(201,201);Vu1mat(201,:)=Vmat(201,:);Vu1mat(1:1:200,:)=Vmat
        (2:1:201,:);
167     Vu1p=Vu1mat.*Probau1;
168     Vu1=reshape(Vu1p',[1,201*201]);
169     %Para u2
170     Probau2=0.4*ones(201,201);Probau2(:,1)=0.6*ones(1,201)';Probau2(200,:)
        =zeros(1,201);Probau2(201,:)=zeros(1,201);
171     Vu2mat=zeros(201,201);Vu2mat(201,:)=Vmat(201,:);Vu2mat(200,:)=Vmat
        (201,:);Vu2mat(1:1:199,:)=Vmat(3:1:201,:);
172     Vu2p=Vu2mat.*Probau2;
173     Vu2=reshape(Vu2p',[1,201*201]);

```

```

174 %Para u3
175 Probau3=0.2*ones(201,201);Probau3(:,1)=zeros(1,201);Probau3(201,:)=
    zeros(1,201);Probau3(200,:)=zeros(1,201);
176 Vu3mat=zeros(201,201);Vu3mat(201,:)=Vmat(201,:);Vu3mat(200,:)=Vmat
    (201,:);Vu3mat(:,1)=Vu3mat(201,:);Vu3mat(1:1:199,2:1:201)=Vmat
    (3:1:201,1:1:200);
177 Vu3p=Vu3mat.*Probau3;
178 Vu3=reshape(Vu3p',[1,201*201]);
179 %Para u4
180 Probau4=0.1*ones(201,201);Probau4(201,:)=0.3*ones(1,201);Probau4
    (200,:)=0.3*ones(1,201);Probau4(:,1)=zeros(1,201);
181 Vu4mat=zeros(201,201);Vu4mat(201,2:1:201)=Vmat(201,1:1:200);Vu4mat
    (:,1)=Vmat(201,:);Vu4mat(1:1:200,2:1:201)=Vmat(2:1:201,1:1:200);
182 Vu4p=Vu4mat.*Probau4;
183 Vu4=reshape(Vu4p',[1,201*201]);
184 %Definimos vu como
185 Vu=Vu1+Vu2+Vu3+Vu4;
186 %Para d (down)
187 Probad1=0.3*ones(201,201); Probad1(:,1)=0.5*ones(201,1);Probad1(1,:)=
    0.6*ones(1,201);Probad1(1,1)=1;
188 Vd1mat=Vmat;
189 Vd1p=Vd1mat.*Probad1;
190 Vd1=reshape(Vd1p',[1,201*201]);
191 %Para d2
192 Probad2=0.3*ones(201,201);Probad2(:,1)=0.5*ones(201,1);Probad2(1,:)=
    zeros(1,201);
193 Vd2mat=zeros(201,201);Vd2mat(1,:)=Vmat(1,:);Vd2mat(2:1:201,:)=Vmat
    (1:1:200,:);
194 Vd2p=Vd2mat.*Probad2;
195 Vd2=reshape(Vd2p',[1,201*201]);
196 %Para d3
197 Probad3=0.2*ones(201,201);Probad3(:,1)=zeros(1,201);Probad3(1,:)=zeros
    (1,201);
198 Vd3mat=zeros(201,201);Vd3mat(1,:)=Vmat(1,:);Vd3mat(:,1)=Vmat(1,:);
    Vd3mat(2:1:201,2:1:201)=Vmat(1:1:200,1:1:200);
199 Vd3p=Vd3mat.*Probad3;
200 Vd3=reshape(Vd3p',[1,201*201]);
201 %Para d4
202 Probad4=0.2*ones(201,201);Probad4(1,:)=0.4*ones(1,201);Probad4(:,1)=
    zeros(1,201);
203 Vd4mat=zeros(201,201);Vd4mat(1,:)=Vmat(1,:);Vd4mat(:,1)=Vmat(1,:);
    Vd4mat(:,2:1:201)=Vmat(:,1:1:200);
204 Vd4p=Vd4mat.*Probad4;
205 Vd4=reshape(Vd4p',[1,201*201]);
206 %Definimos vd como
207 Vd=Vd1+Vd2+Vd3+Vd4;
208 %Para L (upper)
209 ProbaL1=0.2*ones(201,201);ProbaL1(:,1)=0.5*ones(201,1);ProbaL1(:,2)
    =0.5*ones(201,1);ProbaL1(201,:)=0.5*ones(1,201);ProbaL1(201,1:1:2)
    =1;

```

```

210 VL1mat=zeros(201,201);VL1mat(:,1)=Vmat(:,1);VL1mat(:,2:1:201)=Vmat
    (:,1:1:200);
211 VL1p=VL1mat.*ProbaL1;
212 VL1=reshape(VL1p',[1,201*201]);
213 %Para L2
214 ProbaL2=0.3*ones(201,201);ProbaL2(:,1)=0.5*ones(201,1);ProbaL2(:,2)
    =0.5*ones(201,1);ProbaL2(201,:)=zeros(1,201);
215 VL2mat=zeros(201,201);VL2mat(1:1:200,2:1:201)=Vmat(2:1:201,1:1:200);
    VL2mat(1:1:200,1)=Vmat(2:1:201,1);VL2mat(201,:)=Vmat(201,:);
216 VL2p=VL2mat.*ProbaL2;
217 VL2=reshape(VL2p',[1,201*201]);
218 %Para L3
219 ProbaL3=0.2*ones(201,201);ProbaL3(201,:)=zeros(1,201);ProbaL3(:,1)=
    zeros(201,1);ProbaL3(:,2)=zeros(201,1);
220 VL3mat=zeros(201,201);VL3mat(:,1)=Vmat(:,1);VL3mat(:,2)=Vmat(:,201);
    VL3mat(201,:)=VL3mat(:,201)';VL3mat(1:1:200,3:1:201)=Vmat
    (2:1:201,1:1:199);
221 VL3p=VL3mat.*ProbaL3;
222 VL3=reshape(VL3p',[1,201*201]);
223 %Para L4
224 ProbaL4=0.3*ones(201,201);ProbaL4(201,:)=0.5*ones(1,201);ProbaL4(:,1)=
    zeros(201,1);ProbaL4(:,2)=zeros(201,1);
225 VL4mat=zeros(201,201);VL4mat(:,1)=Vmat(:,1);VL4mat(:,2)=Vmat(:,2);
    VL4mat(:,3:1:201)=Vmat(:,1:1:199);
226 VL4p=VL4mat.*ProbaL4;
227 VL4=reshape(VL4p',[1,201*201]);
228 %Definimos vL como
229 VL=VL1+VL2+VL3+VL4;
230 %Para R (right)
231 ProbaR1=0.2*ones(201,201); ProbaR1(:,201)=0.5*ones(201,1);ProbaR1
    (201,:)=0.3*ones(1,201);ProbaR1(201,201)=1;
232 VR1mat=Vmat;
233 VR1p=VR1mat.*ProbaR1;
234 VR1=reshape(VR1p',[1,201*201]);
235 %Para R2
236 ProbaR2=0.1*ones(201,201);ProbaR2(:,201)=0.5*ones(201,1);ProbaR2
    (201,:)=zeros(1,201);
237 VR2mat=zeros(201,201);VR2mat(201,:)=Vmat(201,:);VR2mat(1:1:200,:)=Vmat
    (2:1:201,:);
238 VR2p=VR2mat.*ProbaR2;
239 VR2=reshape(VR2p',[1,201*201]);
240 %Para R3
241 ProbaR3=0.4*ones(201,201);ProbaR3(:,201)=zeros(1,201);ProbaR3(201,:)=
    zeros(1,201);
242 VR3mat=zeros(201,201);VR3mat(201,:)=Vmat(201,:);VR3mat(:,201)=Vmat
    (201,:);VR3mat(1:1:200,1:1:200)=Vmat(2:1:201,2:1:201);
243 VR3p=VR3mat.*ProbaR3;
244 VR3=reshape(VR3p',[1,201*201]);
245 %Para d4

```

```

246     ProbaR4=0.3*ones(201,201);ProbaR4(201,:)=0.7*ones(1,201);ProbaR4
        (:,201)=zeros(1,201)';
247     VR4mat=zeros(201,201);VR4mat(201,:)=Vmat(201,:);VR4mat(:,201)=Vmat
        (:,201);VR4mat(:,1:1:200)=Vmat(:,2:1:201);
248     VR4p=VR4mat.*ProbaR4;
249     VR4=reshape(VR4p',[1,201*201]);
250     %Definimos vR como
251     VR=VR1+VR2+VR3+VR4;
252
253     QV=[Vu;Vd;VL;VR];
254     RencQV=Rencom+lambda*QV;
255     [Vaux,posMax]=max(RencQV); %Solo nos interesa la posición del maximo,
        pueden haber empates
256
257     politica=posMax; %Ojo, esta es solo una de las politicas, hay mas y
        todas estaran en 'Allpolice'.
258
259     %Para sacar empates, es decir, todas las posibles politicas
260     [accionEmp,EstadoEmp]=find(RencQV == Vaux); %Encuentra todas las
        acciones que alcanzan el maximo.
261     AllPolice=zeros(4,L); %Aqui guardamos todas las politicas, cada
        fila es una acción y las columnas son los estados
262     AllPolice(sub2ind(size(AllPolice),accionEmp',EstadoEmp'))=1; %Si en
        columna 's' y fila 'a' sale 1 entonces se puede tomar esa acción
        a en el estado s
263     EstSinEmp=(sum(AllPolice)==1); %Determina los estados que no tiene
        empates
264     EstConEmp=(sum(AllPolice)~=1); %Determina los estados que si tiene
        empates
265     AllPolice(2,:)=2*AllPolice(2,:); % Bajar (down)
266     AllPolice(3,:)=3*AllPolice(3,:); %Izquierda (left)
267     AllPolice(4,:)=4*AllPolice(4,:); % Derecha (Right)
268     %El siguinete vector tiene los valores 1,2,3,4,5,6,7,8,9,11, lo
269     %usamos para poder darle colores a los resultados.
270     EstadEmpatados= sum(AllPolice.*EstSinEmp)+sum(AllPolice.*EstConEmp)+
        min( (AllPolice +(AllPolice==0).*((max(AllPolice)+1).*EstConEmp)
        ).*EstConEmp)+EstConEmp;
271     ColorCrd1=(EstadEmpatados==1)'+(EstadEmpatados==3)'+(EstadEmpatados
        ==6)'+(0.85)*(EstadEmpatados==7)'+(0.5)*(EstadEmpatados==5)
        '+ (0.5)*(EstadEmpatados==8)'+(0.6)*(EstadEmpatados==11)';
272     ColorCrd2=(EstadEmpatados==1)'+(EstadEmpatados==4)'+(0.5)*(
        EstadEmpatados==6)'+(EstadEmpatados==7)'+(0.5)*(EstadEmpatados
        ==5)'+(0.6)*(EstadEmpatados==9)'+(0.5)*(EstadEmpatados==11)';
273     ColorCrd3=(EstadEmpatados==2)'+(0.5)*(EstadEmpatados==5)'+(0.5)*(
        EstadEmpatados==8)'+(0.5)*(EstadEmpatados==9)';
274     ColoresEstadEmpatados=[ColorCrd1,ColorCrd2,ColorCrd3];
275     %***Finaliza la busqueda de la politica
276     toc
277

```

```

278 %IMPORTANTE: Si va a usar 'simulacionAvion.m' ó 'SombrasSimulaciones.m'
      convierta en comentario todo lo que sigue a continuación:
279
280 %Graficamos la politica, todas las politicas con empates y la función de
      valor:
281 %%{
282 PolitSinObsta=(posMax.*(abs(1-PosObst)))-(posMax.*PosMeta)+5*PosMeta; %
      Para graficar sin obstaculos ni meta
283 ColoresSinObsta=[(PolitSinObsta==1)'+(PolitSinObsta==3)'+(PolitSinObsta
      ==5)',(PolitSinObsta==1)'+(PolitSinObsta==4)'+(PolitSinObsta==5)',(
      PolitSinObsta==2)'+(PolitSinObsta==5)'];
284 ColoresConObsta=[(posMax==1)'+(posMax==3)',(posMax==1)'+(posMax==4)',(
      posMax==2)'];
285 pointsize = 20;
286 figure(2)
287 scatter(X, Y, pointsize, ColoresSinObsta,'filled','s');
288 title('Politica óptima observano los obstaculos')
289 figure(3)
290 scatter(X, Y, pointsize, ColoresConObsta,'filled','s');
291 title('Politica óptima no observando los obstaculos')
292 figure(4)
293 scatter(X, Y, pointsize, V,'filled','s'); %Grafica de la función de
      valor
294 title('Valor en cada estado')
295 figure(5)
296 scatter(X, Y, pointsize, ColoresEstadEmpatados,'filled','s');
297 title('Politica óptima incluyendo empates no observano los obstaculos')
298 figure(6)
299 scatter(X, Y, pointsize, ColoresEstadEmpatados,'filled','s');
300 title('Politica óptima incluyendo empates observano los obstaculos')
301 hold on
302 scatter(XObsta, YObsta, pointsize,[0,0,0],'filled','s'); %Grafica los
      obstaculos solamente
303 hold off
304 %}

```

Algoritmo 2: Algoritmo que crea la simulación de un recorrido del avión partiendo desde un punto inicial y para una política dada.

```

1
2
3 function [VecRecorrido]=simulacionAvion(Pini,politica)
4 %Pini es el punto inicial del recorrido, este esta en forma horizontal
5
6
7 %Estados S en coordenadas X y Y
8 X=repmat(0:1:200,1,201);
9 Y=reshape(repmat(0:1:200,201,1),[1,40401]);
10 S=[X;Y];
11 %Coordenadas de los obstaculos

```

```

12 YObsta=[repmat(110:1:135,1,5),repmat(25:1:29,1,26),repmat(90:1:110,1,5),
    repmat(75:1:79,1,21),repmat(125:1:128,1,51),repmat(50:1:100,1,4),repmat
    (150:1:175,1,5),repmat(75:1:79,1,26),repmat(135:1:160,1,5),repmat
    (100:1:104,1,26),repmat(35:1:78,1,4),repmat(75:1:78,1,40),repmat
    (125:1:129,1,26),repmat(130:1:150,1,5)];
13 XObsta=200-[reshape(repmat(171:1:175,26,1),[1,130]),reshape(repmat
    (65:1:90,5,1),[1,130]),reshape(repmat(121:1:125,21,1),[1,105]),reshape(
    repmat(90:1:110,5,1),[1,105]),reshape(repmat(100:1:150,4,1),[1,204]),
    reshape(repmat(72:1:75,51,1),[1,204]),reshape(repmat(121:1:125,26,1)
    ,[1,130]),reshape(repmat(25:1:50,5,1),[1,130]),reshape(repmat
    (96:1:100,26,1),[1,130]),reshape(repmat(40:1:65,5,1),[1,130]),reshape(
    repmat(122:1:125,44,1),[1,176]),reshape(repmat(126:1:165,4,1),[1,160]),
    reshape(repmat(50:1:75,5,1),[1,130]),reshape(repmat(71:1:75,21,1)
    ,[1,105])];
14 Obsta=[XObsta;YObsta];
15 PosObst=(ismember(S',Obsta','rows'))'; %Posición de los obstaculos
16 %Coordenadas de la meta
17 CoordMeta=[repmat(155:1:157,1,3);reshape(repmat(155:1:157,3,1),[1,9])];
18 PosMeta=(ismember(S',CoordMeta','rows'))'; %Posición de los puntos de la
    meta en S
19
20 %Pini es un punto de la forma [a,b].
21 P=Pini'; %Queremos este punto en la forma vertical
22
23 Indic=ismember(P',CoordMeta','rows'); %Indica 1 si el punto esta en la
    meta y 0 si no.
24 VecRecorrido=P; %Aqui cuardamos los puntos por donde pasa el recorrido.
25 iter=0; %Para controlar.
26 while Indic~=1
27     iter=iter+1;
28     [Paux,Pos]=ismember(P',S','rows'); %Posición de P respecto a S
29     accion=politica(Pos); %Acción que le corresponde a P de acuerdo a
        la politica
30     if accion==1 %1 es u Subir
31         %Siguietes 4 lineas son para elegir uno de los puntos a dosnde se
32         %pude subir
33         Prob=[0.3,0.4,0.2,0.1];
34         C=cumsum(Prob);
35         Opci=[1,2,3,4];
36         Elecc= Opci(5-sum((C>=rand)));
37         %Los condicionales discriman el interior y las fronteras
38         if P(1)>=1 && P(2)==199
39             Puntos=[P+[0;1],P+[0;1],P+[-1;1],P+[-1;1]];
40             P=Puntos(:,Elecc);
41         elseif P(1)>=1 && P(2)==200
42             Puntos=[P,P,P+[-1;0],P+[-1;0]];
43             P=Puntos(:,Elecc);
44         elseif P(1)==0 && P(2)<=198
45             Puntos=[P+[0;1],P+[0;2],P+[0;2],P+[0;1]];
46             P=Puntos(:,Elecc);

```

```

47     elseif P(1)==0 && P(2)==199
48         Puntos=[P+[0;1],P+[0;1],P+[0;1],P+[0;1]];
49         P=Puntos(:,Elecc);
50     elseif P(1)==0 && P(2)==200
51         Puntos=[P,P,P,P];
52         P=Puntos(:,Elecc);
53     else
54         Puntos=[P+[0;1],P+[0;2],P+[-1;2],P+[-1;1]];
55         P=Puntos(:,Elecc);
56     end
57     %Los siguientes 'elseif' se describen analogo a como se describio
58     el
59     %caso 'u'
60 elseif accion==2 %2 es d bajar
61     Prob=[0.3,0.3,0.2,0.2];
62     C=cumsum(Prob);
63     Opci=[1,2,3,4];
64     Elecc= Opci(5-sum((C>=rand)));
65     if P(1)>=1 && P(2)==0
66         Puntos=[P,P,P+[-1;0],P+[-1;0]];
67         P=Puntos(:,Elecc);
68     elseif P(1)==0 && P(2)>=1
69         Puntos=[P,P+[0;-1],P+[0;-1],P];
70         P=Puntos(:,Elecc);
71     elseif P(1)==0 && P(2)==0
72         Puntos=[P,P,P,P];
73         P=Puntos(:,Elecc);
74     else
75         Puntos=[P,P+[0;-1],P+[-1;-1],P+[-1;0]];
76         P=Puntos(:,Elecc);
77     end
78 elseif accion==3 %3 es l izquierda
79     Prob=[0.2,0.3,0.2,0.3];
80     C=cumsum(Prob);
81     Opci=[1,2,3,4];
82     Elecc= Opci(5-sum((C>=rand)));
83     if P(1)>=2 && P(2)==200
84         Puntos=[P+[-1;0],P+[-1;0],P+[-2;0],P+[-2;0]];
85         P=Puntos(:,Elecc);
86     elseif P(1)==0 && P(2)==200
87         Puntos=[P,P,P,P];
88         P=Puntos(:,Elecc);
89     elseif P(1)==1 && P(2)==200
90         Puntos=[P+[-1;0],P+[-1;0],P+[-1;0],P+[-1;0]];
91         P=Puntos(:,Elecc);
92     elseif P(1)==0 && P(2)<=199
93         Puntos=[P,P+[0;1],P+[0;1],P];
94         P=Puntos(:,Elecc);
95     elseif P(1)==1 && P(2)<=199
96         Puntos=[P+[-1;0],P+[-1;1],P+[-1;1],P+[-1;0]];

```

```

96         P=Puntos(:,Elecc);
97     else
98         Puntos=[P+[-1;0],P+[-1;1],P+[-2;1],P+[-2;0]];
99         P=Puntos(:,Elecc);
100     end
101 elseif accion==4 %4 es r derecha
102     Prob=[0.2,0.1,0.4,0.3];
103     C=cumsum(Prob);
104     Opci=[1,2,3,4];
105     Elecc= Opci(5-sum((C>=rand)));
106     if P(1)<=199 && P(2)==200
107         Puntos=[P,P,P+[1;0],P+[1;0]];
108         P=Puntos(:,Elecc);
109     elseif P(1)==200 && P(2)<=199
110         Puntos=[P,P+[0;1],P+[0;1],P];
111         P=Puntos(:,Elecc);
112     elseif P(1)==200 && P(2)==200
113         Puntos=[P,P,P,P];
114         P=Puntos(:,Elecc);
115     else
116         Puntos=[P,P+[0;1],P+[1;1],P+[1;0]];
117         P=Puntos(:,Elecc);
118     end
119 end
120
121 Indic=ismember(P',CoordMeta','rows'); %Indica 1 si el punto esta en la
122     meta y 0 si no.
123 VecRecorrido=[VecRecorrido,P];
124 %Esta parte es OPCIONAL, es para ver el recorrido paso por paso
125 %{
126     figure(1)
127     pointsize = 20;
128     scatter(X, Y, pointsize,PosObst,'filled','s');
129     hold on
130     plot(VecRecorrido(1,:),VecRecorrido(2,:),'k*')
131     hold off
132     pause(0.00001)
133     %}
134 end

```

Algoritmo 3: Algoritmo que presenta mas de una simulación de recorridos del avión partiendo desde un punto inicial fijo y para una política dada. Este presenta una imagen de la retícula con los obstáculo y una silueta en escala de rojos en donde los puntos con mayor intensidad de rojo es en donde mas recorridos simulados pasaron.

```

1
2 function[SombraRecorrido]=SombrasSimulaciones(Pini,politica,Nsim)
3 %Pini=punto inicial y esta en forma horizontal

```



```

4 %Nsim= numero de simulaciones a realizar
5
6 %IMPORTANTE: Desactivar toda la parte grafica en 'simulacionAvion.m'.
7
8 tic
9 %Estados S en coordenadas X y Y
10 X= repmat(0:1:200,1,201);
11 Y= reshape(repmat(0:1:200,201,1),[1,40401]);
12 S=[X;Y];
13 %Coordenadas de los Obstaculos
14 YObsta=[repmat(110:1:135,1,5),repmat(25:1:29,1,26),repmat(90:1:110,1,5),
        repmat(75:1:79,1,21),repmat(125:1:128,1,51),repmat(50:1:100,1,4),repmat
        (150:1:175,1,5),repmat(75:1:79,1,26),repmat(135:1:160,1,5),repmat
        (100:1:104,1,26),repmat(35:1:78,1,4),repmat(75:1:78,1,40),repmat
        (125:1:129,1,26),repmat(130:1:150,1,5)];
15 XObsta=200-[reshape(repmat(171:1:175,26,1),[1,130]),reshape(repmat
        (65:1:90,5,1),[1,130]),reshape(repmat(121:1:125,21,1),[1,105]),reshape(
        repmat(90:1:110,5,1),[1,105]),reshape(repmat(100:1:150,4,1),[1,204]),
        reshape(repmat(72:1:75,51,1),[1,204]),reshape(repmat(121:1:125,26,1)
        ,[1,130]),reshape(repmat(25:1:50,5,1),[1,130]),reshape(repmat
        (96:1:100,26,1),[1,130]),reshape(repmat(40:1:65,5,1),[1,130]),reshape(
        repmat(122:1:125,44,1),[1,176]),reshape(repmat(126:1:165,4,1),[1,160]),
        reshape(repmat(50:1:75,5,1),[1,130]),reshape(repmat(71:1:75,21,1)
        ,[1,105])];
16 Obsta=[XObsta;YObsta];
17 PosObst=(ismember(S',Obsta','rows'))';
18 %Coordenadas de la Meta
19 CoordMeta=[repmat(155:1:157,1,3);reshape(repmat(155:1:157,3,1),[1,9])]; %
        Coordenadas de la meta
20
21 %El siguiente vector cuenta las veces que uno de los recorridos simulados
22 %paso por uno de los puntos del espacio de estados S.
23 SumaPosiciones=zeros(1,length(PosObst));
24 for i=1:Nsim
25     VecRecorrido=simulacionAvion(Pini,politica); %Puntos del recorridos
26     PosRecorrido=(ismember(S',VecRecorrido','rows'))'; %Posición de cada
        puntos del recorrido en S.
27     SumaPosiciones=SumaPosiciones+PosRecorrido;
28 end
29 SombraRecorrido=SumaPosiciones; %Esta genera la silueta roja que representa
        todos los recorridos.
30
31 %En esta parate se grafican las simulaciones.
32 pointsize = 20;
33 %el siguiente vector tiene el objetivo de generar una escala de rojos
34 colorEscalaRojo=((max(SumaPosiciones)-SumaPosiciones)')*(0.9/max(
        SumaPosiciones)).*((SumaPosiciones~=0)')+((SumaPosiciones==0)');
35 hold on
36 scatter(X, Y, pointsize, [ones(length(SumaPosiciones),1),colorEscalaRojo,
        colorEscalaRojo],'filled','s'); %Recorridos sumulados

```

```

37 scatter(XObsta, YObsta, pointsize, [zeros(length(XObsta),1),zeros(length(
    XObsta),1),zeros(length(XObsta),1)], 'filled', 's'); %Obstaculos
38 scatter(CoordMeta(1,:), CoordMeta(2,:), pointsize, [zeros(length(CoordMeta
    (1,:)),1),ones(length(CoordMeta(1,:)),1),zeros(length(CoordMeta(1,:)),1)
    ], 'filled', 's');
39 scatter(Pini(1), Pini(2), pointsize, [0,0,1], 'filled', 's'); %Punto inicial
40 title([ num2str( Nsim ) ' simulaciones empezando en el punto azul'])
41 hold off
42 toc

```

Algoritmo 4: Algoritmo que encuentra la política asociada a la solución del problema dual (3).

```

1
2 function[mu,A,PolitMod]=SolucionLinearProgAvion(lambda,c)
3 %c es un vector de numeros positivos que suman 1, debe estar en forma
    vertical;
4 %El numero de estados es 40401, es decir |S|=40401
5
6 tic
7 %Estados S en coordenadas X y Y
8 X= repmat(0:1:200,1,201);
9 Y= reshape(repmat(0:1:200,201,1),[1,40401]);
10 S=[X;Y];
11 L=length(S(1,:)); %En realidad este numero es 40401.
12
13 %Se procede a construir la matriz Q=[Qu;Qd;Ql;Qr], para tal fin se
14 %construye cada Qa una por una para cada acción 'a'.
15 %Todo esto lo hacemos de manera 'sparse' para parovechar la cantidad de
16 %ceros de estas matrices
17 %Para a=u
18 Qu=sparse(40401,40401);
19 %Tramos cada frontera como un caso particular
20 SU1=S(:,S(1,:) >=1 & S(2,:) ==199); %Puntos de la frontera descrita.
21 %Son los puntos a los que se puede llegar via la acción 'a' estando en '
    s'
22 PuntosSU11=SU1+[zeros(1,length(SU1(1,:)));ones(1,length(SU1(1,:)))];
23 PuntosSU12=SU1+[zeros(1,length(SU1(1,:)));ones(1,length(SU1(1,:)))];
24 PuntosSU13=SU1+[-ones(1,length(SU1(1,:)));ones(1,length(SU1(1,:)))];
25 PuntosSU14=SU1+[-ones(1,length(SU1(1,:)));ones(1,length(SU1(1,:)))];
26 %Posición en S de los puntos de la frontera descrita.
27 [indSU1,posSU1]=ismember(SU1',S','row');
28 %Posición en S del puntos PuntosSU11.
29 [indPunSU11,posPunSU11]=ismember(PuntosSU11',S','row');
30 %Probabilidad de padar al punto PuntosSU11 estando en s.
31 Qu1=sparse(posSU1',posPunSU11',0.3,40401,40401);
32 %Posición en S del puntos PuntosSU12.
33 [indPunSU12,posPunSU12]=ismember(PuntosSU12',S','row');
34 %Probabilidad de padar al punto PuntosSU12 estando en s.
35 Qu2=sparse(posSU1',posPunSU12',0.4,40401,40401);

```

```

36     %Posición en S del puntos PuntosSU13.
37     [indPunSU13,posPunSU13]=ismember(PuntosSU13',S','row');
38     %Probabilidad de padar al punto PuntosSU13 estando en s.
39     Qu3=sparse(posSU1',posPunSU13',0.2,40401,40401);
40     %Posición en S del puntos PuntosSU14.
41     [indPunSU14,posPunSU14]=ismember(PuntosSU14',S','row');
42     %Probabilidad de padar al punto PuntosSU14 estando en s.
43     Qu4=sparse(posSU1',posPunSU14',0.1,40401,40401);
44     Qu=Qu+Qu1+Qu2+Qu3+Qu4;
45     %NOTA: La descripción antterior hecha para SU1 es analoga para los casos
46     %siguientes.
47
48     SU2=S(:,S(1,:)>=1 & S(2,:)==200);
49     PuntosSU21=SU2+[zeros(1,length(SU2(1,:)));zeros(1,length(SU2(1,:)))];
50     PuntosSU22=SU2+[zeros(1,length(SU2(1,:)));zeros(1,length(SU2(1,:)))];
51     PuntosSU23=SU2+[-ones(1,length(SU2(1,:)));zeros(1,length(SU2(1,:)))];
52     PuntosSU24=SU2+[-ones(1,length(SU2(1,:)));zeros(1,length(SU2(1,:)))];
53     [indSU2,posSU2]=ismember(SU2',S','row');
54     [indPunSU21,posPunSU21]=ismember(PuntosSU21',S','row');
55     Qu1=sparse(posSU2,posPunSU21,0.3,40401,40401);
56     [indPunSU22,posPunSU22]=ismember(PuntosSU22',S','row');
57     Qu2=sparse(posSU2,posPunSU22,0.4,40401,40401);
58     [indPunSU23,posPunSU23]=ismember(PuntosSU23',S','row');
59     Qu3=sparse(posSU2,posPunSU23,0.2,40401,40401);
60     [indPunSU24,posPunSU24]=ismember(PuntosSU24',S','row');
61     Qu4=sparse(posSU2,posPunSU24,0.1,40401,40401);
62     Qu=Qu+Qu1+Qu2+Qu3+Qu4;
63
64     SU3=S(:,S(1,:)==0 & S(2,:)<=198);
65     PuntosSU31=SU3+[zeros(1,length(SU3(1,:)));ones(1,length(SU3(1,:)))];
66     PuntosSU32=SU3+[zeros(1,length(SU3(1,:)));2*ones(1,length(SU3(1,:)))];
67     PuntosSU33=SU3+[zeros(1,length(SU3(1,:)));2*ones(1,length(SU3(1,:)))];
68     PuntosSU34=SU3+[zeros(1,length(SU3(1,:)));ones(1,length(SU3(1,:)))];
69     [indSU3,posSU3]=ismember(SU3',S','row');
70     [indPunSU31,posPunSU31]=ismember(PuntosSU31',S','row');
71     Qu1=sparse(posSU3,posPunSU31,0.3,40401,40401);
72     [indPunSU32,posPunSU32]=ismember(PuntosSU32',S','row');
73     Qu2=sparse(posSU3,posPunSU32,0.4,40401,40401);
74     [indPunSU33,posPunSU33]=ismember(PuntosSU33',S','row');
75     Qu3=sparse(posSU3,posPunSU33,0.2,40401,40401);
76     [indPunSU34,posPunSU34]=ismember(PuntosSU34',S','row');
77     Qu4=sparse(posSU3,posPunSU34,0.1,40401,40401);
78     Qu=Qu+Qu1+Qu2+Qu3+Qu4;
79
80     SU4=S(:,S(1,:)==0 & S(2,:)==199);
81     PuntosSU41=SU4+[zeros(1,length(SU4(1,:)));ones(1,length(SU4(1,:)))];
82     PuntosSU42=SU4+[zeros(1,length(SU4(1,:)));ones(1,length(SU4(1,:)))];
83     PuntosSU43=SU4+[zeros(1,length(SU4(1,:)));ones(1,length(SU4(1,:)))];
84     PuntosSU44=SU4+[zeros(1,length(SU4(1,:)));ones(1,length(SU4(1,:)))];
85     [indSU4,posSU4]=ismember(SU4',S','row');

```

```

86 [indPunSU41, posPunSU41]=ismember(PuntosSU41',S','row');
87 Qu1=sparse(posSU4, posPunSU41, 0.3, 40401, 40401);
88 [indPunSU42, posPunSU42]=ismember(PuntosSU42',S','row');
89 Qu2=sparse(posSU4, posPunSU42, 0.4, 40401, 40401);
90 [indPunSU43, posPunSU43]=ismember(PuntosSU43',S','row');
91 Qu3=sparse(posSU4, posPunSU43, 0.2, 40401, 40401);
92 [indPunSU44, posPunSU44]=ismember(PuntosSU44',S','row');
93 Qu4=sparse(posSU4, posPunSU44, 0.1, 40401, 40401);
94 Qu=Qu+Qu1+Qu2+Qu3+Qu4;
95
96 SU5=S(:, S(1,:)==0 & S(2,:)==200);
97 PuntosSU51=SU5+[zeros(1, length(SU5(1,:))); zeros(1, length(SU5(1,:)))];
98 PuntosSU52=SU5+[zeros(1, length(SU5(1,:))); zeros(1, length(SU5(1,:)))];
99 PuntosSU53=SU5+[zeros(1, length(SU5(1,:))); zeros(1, length(SU5(1,:)))];
100 PuntosSU54=SU5+[zeros(1, length(SU5(1,:))); zeros(1, length(SU5(1,:)))];
101 [indSU5, posSU5]=ismember(SU5',S','row');
102 [indPunSU51, posPunSU51]=ismember(PuntosSU51',S','row');
103 Qu1=sparse(posSU5, posPunSU51, 0.3, 40401, 40401);
104 [indPunSU52, posPunSU52]=ismember(PuntosSU52',S','row');
105 Qu2=sparse(posSU5, posPunSU52, 0.4, 40401, 40401);
106 [indPunSU53, posPunSU53]=ismember(PuntosSU53',S','row');
107 Qu3=sparse(posSU5, posPunSU53, 0.2, 40401, 40401);
108 [indPunSU54, posPunSU54]=ismember(PuntosSU54',S','row');
109 Qu4=sparse(posSU5, posPunSU54, 0.1, 40401, 40401);
110 Qu=Qu+Qu1+Qu2+Qu3+Qu4;
111
112 SU6=S(:, not((S(1,:)>=1 & S(2,:)==199)|(S(1,:)>=1 & S(2,:)==200)|(S(1,:)
    ==0 & S(2,:)<=198)|(S(1,:)==0 & S(2,:)==199)|(S(1,:)==0 & S(2,:)==200)
    ));
113 PuntosSU61=SU6+[zeros(1, length(SU6(1,:))); ones(1, length(SU6(1,:)))];
114 PuntosSU62=SU6+[zeros(1, length(SU6(1,:))); 2*ones(1, length(SU6(1,:)))];
115 PuntosSU63=SU6+[-ones(1, length(SU6(1,:))); 2*ones(1, length(SU6(1,:)))];
116 PuntosSU64=SU6+[-ones(1, length(SU6(1,:))); ones(1, length(SU6(1,:)))];
117 [indSU6, posSU6]=ismember(SU6',S','row');
118 [indPunSU61, posPunSU61]=ismember(PuntosSU61',S','row');
119 Qu1=sparse(posSU6, posPunSU61, 0.3, 40401, 40401);
120 [indPunSU62, posPunSU62]=ismember(PuntosSU62',S','row');
121 Qu2=sparse(posSU6, posPunSU62, 0.4, 40401, 40401);
122 [indPunSU63, posPunSU63]=ismember(PuntosSU63',S','row');
123 Qu3=sparse(posSU6, posPunSU63, 0.2, 40401, 40401);
124 [indPunSU64, posPunSU64]=ismember(PuntosSU64',S','row');
125 Qu4=sparse(posSU6, posPunSU64, 0.1, 40401, 40401);
126 Qu=Qu+Qu1+Qu2+Qu3+Qu4;
127
128 %Para a=d
129 Qd=sparse(40401, 40401);
130 SD=S(:, S(1,:)>=1 & S(2,:)==0);
131 PuntosSD1=SD+[zeros(1, length(SD(1,:))); zeros(1, length(SD(1,:)))];
132 PuntosSD2=SD+[zeros(1, length(SD(1,:))); zeros(1, length(SD(1,:)))];
133 PuntosSD3=SD+[-ones(1, length(SD(1,:))); zeros(1, length(SD(1,:)))];

```

```

134 PuntosSD4=SD+[-ones(1,length(SD(1,:)));zeros(1,length(SD(1,:)))];
135 [indSD,posSD]=ismember(SD','S','row');
136 [indPunSD1,posPunSD1]=ismember(PuntosSD1','S','row');
137 Qd1=sparse(posSD',posPunSD1',0.3,40401,40401);
138 [indPunSD2,posPunSD2]=ismember(PuntosSD2','S','row');
139 Qd2=sparse(posSD',posPunSD2',0.3,40401,40401);
140 [indPunSD3,posPunSD3]=ismember(PuntosSD3','S','row');
141 Qd3=sparse(posSD',posPunSD3',0.2,40401,40401);
142 [indPunSD4,posPunSD4]=ismember(PuntosSD4','S','row');
143 Qd4=sparse(posSD',posPunSD4',0.2,40401,40401);
144 Qd=Qd+Qd1+Qd2+Qd3+Qd4;
145
146 SD=S(:,S(1,:)==0 & S(2,:)>=1);
147 PuntosSD1=SD+[zeros(1,length(SD(1,:)));zeros(1,length(SD(1,:)))];
148 PuntosSD2=SD+[zeros(1,length(SD(1,:)));-ones(1,length(SD(1,:)))];
149 PuntosSD3=SD+[zeros(1,length(SD(1,:)));-ones(1,length(SD(1,:)))];
150 PuntosSD4=SD+[zeros(1,length(SD(1,:)));zeros(1,length(SD(1,:)))];
151 [indSD,posSD]=ismember(SD','S','row');
152 [indPunSD1,posPunSD1]=ismember(PuntosSD1','S','row');
153 Qd1=sparse(posSD',posPunSD1',0.3,40401,40401);
154 [indPunSD2,posPunSD2]=ismember(PuntosSD2','S','row');
155 Qd2=sparse(posSD',posPunSD2',0.3,40401,40401);
156 [indPunSD3,posPunSD3]=ismember(PuntosSD3','S','row');
157 Qd3=sparse(posSD',posPunSD3',0.2,40401,40401);
158 [indPunSD4,posPunSD4]=ismember(PuntosSD4','S','row');
159 Qd4=sparse(posSD',posPunSD4',0.2,40401,40401);
160 Qd=Qd+Qd1+Qd2+Qd3+Qd4;
161
162 SD=S(:,S(1,:)==0 & S(2,:)==0);
163 PuntosSD1=SD+[zeros(1,length(SD(1,:)));zeros(1,length(SD(1,:)))];
164 PuntosSD2=SD+[zeros(1,length(SD(1,:)));zeros(1,length(SD(1,:)))];
165 PuntosSD3=SD+[zeros(1,length(SD(1,:)));zeros(1,length(SD(1,:)))];
166 PuntosSD4=SD+[zeros(1,length(SD(1,:)));zeros(1,length(SD(1,:)))];
167 [indSD,posSD]=ismember(SD','S','row');
168 [indPunSD1,posPunSD1]=ismember(PuntosSD1','S','row');
169 Qd1=sparse(posSD',posPunSD1',0.3,40401,40401);
170 [indPunSD2,posPunSD2]=ismember(PuntosSD2','S','row');
171 Qd2=sparse(posSD',posPunSD2',0.3,40401,40401);
172 [indPunSD3,posPunSD3]=ismember(PuntosSD3','S','row');
173 Qd3=sparse(posSD',posPunSD3',0.2,40401,40401);
174 [indPunSD4,posPunSD4]=ismember(PuntosSD4','S','row');
175 Qd4=sparse(posSD',posPunSD4',0.2,40401,40401);
176 Qd=Qd+Qd1+Qd2+Qd3+Qd4;
177
178 SD=S(:,not((S(1,:)>=1 & S(2,:)==0)|(S(1,:)==0 & S(2,:)>=1)|(S(1,:)==0 & S
    (2,:)==0)));
179 PuntosSD1=SD+[zeros(1,length(SD(1,:)));zeros(1,length(SD(1,:)))];
180 PuntosSD2=SD+[zeros(1,length(SD(1,:)));-ones(1,length(SD(1,:)))];
181 PuntosSD3=SD+[-ones(1,length(SD(1,:)));-ones(1,length(SD(1,:)))];
182 PuntosSD4=SD+[-ones(1,length(SD(1,:)));zeros(1,length(SD(1,:)))];

```

```

183 [indSD, posSD]=ismember(SD',S','row');
184 [indPunSD1, posPunSD1]=ismember(PuntosSD1',S','row');
185 Qd1=sparse(posSD',posPunSD1',0.3,40401,40401);
186 [indPunSD2, posPunSD2]=ismember(PuntosSD2',S','row');
187 Qd2=sparse(posSD',posPunSD2',0.3,40401,40401);
188 [indPunSD3, posPunSD3]=ismember(PuntosSD3',S','row');
189 Qd3=sparse(posSD',posPunSD3',0.2,40401,40401);
190 [indPunSD4, posPunSD4]=ismember(PuntosSD4',S','row');
191 Qd4=sparse(posSD',posPunSD4',0.2,40401,40401);
192 Qd=Qd+Qd1+Qd2+Qd3+Qd4;
193
194 %Para a=1
195 Q1=sparse(40401,40401);
196 SL=S(:,S(1,:) >=2 & S(2,:) ==200);
197 PuntosSL1=SL+[-ones(1,length(SL(1,:)));zeros(1,length(SL(1,:)))];
198 PuntosSL2=SL+[-ones(1,length(SL(1,:)));zeros(1,length(SL(1,:)))];
199 PuntosSL3=SL+[-2*ones(1,length(SL(1,:)));zeros(1,length(SL(1,:)))];
200 PuntosSL4=SL+[-2*ones(1,length(SL(1,:)));zeros(1,length(SL(1,:)))];
201 [indSL, posSL]=ismember(SL',S','row');
202 [indPunSL1, posPunSL1]=ismember(PuntosSL1',S','row');
203 Q11=sparse(posSL',posPunSL1',0.2,40401,40401);
204 [indPunSL2, posPunSL2]=ismember(PuntosSL2',S','row');
205 Q12=sparse(posSL',posPunSL2',0.3,40401,40401);
206 [indPunSL3, posPunSL3]=ismember(PuntosSL3',S','row');
207 Q13=sparse(posSL',posPunSL3',0.2,40401,40401);
208 [indPunSL4, posPunSL4]=ismember(PuntosSL4',S','row');
209 Q14=sparse(posSL',posPunSL4',0.3,40401,40401);
210 Q1=Q1+Q11+Q12+Q13+Q14;
211
212 SL=S(:,S(1,:)==0 & S(2,:) ==200);
213 PuntosSL1=SL+[zeros(1,length(SL(1,:)));zeros(1,length(SL(1,:)))];
214 PuntosSL2=SL+[zeros(1,length(SL(1,:)));zeros(1,length(SL(1,:)))];
215 PuntosSL3=SL+[zeros(1,length(SL(1,:)));zeros(1,length(SL(1,:)))];
216 PuntosSL4=SL+[zeros(1,length(SL(1,:)));zeros(1,length(SL(1,:)))];
217 [indSL, posSL]=ismember(SL',S','row');
218 [indPunSL1, posPunSL1]=ismember(PuntosSL1',S','row');
219 Q11=sparse(posSL',posPunSL1',0.2,40401,40401);
220 [indPunSL2, posPunSL2]=ismember(PuntosSL2',S','row');
221 Q12=sparse(posSL',posPunSL2',0.3,40401,40401);
222 [indPunSL3, posPunSL3]=ismember(PuntosSL3',S','row');
223 Q13=sparse(posSL',posPunSL3',0.2,40401,40401);
224 [indPunSL4, posPunSL4]=ismember(PuntosSL4',S','row');
225 Q14=sparse(posSL',posPunSL4',0.3,40401,40401);
226 Q1=Q1+Q11+Q12+Q13+Q14;
227
228 SL=S(:,S(1,:) ==1 & S(2,:) ==200);
229 PuntosSL1=SL+[-ones(1,length(SL(1,:)));zeros(1,length(SL(1,:)))];
230 PuntosSL2=SL+[-ones(1,length(SL(1,:)));zeros(1,length(SL(1,:)))];
231 PuntosSL3=SL+[-ones(1,length(SL(1,:)));zeros(1,length(SL(1,:)))];
232 PuntosSL4=SL+[-ones(1,length(SL(1,:)));zeros(1,length(SL(1,:)))];

```

```

233 [indSL, posSL]=ismember(SL',S','row');
234 [indPunSL1, posPunSL1]=ismember(PuntosSL1',S','row');
235 Q11=sparse(posSL',posPunSL1',0.2,40401,40401);
236 [indPunSL2, posPunSL2]=ismember(PuntosSL2',S','row');
237 Q12=sparse(posSL',posPunSL2',0.3,40401,40401);
238 [indPunSL3, posPunSL3]=ismember(PuntosSL3',S','row');
239 Q13=sparse(posSL',posPunSL3',0.2,40401,40401);
240 [indPunSL4, posPunSL4]=ismember(PuntosSL4',S','row');
241 Q14=sparse(posSL',posPunSL4',0.3,40401,40401);
242 Q1=Q1+Q11+Q12+Q13+Q14;
243
244 SL=S(:,S(1,:)==0 & S(2,:) <=199);
245 PuntosSL1=SL+[zeros(1,length(SL(1,:)));zeros(1,length(SL(1,:)))];
246 PuntosSL2=SL+[zeros(1,length(SL(1,:)));ones(1,length(SL(1,:)))];
247 PuntosSL3=SL+[zeros(1,length(SL(1,:)));ones(1,length(SL(1,:)))];
248 PuntosSL4=SL+[zeros(1,length(SL(1,:)));zeros(1,length(SL(1,:)))];
249 [indSL, posSL]=ismember(SL',S','row');
250 [indPunSL1, posPunSL1]=ismember(PuntosSL1',S','row');
251 Q11=sparse(posSL',posPunSL1',0.2,40401,40401);
252 [indPunSL2, posPunSL2]=ismember(PuntosSL2',S','row');
253 Q12=sparse(posSL',posPunSL2',0.3,40401,40401);
254 [indPunSL3, posPunSL3]=ismember(PuntosSL3',S','row');
255 Q13=sparse(posSL',posPunSL3',0.2,40401,40401);
256 [indPunSL4, posPunSL4]=ismember(PuntosSL4',S','row');
257 Q14=sparse(posSL',posPunSL4',0.3,40401,40401);
258 Q1=Q1+Q11+Q12+Q13+Q14;
259
260 SL=S(:,S(1,:)==1 & S(2,:) <=199);
261 PuntosSL1=SL+[-ones(1,length(SL(1,:)));zeros(1,length(SL(1,:)))];
262 PuntosSL2=SL+[-ones(1,length(SL(1,:)));ones(1,length(SL(1,:)))];
263 PuntosSL3=SL+[-ones(1,length(SL(1,:)));ones(1,length(SL(1,:)))];
264 PuntosSL4=SL+[-ones(1,length(SL(1,:)));zeros(1,length(SL(1,:)))];
265 [indSL, posSL]=ismember(SL',S','row');
266 [indPunSL1, posPunSL1]=ismember(PuntosSL1',S','row');
267 Q11=sparse(posSL',posPunSL1',0.2,40401,40401);
268 [indPunSL2, posPunSL2]=ismember(PuntosSL2',S','row');
269 Q12=sparse(posSL',posPunSL2',0.3,40401,40401);
270 [indPunSL3, posPunSL3]=ismember(PuntosSL3',S','row');
271 Q13=sparse(posSL',posPunSL3',0.2,40401,40401);
272 [indPunSL4, posPunSL4]=ismember(PuntosSL4',S','row');
273 Q14=sparse(posSL',posPunSL4',0.3,40401,40401);
274 Q1=Q1+Q11+Q12+Q13+Q14;
275
276 SL=S(:,not((S(1,:) >=2 & S(2,:) ==200)|(S(1,:) ==0 & S(2,:) ==200)|(S(1,:) ==1
    & S(2,:) ==200)|(S(1,:) ==0 & S(2,:) <=199)|(S(1,:) ==1 & S(2,:) <=199)));
277 PuntosSL1=SL+[-ones(1,length(SL(1,:)));zeros(1,length(SL(1,:)))];
278 PuntosSL2=SL+[-ones(1,length(SL(1,:)));ones(1,length(SL(1,:)))];
279 PuntosSL3=SL+[-2*ones(1,length(SL(1,:)));ones(1,length(SL(1,:)))];
280 PuntosSL4=SL+[-2*ones(1,length(SL(1,:)));zeros(1,length(SL(1,:)))];
281 [indSL, posSL]=ismember(SL',S','row');

```



```

282 [indPunSL1, posPunSL1]=ismember(PuntosSL1',S','row');
283 Ql1=sparse(posSL',posPunSL1',0.2,40401,40401);
284 [indPunSL2, posPunSL2]=ismember(PuntosSL2',S','row');
285 Ql2=sparse(posSL',posPunSL2',0.3,40401,40401);
286 [indPunSL3, posPunSL3]=ismember(PuntosSL3',S','row');
287 Ql3=sparse(posSL',posPunSL3',0.2,40401,40401);
288 [indPunSL4, posPunSL4]=ismember(PuntosSL4',S','row');
289 Ql4=sparse(posSL',posPunSL4',0.3,40401,40401);
290 Ql=Ql+Ql1+Ql2+Ql3+Ql4;
291
292 %Para a=r
293 Qr=sparse(40401,40401);
294 SR=S(:,S(1,:) <=199 & S(2,:) ==200);
295 PuntosSR1=SR+[zeros(1,length(SR(1,:)));zeros(1,length(SR(1,:)))];
296 PuntosSR2=SR+[zeros(1,length(SR(1,:)));zeros(1,length(SR(1,:)))];
297 PuntosSR3=SR+[ones(1,length(SR(1,:)));zeros(1,length(SR(1,:)))];
298 PuntosSR4=SR+[ones(1,length(SR(1,:)));zeros(1,length(SR(1,:)))];
299 [indSR, posSR]=ismember(SR',S','row');
300 [indPunSR1, posPunSR1]=ismember(PuntosSR1',S','row');
301 Qr1=sparse(posSR',posPunSR1',0.2,40401,40401);
302 [indPunSR2, posPunSR2]=ismember(PuntosSR2',S','row');
303 Qr2=sparse(posSR',posPunSR2',0.1,40401,40401);
304 [indPunSR3, posPunSR3]=ismember(PuntosSR3',S','row');
305 Qr3=sparse(posSR',posPunSR3',0.4,40401,40401);
306 [indPunSR4, posPunSR4]=ismember(PuntosSR4',S','row');
307 Qr4=sparse(posSR',posPunSR4',0.3,40401,40401);
308 Qr=Qr+Qr1+Qr2+Qr3+Qr4;
309
310 SR=S(:,S(1,:) ==200 & S(2,:) <=199);
311 PuntosSR1=SR+[zeros(1,length(SR(1,:)));zeros(1,length(SR(1,:)))];
312 PuntosSR2=SR+[zeros(1,length(SR(1,:)));ones(1,length(SR(1,:)))];
313 PuntosSR3=SR+[zeros(1,length(SR(1,:)));ones(1,length(SR(1,:)))];
314 PuntosSR4=SR+[zeros(1,length(SR(1,:)));zeros(1,length(SR(1,:)))];
315 [indSR, posSR]=ismember(SR',S','row');
316 [indPunSR1, posPunSR1]=ismember(PuntosSR1',S','row');
317 Qr1=sparse(posSR',posPunSR1',0.2,40401,40401);
318 [indPunSR2, posPunSR2]=ismember(PuntosSR2',S','row');
319 Qr2=sparse(posSR',posPunSR2',0.1,40401,40401);
320 [indPunSR3, posPunSR3]=ismember(PuntosSR3',S','row');
321 Qr3=sparse(posSR',posPunSR3',0.4,40401,40401);
322 [indPunSR4, posPunSR4]=ismember(PuntosSR4',S','row');
323 Qr4=sparse(posSR',posPunSR4',0.3,40401,40401);
324 Qr=Qr+Qr1+Qr2+Qr3+Qr4;
325
326 SR=S(:,S(1,:) ==200 & S(2,:) ==200);
327 PuntosSR1=SR+[zeros(1,length(SR(1,:)));zeros(1,length(SR(1,:)))];
328 PuntosSR2=SR+[zeros(1,length(SR(1,:)));zeros(1,length(SR(1,:)))];
329 PuntosSR3=SR+[zeros(1,length(SR(1,:)));zeros(1,length(SR(1,:)))];
330 PuntosSR4=SR+[zeros(1,length(SR(1,:)));zeros(1,length(SR(1,:)))];
331 [indSR, posSR]=ismember(SR',S','row');

```



```

332 [indPunSR1,posPunSR1]=ismember(PuntosSR1','S','row');
333 Qr1=sparse(posSR',posPunSR1',0.2,40401,40401);
334 [indPunSR2,posPunSR2]=ismember(PuntosSR2','S','row');
335 Qr2=sparse(posSR',posPunSR2',0.1,40401,40401);
336 [indPunSR3,posPunSR3]=ismember(PuntosSR3','S','row');
337 Qr3=sparse(posSR',posPunSR3',0.4,40401,40401);
338 [indPunSR4,posPunSR4]=ismember(PuntosSR4','S','row');
339 Qr4=sparse(posSR',posPunSR4',0.3,40401,40401);
340 Qr=Qr+Qr1+Qr2+Qr3+Qr4;
341
342 SR=S(:,not((S(1,:)<=199 & S(2,:)==200)|(S(1,:)==200 & S(2,:)<=199)|(S
    (1,:)==200 & S(2,:)==200)));
343 PuntosSR1=SR+[zeros(1,length(SR(1,:)));zeros(1,length(SR(1,:)))];
344 PuntosSR2=SR+[zeros(1,length(SR(1,:)));ones(1,length(SR(1,:)))];
345 PuntosSR3=SR+[ones(1,length(SR(1,:)));ones(1,length(SR(1,:)))];
346 PuntosSR4=SR+[ones(1,length(SR(1,:)));zeros(1,length(SR(1,:)))];
347 [indSR,posSR]=ismember(SR','S','row');
348 [indPunSR1,posPunSR1]=ismember(PuntosSR1','S','row');
349 Qr1=sparse(posSR',posPunSR1',0.2,40401,40401);
350 [indPunSR2,posPunSR2]=ismember(PuntosSR2','S','row');
351 Qr2=sparse(posSR',posPunSR2',0.1,40401,40401);
352 [indPunSR3,posPunSR3]=ismember(PuntosSR3','S','row');
353 Qr3=sparse(posSR',posPunSR3',0.4,40401,40401);
354 [indPunSR4,posPunSR4]=ismember(PuntosSR4','S','row');
355 Qr4=sparse(posSR',posPunSR4',0.3,40401,40401);
356 Qr=Qr+Qr1+Qr2+Qr3+Qr4;
357
358 Q=[Qu;Qd;Ql;Qr];
359 Identidades=[speye(40401);speye(40401);speye(40401);speye(40401)];
360
361 A=(Identidades-lambda*Q)';
362
363 %Ahora creamos el vector Recompensas, priemro pasamos por los obstaculos
364 %Coordenadas de los Obstaculos
365 YObsta=[repmat(110:1:135,1,5),repmat(25:1:29,1,26),repmat(90:1:110,1,5),
    repmat(75:1:79,1,21),repmat(125:1:128,1,51),repmat(50:1:100,1,4),repmat
    (150:1:175,1,5),repmat(75:1:79,1,26),repmat(135:1:160,1,5),repmat
    (100:1:104,1,26),repmat(35:1:78,1,4),repmat(75:1:78,1,40),repmat
    (125:1:129,1,26),repmat(130:1:150,1,5)];
366 XObsta=200-[reshape(repmat(171:1:175,26,1),[1,130]),reshape(repmat
    (65:1:90,5,1),[1,130]),reshape(repmat(121:1:125,21,1),[1,105]),reshape(
    repmat(90:1:110,5,1),[1,105]),reshape(repmat(100:1:150,4,1),[1,204]),
    reshape(repmat(72:1:75,51,1),[1,204]),reshape(repmat(121:1:125,26,1)
    ,[1,130]),reshape(repmat(25:1:50,5,1),[1,130]),reshape(repmat
    (96:1:100,26,1),[1,130]),reshape(repmat(40:1:65,5,1),[1,130]),reshape(
    repmat(122:1:125,44,1),[1,176]),reshape(repmat(126:1:165,4,1),[1,160]),
    reshape(repmat(50:1:75,5,1),[1,130]),reshape(repmat(71:1:75,21,1)
    ,[1,105])];
367 Obsta=[XObsta;YObsta];
368 PosObst=(ismember(S',Obsta','rows'))'; %Posición de los obstaculos

```

```

369 %Coordenadas de la meta
370 CoordMeta=[ repmat(155:1:157,1,3); reshape(repmat(155:1:157,3,1),[1,9])];
371 PosMeta=(ismember(S',CoordMeta','rows'))';
372
373 %Definimos la recompensa
374 RenObs=PosObst*(-100000000);
375 RenEstad=ones(1,L);
376 RenMeta=PosMeta*(10000);
377 RencomAux=RenObs+RenEstad+RenMeta;
378 RencomAuxMat=reshape(RencomAux,[201,201])';
379 %Para u
380 RencomU1Mat=[[2*RencomAuxMat(2:1:200,1);4;4],[RencomAuxMat
(2:1:200,2:1:201);2*RencomAuxMat(201,2:1:201);2*RencomAuxMat
(201,2:1:201)]];
381 RencomU2Mat=[[2*RencomAuxMat(3:1:201,1),RencomAuxMat(3:1:201,2:1:201)
];zeros(1,201);zeros(1,201)];
382 RencomU3Mat=[zeros(201,1),[RencomAuxMat(3:1:201,1:1:200);zeros(1,200);
zeros(1,200)]];
383 RencomU4Mat=[zeros(201,1),[RencomAuxMat(2:1:200,1:1:200);2*
RencomAuxMat(201,1:1:200);2*RencomAuxMat(201,1:1:200)]];
384 RencomUMat=RencomU1Mat+RencomU2Mat+RencomU3Mat+RencomU4Mat;
385 ReomU=reshape(RencomUMat',[1,201*201]);
386 %Para d
387 RencomD1Mat=[[4;2*RencomAuxMat(2:1:201,1)],[2*RencomAuxMat(1,2:1:201);
RencomAuxMat(2:1:201,2:1:201)]];
388 RencomD2Mat=[zeros(1,201);[2*RencomAuxMat(1:1:200,1),RencomAuxMat
(1:1:200,2:1:201)]];
389 RencomD3Mat=[zeros(201,1),[zeros(1,200);RencomAuxMat(1:1:200,1:1:200)
]];
390 RencomD4Mat=[zeros(201,1),[2*RencomAuxMat(1,1:1:200);RencomAuxMat
(2:1:201,1:1:200)]];
391 RencomDMat=RencomD1Mat+RencomD2Mat+RencomD3Mat+RencomD4Mat;
392 ReomD=reshape(RencomDMat',[1,201*201]);
393 %Para L
394 RencomL1Mat=[[2*RencomAuxMat(1:1:200,1);4],[2*RencomAuxMat(1:1:200,1)
;4],[RencomAuxMat(1:1:200,2:1:200);2*RencomAuxMat(201,2:1:200)]];
395 RencomL2Mat=[[2*RencomAuxMat(2:1:201,1),RencomAuxMat(2:1:201,2:1:201)
];zeros(1,201)];
396 RencomL3Mat=[zeros(201,1),zeros(201,1),[RencomAuxMat(2:1:201,1:1:199);
zeros(1,199)];
397 RencomL4Mat=[zeros(201,1),zeros(201,1),[RencomAuxMat(1:1:200,1:1:199)
;2*RencomAuxMat(201,1:1:199)]];
398 RencomLMat=RencomL1Mat+RencomL2Mat+RencomL3Mat+RencomL4Mat;
399 ReomL=reshape(RencomLMat',[1,201*201]);
400 %Para R
401 RencomR1Mat=[[RencomAuxMat(1:1:200,1:1:200);2*RencomAuxMat
(201,1:1:200)],[2*RencomAuxMat(1:1:200,201);4]];
402 RencomR2Mat=[[RencomAuxMat(2:1:201,1:1:200),2*RencomAuxMat
(2:1:201,201)];zeros(1,201)];
403 RencomR3Mat=[[RencomAuxMat(2:1:201,2:1:201);zeros(1,200)],zeros(201,1)

```

```

];
404 RencomR4Mat=[[RencomAuxMat(1:1:200,2:1:201);2*RencomAuxMat
      (201,2:1:201)],zeros(201,1)];
405 RencomRMat=RencomR1Mat+RencomR2Mat+RencomR3Mat+RencomR4Mat;
406 RecomR=reshape(RencomRMat',[1,201*201]);
407 %Vector de recompensas
408 Rencom=[RecomU,RecomD,RecomL,RecomR]'; %Este es el vector r del
      documento.
409
410 %Aquí podemos cambiar el algoritmo y usar: 'dual-simplex' o 'interior-point
      ', pero 'dual-simplex' no funciona en este caso.
411 options = optimoptions('linprog','Algorithm','interior-point');
412 mu = linprog(-Rencom,[],[],A,c,zeros(4*L,1),[],options);
413
414
415 %La política es:
416 Polit=mu./repmat(mu(1:1:40401)+mu(40402:1:2*40401)+mu(2*40401+1:1:3*40401)+
      mu(3*40401+1:1:4*40401),4,1);
417 %Escribimos la política como una matriz de tamaño |S|×4, donde cada columna
418 %es una acción. columna 1 es subir, 2 es baja, 3 es izquierda y 4 es
419 %Derecha
420 PolitMod=reshape(Polit',[40401,4]);
421
422 %Graficamos esa política, creamos tres figuras, una por cada acción
423 pointsize=20;
424 figure(1) %Subir
425 scatter(X, Y, pointsize, PolitMod(:,1),'filled','s');
426 colbar1=colorbar;
427 title(colbar1,'Probabilidad')
428 title('Subir (u)')
429 figure(2) %Bajar
430 scatter(X, Y, pointsize, PolitMod(:,2),'filled','s');
431 colbar2=colorbar;
432 title(colbar2,'Probabilidad')
433 title('Bajar (d)')
434 figure(3) %Izquierda
435 scatter(X, Y, pointsize, PolitMod(:,3),'filled','s');
436 colbar3=colorbar;
437 title(colbar3,'Probabilidad')
438 title('Izquierda (l)')
439 figure(4) %Derecha
440 scatter(X, Y, pointsize, PolitMod(:,4),'filled','s');
441 colbar4=colorbar;
442 title(colbar4,'Probabilidad')
443 title('Derecha (r)')
444 toc

```

Algoritmo 5: Algoritmo que crea la simulación de un recorrido del avión partiendo desde un punto inicial y para una política no determinista dada por el

método de Linear Programming.

```

1
2 function [VecRecorrido]=simulacionAvionPolitLinearProg(Pini,politica)
3
4 %Politica debe ser una matriz de tamaño |S|x4.
5 %Pini es el punto inicial del recorrido, este esta en forma horizontal
6
7 %Estados S en coordenadas X y Y
8 X= repmat(0:1:200,1,201);
9 Y= reshape(repmat(0:1:200,201,1),[1,40401]);
10 S=[X;Y];
11 %Coordenadas de los obstaculos
12 YObsta=[repmat(110:1:135,1,5),repmat(25:1:29,1,26),repmat(90:1:110,1,5),
    repmat(75:1:79,1,21),repmat(125:1:128,1,51),repmat(50:1:100,1,4),repmat
    (150:1:175,1,5),repmat(75:1:79,1,26),repmat(135:1:160,1,5),repmat
    (100:1:104,1,26),repmat(35:1:78,1,4),repmat(75:1:78,1,40),repmat
    (125:1:129,1,26),repmat(130:1:150,1,5)];
13 XObsta=200-[reshape(repmat(171:1:175,26,1),[1,130]),reshape(repmat
    (65:1:90,5,1),[1,130]),reshape(repmat(121:1:125,21,1),[1,105]),reshape(
    repmat(90:1:110,5,1),[1,105]),reshape(repmat(100:1:150,4,1),[1,204]),
    reshape(repmat(72:1:75,51,1),[1,204]),reshape(repmat(121:1:125,26,1)
    ,[1,130]),reshape(repmat(25:1:50,5,1),[1,130]),reshape(repmat
    (96:1:100,26,1),[1,130]),reshape(repmat(40:1:65,5,1),[1,130]),reshape(
    repmat(122:1:125,44,1),[1,176]),reshape(repmat(126:1:165,4,1),[1,160]),
    reshape(repmat(50:1:75,5,1),[1,130]),reshape(repmat(71:1:75,21,1)
    ,[1,105])];
14 Obsta=[XObsta;YObsta];
15 PosObst=(ismember(S',Obsta','rows'))'; %Posición de los obstaculos
16 %Coordenadas de la meta
17 CoordMeta=[repmat(155:1:157,1,3);reshape(repmat(155:1:157,3,1),[1,9])]; %
    Coordenadas de la meta
18 PosMeta=(ismember(S',CoordMeta','rows'))';
19
20 %Pini es un punto de la forma [a,b].
21 P=Pini'; %Queremos el punto en la forma vertical
22
23 Indic=ismember(P',CoordMeta','rows'); %Indica 1 si el punto esta en la
    meta y 0 si no
24 VecRecorrido=P; %Aqui cuardamos los puntos por donde pasa el recorrido.
25 iter=0; %Para controlar.
26 while Indic~=1
27     iter=iter+1;
28     [Paux,Pos]=ismember(P',S','rows'); %Posición de P respecto a S
29     %Sigüientes 4 líneas son para elegir aleatoriamente una de las
30     %acciones, la probabilidad es dada por la politica no determinista
31     ProbAccion=politica(Pos,:);
32     CAccion=cumsum(ProbAccion);
33     OpciAccion=[1,2,3,4];
34     accion= OpciAccion(5-sum((CAccion>=rand)));
35     if accion==1 %u Subir

```

```

36 %Siguietes 4 lineas son para elegir uno de los puntos a dosnde se
37 %pude subir
38 Prob=[0.3,0.4,0.2,0.1];
39 C=cumsum(Prob);
40 Opci=[1,2,3,4];
41 Elecc= Opci(5-sum((C>=rand)));
42 %Los condicionales discriman el interior y las fronteras
43 if P(1)>=1 && P(2)==199
44     Puntos=[P+[0;1],P+[0;1],P+[-1;1],P+[-1;1]];
45     P=Puntos(:,Elecc);
46 elseif P(1)>=1 && P(2)==200
47     Puntos=[P,P,P+[-1;0],P+[-1;0]];
48     P=Puntos(:,Elecc);
49 elseif P(1)==0 && P(2)<=198
50     Puntos=[P+[0;1],P+[0;2],P+[0;2],P+[0;1]];
51     P=Puntos(:,Elecc);
52 elseif P(1)==0 && P(2)==199
53     Puntos=[P+[0;1],P+[0;1],P+[0;1],P+[0;1]];
54     P=Puntos(:,Elecc);
55 elseif P(1)==0 && P(2)==200
56     Puntos=[P,P,P,P];
57     P=Puntos(:,Elecc);
58 else
59     Puntos=[P+[0;1],P+[0;2],P+[-1;2],P+[-1;1]];
60     P=Puntos(:,Elecc);
61 end
62 %Los siguientes 'elseif' se describen analogo a como se describio
63 el
64 %caso 'u'
65 elseif accion==2 %d bajar
66     Prob=[0.3,0.3,0.2,0.2];
67     C=cumsum(Prob);
68     Opci=[1,2,3,4];
69     Elecc= Opci(5-sum((C>=rand)));
70 if P(1)>=1 && P(2)==0
71     Puntos=[P,P,P+[-1;0],P+[-1;0]];
72     P=Puntos(:,Elecc);
73 elseif P(1)==0 && P(2)>=1
74     Puntos=[P,P+[0;-1],P+[0;-1],P];
75     P=Puntos(:,Elecc);
76 elseif P(1)==0 && P(2)==0
77     Puntos=[P,P,P,P];
78     P=Puntos(:,Elecc);
79 else
80     Puntos=[P,P+[0;-1],P+[-1;-1],P+[-1;0]];
81     P=Puntos(:,Elecc);
82 end
83 elseif accion==3 %l izquierda
84     Prob=[0.2,0.3,0.2,0.3];
85     C=cumsum(Prob);

```

```

85     Opci=[1,2,3,4];
86     Elecc= Opci(5-sum((C>=rand)));
87     if P(1)>=2 && P(2)==200
88         Puntos=[P+[-1;0],P+[-1;0],P+[-2;0],P+[-2;0]];
89         P=Puntos(:,Elecc);
90     elseif P(1)==0 && P(2)==200
91         Puntos=[P,P,P,P];
92         P=Puntos(:,Elecc);
93     elseif P(1)==1 && P(2)==200
94         Puntos=[P+[-1;0],P+[-1;0],P+[-1;0],P+[-1;0]];
95         P=Puntos(:,Elecc);
96     elseif P(1)==0 && P(2)<=199
97         Puntos=[P,P+[0;1],P+[0;1],P];
98         P=Puntos(:,Elecc);
99     elseif P(1)==1 && P(2)<=199
100        Puntos=[P+[-1;0],P+[-1;1],P+[-1;1],P+[-1;0]];
101        P=Puntos(:,Elecc);
102    else
103        Puntos=[P+[-1;0],P+[-1;1],P+[-2;1],P+[-2;0]];
104        P=Puntos(:,Elecc);
105    end
106 elseif accion==4 %r derecha
107     Prob=[0.2,0.1,0.4,0.3];
108     C=cumsum(Prob);
109     Opci=[1,2,3,4];
110     Elecc= Opci(5-sum((C>=rand)));
111     if P(1)<=199 && P(2)==200
112         Puntos=[P,P,P+[1;0],P+[1;0]];
113         P=Puntos(:,Elecc);
114     elseif P(1)==200 && P(2)<=199
115         Puntos=[P,P+[0;1],P+[0;1],P];
116         P=Puntos(:,Elecc);
117     elseif P(1)==200 && P(2)==200
118         Puntos=[P,P,P,P];
119         P=Puntos(:,Elecc);
120     else
121         Puntos=[P,P+[0;1],P+[1;1],P+[1;0]];
122         P=Puntos(:,Elecc);
123     end
124 end
125
126 Indic=ismember(P',CoordMeta','rows'); %Indica 1 si el punto esta en la
127     meta y 0 si no.
128 VecRecorrido=[VecRecorrido,P];
129 %Esta parte es OPCIONAL, es para ver el recorrido pasa por paso
130 %{
131     figure(1)
132     pointsize = 20;
133     scatter(X, Y, pointsize,PosObst,'filled','s');
134     hold on

```

```

134     scatter(CoordMeta(1,:),CoordMeta(2,:),pointsize,[1,1,1], 'filled', 's')
135     plot(VecRecorrido(1,:),VecRecorrido(2,:), 'k*')
136     hold off
137     pause(0.000001)
138     %}
139 end

```

Algoritmo 6: Algoritmo que presenta mas de una simulación de recorridos del avión partiendo desde un punto inicial fijo y para una no determinística política dada. Este presenta una imagen de la retícula con los obstáculos y una silueta en escala de grises en donde los puntos con mayor intensidad de rojo es en donde mas recorridos simulados pasaron.

```

1
2 function[SombraRecorrido]=SombrasSimulacionesPolitLinearProg(Pini,politica,
   Nsim)
3 %Pini=punto inicial y esta en forma horizontal
4 %Nsim= numero de simulaciones a realizar
5
6 %IMPORTANTE: Desactivar toda la parte grafica en 'ValueIterationAvion.m'.
7
8 tic
9 %Estados S en coordenadas X y Y
10 X= repmat(0:1:200,1,201);
11 Y= reshape(repmat(0:1:200,201,1),[1,40401]);
12 S=[X;Y];
13 %Coordenadas de los Obstaculos
14 YObsta=[repmat(110:1:135,1,5),repmat(25:1:29,1,26),repmat(90:1:110,1,5),
   repmat(75:1:79,1,21),repmat(125:1:128,1,51),repmat(50:1:100,1,4),repmat
   (150:1:175,1,5),repmat(75:1:79,1,26),repmat(135:1:160,1,5),repmat
   (100:1:104,1,26),repmat(35:1:78,1,4),repmat(75:1:78,1,40),repmat
   (125:1:129,1,26),repmat(130:1:150,1,5)];
15 XObsta=200-[reshape(repmat(171:1:175,26,1),[1,130]),reshape(repmat
   (65:1:90,5,1),[1,130]),reshape(repmat(121:1:125,21,1),[1,105]),reshape(
   repmat(90:1:110,5,1),[1,105]),reshape(repmat(100:1:150,4,1),[1,204]),
   reshape(repmat(72:1:75,51,1),[1,204]),reshape(repmat(121:1:125,26,1)
   ,[1,130]),reshape(repmat(25:1:50,5,1),[1,130]),reshape(repmat
   (96:1:100,26,1),[1,130]),reshape(repmat(40:1:65,5,1),[1,130]),reshape(
   repmat(122:1:125,44,1),[1,176]),reshape(repmat(126:1:165,4,1),[1,160]),
   reshape(repmat(50:1:75,5,1),[1,130]),reshape(repmat(71:1:75,21,1)
   ,[1,105])];
16 Obsta=[XObsta;YObsta];
17 PosObst=(ismember(S',Obsta','rows'))';
18 %Coordenadas de la Meta
19 CoordMeta=[repmat(155:1:157,1,3);reshape(repmat(155:1:157,3,1),[1,9])]; %
   Coordenadas de la meta
20
21 %El siguiente vector cuenta las veces que uno de los recorridos simulados
22 %paso por uno de los puntos del espacio de estados S.

```

```

23 SumaPosiciones=zeros(1,length(PosObst) );
24 for i=1:Nsim
25     VecRecorrido=simulacionAvionPolitLinearProg(Pini,politica); %Puntos
        del recorridos
26     PosRecorrido=(ismember(S',VecRecorrido','rows'))'; %Posición de cada
        puntos del recorrido en S.
27     SumaPosiciones=SumaPosiciones+PosRecorrido;
28 end
29 SombraRecorrido=SumaPosiciones; %Esta genera la silueta roja que representa
        todos los recorridos.
30
31 %En esta parate se grafican las simulaciones.
32 pointsize = 20;
33 %el siguiente vector tiene el objetivo de generar una escala de rojos
34 colorEscalaRojo=((max(SumaPosiciones)-SumaPosiciones)')*(0.9/max(
        SumaPosiciones)).*((SumaPosiciones~=0)')+((SumaPosiciones==0)');
35 hold on
36 scatter(X, Y, pointsize, [ones(length(SumaPosiciones),1),colorEscalaRojo,
        colorEscalaRojo],'filled','s'); %Recorridos simulados
37 scatter(XObsta, YObsta, pointsize, [zeros(length(XObsta),1),zeros(length(
        XObsta),1),zeros(length(XObsta),1)],'filled','s'); %Obstaculos
38 scatter(CoordMeta(1,:),CoordMeta(2,:), pointsize, [zeros(length(CoordMeta
        (1,:)),1),ones(length(CoordMeta(1,:)),1),zeros(length(CoordMeta(1,:)),1)
        ],'filled','s');
39 scatter(Pini(1),Pini(2), pointsize, [0,0,1],'filled','s'); %Punto inicial
40 title([ num2str( Nsim ) ' simulaciones empezando en el punto azul'])
41 hold off
42 toc

```