TRABAJO DE FIN DE MÁSTER

# Control de producción en invernadero mediante aprendizaje por refuerzo

Documento:
Memoria

Autor:
Wenjie Yin

Director:
Bernardo Morcego Seix

Titulación:
Máster universitario en Ingeniería de Sistemas Automáticos y Electrónica Industrial

Convocatoria:
Primavera 2022

## Resumen

En este proyecto, se estudia, diseña y simula agentes de aprendizaje profundo por refuerzo para controlar el clima interior de un invernadero en un modelo de invernadero de lechugas que tiene incertidumbre en la predicción del clima externo futuro.

El objetivo del proyecto es diseñar y entrenar un agente que pueda controlar de forma estable y robusta el clima interior y maximizar el uso de los recursos aportados por clima exterior, encontrando así un equilibrio que permita el crecimiento saludable de las lechugas sin consumir demasiada energía.

Para ello, en este trabajo se hablará del algoritmo de aprendizaje profundo por refuerzo, el proceso de diseño y ajuste del agente, la evaluación del agente entrenado y la comparación de los resultados con el control predictivo por modelo.

## Abstract

In this project, deep reinforcement learning agents are studied, designed and simulated to control the internal climate of a greenhouse in a lettuce greenhouse model that has uncertainty in the prediction of the future external climate.

The objective of the project is to design and train an agent that can control the indoor climate in a stable and robust way and maximize the use of the resources provided by the outdoor climate, thus finding a balance that allows the healthy growth of lettuce without consuming too much energy.

To this end, this document will discuss the deep reinforcement learning algorithm, the agent design and tuning process, the evaluation of the trained agent, and the comparison of the results with model-based predictive control.

# Agradecimientos

En primer lugar, me gustaría agradecer a mi director del trabajo fin de máster, Bernardo Mosergo Seix. Por el tiempo invertido en resolver todas las dudas que han surgido durante estos meses, siguiendo el progreso de este trabajo, ayudándome a desarrollar mi memoria y dándome muchos consejos y sugerencias.

En segundo lugar, quería dar las gracias al Farm Technology Group-WUR por haber aportado a este trabajo el modelo de invernadero, así como el controlador MPC y por haber ayudándonos a resolver las dudas sobre dichos modelos.

Por último, me gustaría agradecer a mi familia y amigos por su apoyo y compañía durante este tiempo.

# Índice

# Índice de tablas

# Índice de figuras

# 1. Introducción

La población mundial ha crecido de forma drástica en las últimas décadas, aunque el crecimiento se está desacelerando, se estima que la población mundial aumentará en 2.000 millones de personas en los próximos 30 años, llegará 9.700 millones de personas en 2050, según el estudio de Organización de las Naciones Unidas (ONU) [1].

Para alimentar esta inmensa población en 2050, las proyecciones muestran que sería necesario aumentar la producción de alimentos en un 70 % entre 2005/07 y 2050. Y la producción casi tendría que duplicarse en los países en desarrollo [2].

Por otra parte, el Grupo Intergubernamental de Expertos sobre el Cambio Climático (IPCC) declaró que es crucial y urgente cambiar la forma de gestión del uso de la tierra y métodos de producción agrícola para frenar el calentamiento global. Las actividades relativas a la agricultura, silvicultura y otros usos de la tierra representaron alrededor del 13 % de las emisiones de CO2, el 44 % de las de metano (CH4) y el 81 % de las de óxido nitroso (N2O) procedentes de las actividades humanas a nivel mundial durante 2007-2016, lo que representa el 23 % del total de emisiones antropógenas netas de gases de efecto invernadero (GEI). Si se incluyen las emisiones asociadas con las actividades previas y posteriores a la producción en el sistema alimentario mundial, se estima que las emisiones se sitúan entre el 21 % y el 37 % del total de las emisiones antropógenas netas de GEI [3].

El calentamiento global provocará más olas de calor, sequías y fuertes precipitaciones, así como la degradación de la tierra y la desertificación. Los fenómenos meteorológicos extremos serán más frecuentes y las cadenas mundiales de suministro de alimentos se verán más alteradas, lo que provocará un aumento de los precios de los alimentos y un mayor riesgo de hambre e inseguridad alimentaria.

Todo esto impone una inmensa responsabilidad en el sector agrícola para mejorar la producción de cultivos y aumentar el rendimiento por hectárea. Una solución es convertir la forma de cultivar en más inteligente con la ayuda de las tecnologías innovadoras como internet de las cosas (IoT) y tecnologías aliadas como la inteligencia artificial (IA). El reciente avance en las Tecnologías de la Información y la Comunicación (TIC) y las investigaciones relacionadas han identificado que la IoT y la IA como tecnologías clave para transformar a los métodos agrícolas modernos. Al incorporar el uso de tecnologías digitales como IoT y IA, se puede mejorar la compresión de los datos obtenidos sobre el campo, permitiendo planificar metódicamente las técnicas agrícolas con un mínimo de trabajo manual.

Según un estudio realizado por IBM [4], el 90% de todas las pérdidas de cultivos se deben al clima. Con la integración de IoT y datos meteorológicos, se podrían crear modelos meteorológicos predictivos, lo cual puede proporcionar información que ayude a los agricultores a tomar decisiones estratégicas sobre la plantación de cultivos y tomar las medidas necesarias para prevenir los daños causados por el clima extremo. Con los datos meteorológicos, se puede incluso construir un sistema de irrigación avanzado para ahorrar agua y evitar el desperdicio de pesticidas mediante la predicción de la lluvia.

Sin embargo, la meteorología suele tener un comportamiento no determinista, lo que limita su previsibilidad y da como resultado previsiones inciertas. En el caso de utilizar un sistema de control automático, cuando se anticipa una entrada externa futura del sistema mediante el uso de una previsión, la incertidumbre de la previsión crea un riesgo de actuación insuficiente o excesiva del algoritmo de control, y un riesgo de bajo rendimiento.

De aquí surge el propósito del presente proyecto, que se trata de diseñar un controlador automático robusto e inteligente, que es capaz de conseguir las acciones óptimas para

lograr el mejor uso de los recursos naturales y la eficiencia de producción ante la presencia de la incertidumbre en la previsión del clima futuro.

Este proyecto es una colaboración con University of Wageningen para estudiar el modelo de invernadero y el control predictivo por modelo (MPC) que nos proporciona [5]. Y, como continuación del trabajo, diseñaremos un agente basado aprendizaje por refuerzo y lo compararemos con MPC utilizando el modelo de invernadero proporcionado como entorno de simulación.

## 1.1 Objeto

El objeto del presente trabajo es diseñar y simular un controlador mediante aprendizaje por refuerzo, con el propósito de controlar el clima de un modelo de invernadero de lechugas con incertidumbre en la predicción meteorológica. Dicho modelo de invernadero se extrae de [5].

## 1.2 Objetivo

Los objetivos principales que se pretenden lograr con la realización del presente trabajo son los siguientes:

- Control automático: el controlador diseñado será capaz de ajustar automáticamente las acciones de control dependiendo de los estados actuales y predicciones del clima futuro (con incertidumbre).
- Aprovechamiento de recursos naturales: el controlador diseñado será capaz de aprovechar máximo los recursos aportados por la naturaleza, como irradiación solar, lluvia, etc. A fin de mejorar la utilización de los recursos.
- Reducción de emisiones: con la ayuda del controlador diseñado, será capaz de conseguir la reducción de la emisión de gases de efecto invernadero en comparación con los métodos de control temprano bajo la condición de no disminuir la producción.
- Aumento de producción: con la ayuda del controlador diseñado, será capaz de conseguir el aumento de materia seca de lechuga por metro cuadrado (en unidad $kg/m^2$), en comparación con el caso de no utilizar un controlador automático.

## 1.3 Alcance

Los siguientes puntos se encuentran dentro del alcance del proyecto:

- Estudio del entorno de simulación (modelo de invernadero) *.
- Estudio y diseño del controlador mediante aprendizaje por refuerzo:
  - Estudiar y determinar la estructura del controlador.
  - Entrenamiento del controlador.
  - Análisis del comportamiento del controlador.
- Estudio del efecto de las previsiones meteorológicas inciertas sobre las acciones de control y el rendimiento.
- Realizar una comparación entre el controlador RL diseñado y control predictivo por modelo y sacar unas conclusiones sobre sus eficiencias.

- Estudio del impacto ambiental.

(*) Aunque este TFM trate de diseñar y simular un controlador mediante aprendizaje por refuerzo, el diseño del entorno de simulación (modelo de invernadero) queda excluido del trabajo.

# 2 Modelo de invernadero

En este capítulo presentaremos las expresiones matemáticas del modelo de invernadero que utilizamos y algunas simulaciones del mismo para comprobar el rendimiento, las deficiencias y los efectos sobre el cultivo en diferentes condiciones climáticas.

En comparación con el entorno real, el uso del modelo de simulación tiene las siguientes principales ventajas:

- La velocidad de simulación del modelo de invernadero es mucho mayor que el entorno en tiempo real, lo que permite acelerar el proceso de aprendizaje del controlador.
- Será más fácil modelar situaciones difíciles de probar, como el caso en clima extremo.
- La simulación es más segura, ya que no hay riesgo de dañar el hardware.

Sin embargo, el entorno de simulación no siempre es mejor que el entorno real y a menudo se requiere inviertir mucho tiempo en la modelización para obtener un entorno de simulación relativamente realista. Por lo tanto, en este trabajo no aspiramos a crear un nuevo modelo de invernadero, sino que utilizamos el modelo que se nos proporciona en [5].

## 2.1 Descripción matemática del modelo de invernadero

El modelo del invernadero que se utiliza en este trabajo es un modelo en tiempo discreto. El modelo contiene dos tipos de entradas, una entrada controlable $u(k)$ que representa las acciones del controlador y otra entrada no controlable que representa perturbaciones de clima $d(k)$.

El estado del sistema $x$ contiene cuatro variables ($x \epsilon \mathbb{R}^4$), que son: masa seca de lechuga ($W$) en $kg \cdot m^{-2}$, concentración del dióxido de carbono dentro del invernadero ($C_{CO_{2,a}}$) en $kg \cdot m^{-3}$, temperatura interior ($T_a$) en $°C$ y humedad ($C_{H_2O,a}$) en $kg \cdot m^{-3}$, respectivamente. Se representa el estado del sistema en forma de vector columna:

$$x = \left[W, C_{CO_{2,a}}, T_a, C_{H_2O,a}\right]^T$$

El controlador del invernadero tiene la capacidad de controlar la tasa de suministro de dióxido de carbono ($u_{CO_{2,o}}$) en $mg \cdot m^{-2} \cdot s^{-1}$, tasa de ventilación a través de las rejillas de ventilación ( $u_v$) en $mm \cdot s^{-1}$ y suministro de energía por el sistema de calefacción en $W \cdot m^{-2}$. Y la perturbación del clima contiene irradiancia solar ($I_o$) en $W \cdot m^{-2}$, concentración del dióxido de carbono exterior ($C_{CO_2,O}$) en $kg \cdot m^{-3}$, temperatura exterior ($T_O$) en $°C$, y humedad exterior ($C_{H_2O,O}$) en $kg \cdot m^{-3}$:

$$u(k) = \left[u_{CO_{2,o}},\ u_v,\ \ u_q\right]^T$$

$$d(k) = \left[I_0,\ C_{CO_2,O},\ T_O,\ C_{H_2O,o}\right]^T$$

La *figura 1* muestra la interacción entre estados, señales de control y perturbaciones. Donde $y(k)$ representa medidas de estados, que contiene mismas señales que $x(k)$, pero con diferentes unidades (sensor estándar). Como se puede observar, la irradiancia solar tiene una influencia directa sobre el crecimiento de lechuga, mientras que todas entradas restantes influyen en el clima interior del invernadero, lo cual también controla el crecimiento de lechuga.

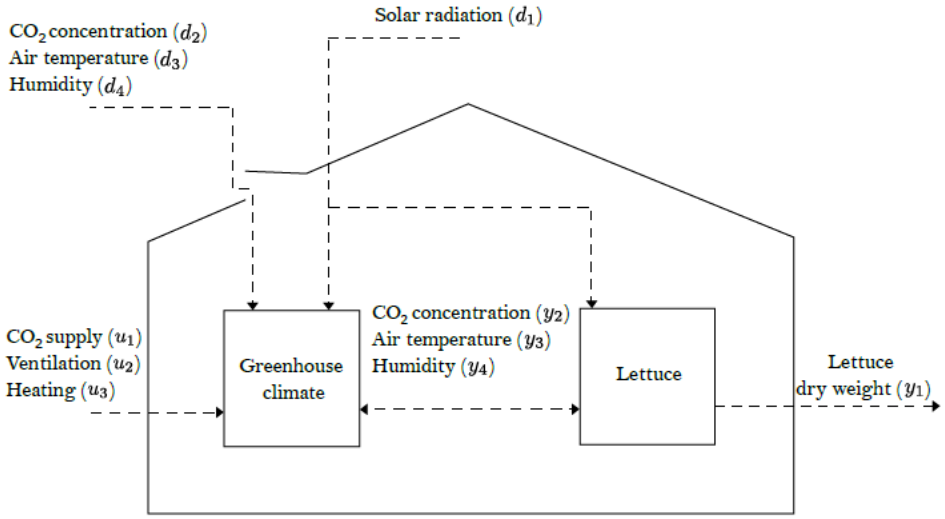



*Figura 1: Representación esquemática del modelo de invernadero (Fuente: [5])*

La transición del estado sigue una función no lineal $f$, el estado del sistema en el instante siguiente $x(k+1)$ depende del estado actual $x(k)$, perturbaciones $d(k)$ y señales de control $u(k)$:

$$x(k+1) = f\big(x(k), d(k), u(k)\big)$$

Expandiendo la función $f$, se obtiene:

$$x_1(k+1) = x_1(k) + h\left(c_{1,1}\cdot\phi_{phot,c}(k) - c_{1,2}\cdot x_1(k)\cdot 2^{\frac{x_3(k)}{10}-\frac{5}{2}}\right)$$

$$x_2(k+1) = x_2(k) + \frac{h}{c_{2,1}}\left(-\phi_{phot,c}(k) + c_{2,2}\cdot x_1(k)\cdot 2^{\frac{x_3(k)}{10}-\frac{5}{2}} + u_1(k)\cdot 10^{-6} - \phi_{vent,c}(k)\right)$$

$$x_3(k+1) = x_3(k) + \frac{h}{c_{3,1}}\Big(u_3(k) - \big(c_{3,2}\cdot u_2(k)\cdot 10^{-6} + c_{3,3}\big)\cdot\big(x_3(k) - d_3(k)\big) - c_{3,4}\cdot d_1(k)\Big)$$

$$x_4(k+1) = x_4(k) + \frac{h}{c_{4,1}}\Big(\phi_{transp,h}(k) - \phi_{vent,h}(k)\Big)$$

Donde $h$ es el periodo de muestreo, $\phi_{phot,c}(k)$ es la tasa de fotosíntesis bruta del dosel, $\phi_{vent,c}(k)$ es el intercambio de masa de $CO_2$ a través de las ventilaciones, $\phi_{transp,h}(k)$ es la transpiración del dosel y $\phi_{vent,h}(k)$ es el intercambio de masa de $H_2O$ a través de las ventilaciones, que están expresado como:

$$\phi_{phot,c}(k) = \frac{\Big(1-\exp\big(-c_{1,3}\cdot x_1(k)\big)\Big)\cdot\Big(c_{1,4}\cdot d_1(k)\cdot\big(-c_{1,5}\cdot x_3(k)^2 + c_{1,6}\cdot x_3(k) - c_{1,7}\big)\cdot\big(x_2(k)+c_{1,8}\big)\Big)}{c_{1,4}\cdot d_1(k) + \big(-c_{1,5}\cdot x_3(k)^2 + c_{1,6}\cdot x_3(k) - c_{1,7}\big)\cdot\big(x_2(k) - c_{1,8}\big)}$$

$$\phi_{vent,c}(k) = \big(u_2(k)\cdot 10^{-3} + c_{2,3}\big)\cdot\big(x_2(k) - d_2(k)\big)$$

$$\phi_{transp,h}(k) = c_{4,1}\cdot\Big(1-\exp\big(-c_{1,3}\cdot x_1(k)\big)\Big)\cdot\left(\frac{c_{4,3}}{c_{4,4}\cdot\big(x_3(k)+c_{4,5}\big)}\cdot\exp\left(\frac{c_{4,6}\cdot x_3(k)}{x_3(k)+c_{4,7}}\right) - x_4(k)\right)$$

$$\phi_{vent,h}(k) = \big(u_2(k)\cdot 10^{-3} + c_{2,3}\big)\cdot\big(x_4(k) - d_4(k)\big)$$

Por otra parte, se obtienen las medidas $y(k)$ aplicando una conversión sobre el estado $x(k)$:

$$y(k) = g\big(x(k)\big)$$

$$y_1(k) = 10^3 \cdot x_1(k)$$

$$y_2(k) = \frac{c_{2,4} \cdot \big(x_3(k) + c_{2,5}\big)}{c_{2,6} \cdot c_{2,7}} \cdot x_2(k)$$

$$y_3(k) = x_3(k)$$

$$y_4(k) = \min\left(100\,, \frac{c_{2,4} \cdot \big(x_3(k) + c_{2,5}\big)}{\exp\left(\frac{c_{4,8} \cdot x_3(k)}{x_3(k) + c_{4,9}}\right)} \cdot x_4(k)\right)$$

Todos los parámetros $c_{i,j}$ que aparecen en las ecuaciones anteriores son valores constantes, que están definidos en la tabla siguiente:

| Constante | Descripción | Valor | Unidad |
|---|---|---|---|
| $c_{1,1}$ | Factor de producción | $0.544$ | - |
| $c_{1,2}$ | Tasa de respiración | $2.65 \cdot 10^{-7}$ | $s^{-1}$ |
| $c_{1,3}$ | Superficie efectiva de dosel | $53$ | $\frac{m^2\{leaf\}}{kg\{dw\}}$ |
| $c_{1,4}$ | Eficiencia de uso de la luz | $3.55 \cdot 10^{-9}$ | $\frac{kg\{CO_2\}}{J}$ |
| $c_{1,5}$ | Influencia de la temperatura en la fotosíntesis | $5.11 \cdot 10^{-6}$ | $\frac{m}{s \cdot °C^2}$ |
| $c_{1,6}$ | Influencia de la temperatura en la fotosíntesis | $2.3 \cdot 10^{-4}$ | $\frac{m}{s \cdot °C}$ |
| $c_{1,7}$ | Influencia de la temperatura en la fotosíntesis | $6.29 \cdot 10^{-4}$ | $\frac{m}{s}$ |
| $c_{1,8}$ | Punto de compensación de dióxido de carbono | $5.2 \cdot 10^{-5}$ | $\frac{kg\{CO_2\}}{m^3\{air\}}$ |
| $c_{2,1}$ | Capacidad de $CO_2$ del invernadero | $4.1$ | $\frac{m^3\{air\}}{m^2\{gh\}}$ |
| $c_{2,2}$ | Coeficiente de respiración | $4.87 \cdot 10^{-7}$ | $s^{-1}$ |
| $c_{2,3}$ | Ventilación a través de la cubierta | $7.5 \cdot 10^{-6}$ | $\frac{m}{s}$ |
| $c_{2,4}$ | | $75560$ | |
| $c_{2,5}$ | Conversión de Celsius a Kelvin | $273.15$ | $K$ |
| $c_{2,6}$ | Presión (se supone que es 1 atm) | $101325$ | $Pa$ |
| $c_{2,7}$ | Masa molar de $CO_2$ | $44.01 \cdot 10^{-3}$ | $\frac{kg}{mol}$ |

| | | | |
|---|---|---|---|
| $c_{3,1}$ | Capacidad calorífica efectiva del aire del invernadero | $3 \cdot 10^4$ | $\frac{J}{m^2\{gh\} \cdot ℃}$ |
| $c_{3,2}$ | Capacidad calorífica por volumen de aire del invernadero | 1290 | $\frac{J}{m^3\{gh\} \cdot ℃}$ |
| $c_{3,3}$ | Transferencia de calor total a través de la cubierta | 6.1 | $\frac{W}{m^2\{gh\} \cdot ℃}$ |
| $c_{3,4}$ | Coeficiente de carga de calor debido a la radiación solar | 0.2 | - |

*Tabla 1: Parámetros del modelo de invernadero*

## 2.2 Simulaciones sobre el modelo de invernadero

En esta sección realizamos una serie de simulaciones con el fin de observar el efecto de cada entrada en el estado del invernadero.

Los datos meteorológicos $d(k)$ utilizados en las simulaciones son datos reales que se presentan en [6]. Estos datos se recogen durante los experimentos realizados en el invernadero llamado "*the Venlow Energy greenhouse*" que se encuentra en Bleiswijk, Holanda. Utilizaremos estos mismos datos meteorológicos en todas las simulaciones posteriores.

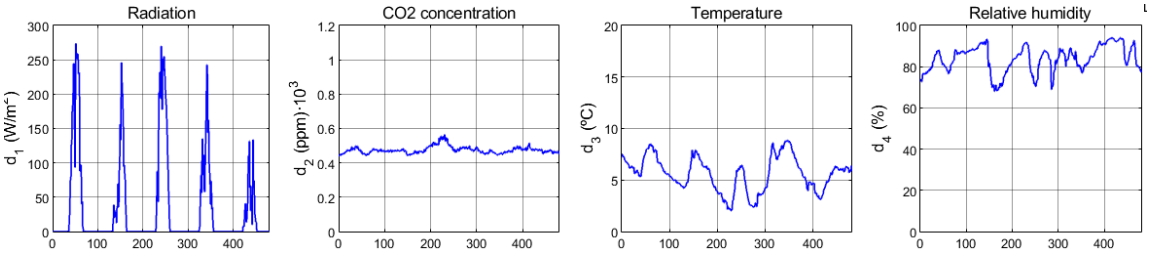



*Figura 2: Datos meteorológicos nominales*

En primer lugar, simulamos cómo se vería afectado el crecimiento de las lechugas sólo por las condiciones meteorológicas externas, sin intervención de las acciones controlables.

La Figura 3 representa una simulación de 5 días, el periodo de muestreo es 15 minutos, por lo que 96 pasos de simulación equivalen un día de simulación del invernadero.

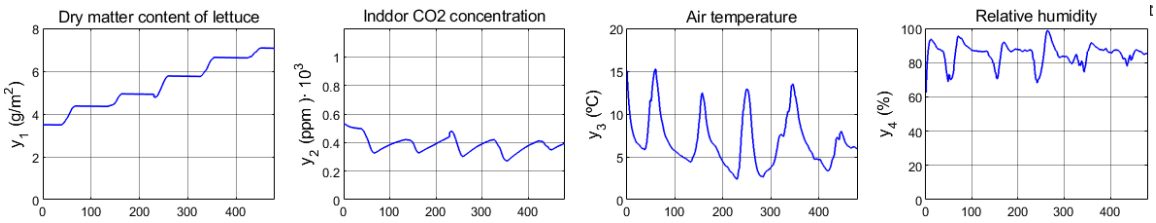



*Figura 3: Simulación del invernadero sin entradas controlables*

Como se muestra en la Figura 3, la lechuga sigue manteniendo el crecimiento cuando sólo se utiliza el clima externo como entrada. La masa seca de lechuga es 3.5 g/m$^2$ en el instante inicial de simulación, después de 24 horas la masa aumentó a 4.4 g/m$^2$; luego 4.9 g/m$^2$ a los 2 días, 5.8 g/m$^2$ a los 3 días, 6.5 g/m$^2$ a los 4 días y 7.1 g/m$^2$ a los 5 días.

Hemos comprobado que la masa de la lechuga sólo crece en presencia de la irradiancia solar, y cuanta más irradiancia solar recibe, más rápido crece.

La concentración de $CO_2$ en el invernadero cambia constantemente cuando la concentración de $CO_2$ en el exterior se mantiene básicamente constante en 500 ppm. Esto se debe a que el crecimiento de la lechuga consume dióxido de carbono.

Por otro lado, la temperatura interior tiende a ser similar a la exterior por la noche. Sin embargo, durante el día, la temperatura interior se mantiene más alta que en el exterior, ya que el interior puede aprovechar y conservar mejor el calor aportado por el sol.

Al mismo tiempo, existe una clara correlación entre la temperatura y la humedad relativa. Como que el aire puede contener más moléculas de agua a mayor temperatura, el aumento de la temperatura con la misma cantidad de moléculas de agua en el aire da lugar a una menor humedad relativa. Sin embargo, el cambio es significativamente más fuerte en el interior en comparación con el cambio de la humedad relativa en el exterior, que es causada por la transpiración del cultivo.

Posteriormente, se realizaron varias simulaciones con la introducción manual de las entradas controlables.

En primer lugar, introducimos sólo la inyección de $CO_2$ y mantenemos su valor constante en 0,5 mg/($m^2$s). Se puede observar en la *Figura 4*, con la inyección de $CO_2$ aumenta la concentración de $CO_2$ del invernadero y afecta significativamente al crecimiento de la lechuga. En comparación con la simulación sin ninguna entrada controlable *Figura 3*, la producción a los cinco días aumentó de 7.1 g/$m^2$ a 11 g/$m^2$. En cambio, la inyección de $CO_2$ tuvo un efecto insignificante sobre la temperatura y la humedad relativa del invernadero.

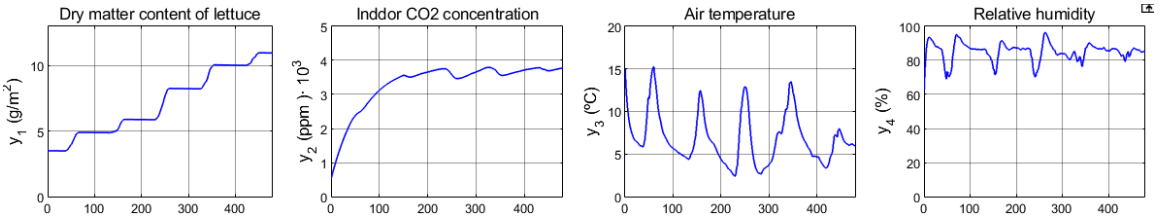



*Figura 4: Simulación del invernadero con inyección de $CO_2$*

En la segunda simulación, cambiamos la entrada controlable por únicamente ventilación y dimos una velocidad de ventilación constante de 3 mm/s. Se observa que la ventilación acelera el equilibrio entre el clima interior y el exterior.

La producción de las lechugas aumentó ligeramente de 7.1 g/$m^2$ a 7.5 g/$m^2$, esto se debe a que en los datos meteorológicos que usamos para simulación, la concentración de dióxido de carbono es mayor en el exterior que en el interior y, con la ayuda de ventilación puede llevar el dióxido de carbono exterior al invernadero. Así pues, aumenta la producción.

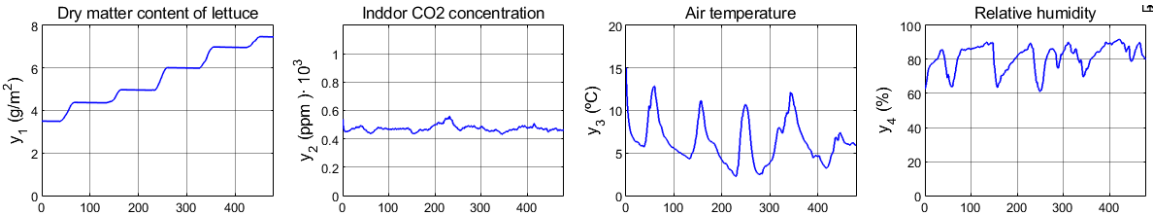



*Figura 5: Simulación del invernadero con ventilación*

En esta tercera simulación, ajustamos la entrada controlable sólo al sistema de calefacción y la fijamos a un valor constante de 50 W/$m^2$.

Se puede observar *Figura 6* que el sistema de calefacción aumenta significativamente la temperatura del invernadero y la mayor temperatura ayuda a aumentar la producción de las lechugas de 7.1 g/m$^2$ a 8.0 g/m$^2$.

La temperatura también afecta a la humedad relativa y la mayor temperatura reduce la humedad relativa. Además, como la lechuga crecía más rápido, esto significaba que también se consumía más dióxido de carbono y se puede observar que hay una mayor disminución de la concentración de dióxido de carbono interior del invernadero.

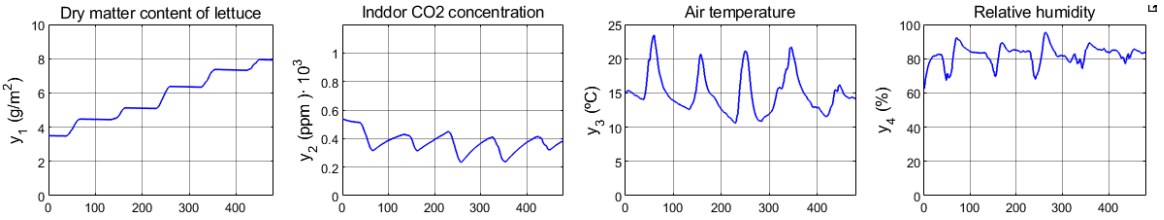



*Figura 6: Simulación del invernadero con calefacción*

Como podemos interpretar por los ejemplos simulados anteriormente, tanto la temperatura como la concentración de $CO_2$ tienen un impacto significativo en el crecimiento de la lechuga. Aunque la humedad relativa no afecta directamente al crecimiento del cultivo, una mayor humedad relativa puede, de hecho, favorecer la aparición de las enfermedades fúngicas como el estafilococo o el moho. El control de la humedad requiere el uso de ventilación, lo que afecta directamente a la temperatura y a la concentración de $CO_2$ del invernadero. Esto hace que resulte muy difícil de encontrar el control óptimo.

# 3 Control predictivo por modelo

En este capítulo presentamos brevemente qué es el MPC y su modelo matemático aplicado en el modelo de invernadero.

## 3.1 Introducción del control predictivo por modelo

El método de control predictivo basado en modelo (MPC) [7] se caracteriza por contener un modelo del sistema de estudio, para tratar de predecir el comportamiento futuro del mismo. El método MPC emplea un horizonte de predicción $N_p$, para el cual se calcula las acciones de control óptimas minimizando una función de coste $V$, utilizando para controlar únicamente la primera componente que corresponde al instante actual.

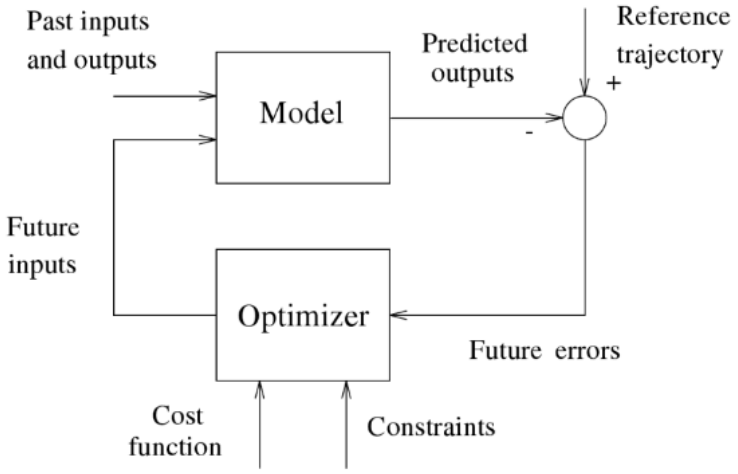



*Figura 7: Estructura básica del MPC (Fuente: [7])*

Una de las características que destacan de este control óptimo, es el elevado número de entradas y salidas con las que es capaz de trabajar, así como la definición de restricciones del problema de control, donde se puede limitar los valores del estado y la acción de control. Esto presenta una gran ventaja en comparación con otras estrategias de control, ya que a la hora de implementarlo en un sistema real, este tendrá limitaciones en las derivadas de las variables (como aceleración o velocidad) o potencia consumida, etc. Que se contemplan con estas restricciones.

En el estudio sobre el sistema de invernaderos, las temperaturas demasiado altas o bajas pueden afectar al crecimiento de las lechugas, mientras que también debe evitarse una humedad interior muy alta, ya que puede favorecer la aparición de enfermedades fúngicas como el estafilococo o el moho. Por lo tanto, estos estados deben ser limitados por MPC.

## 3.2 Control predictivo por modelo aplicado en invernadero [5]

La forma genérica del modelo de controlador se caracteriza por:

$$\hat{x}(k+1) = f\left(\hat{x}(k), u(k), \hat{d}(k)\right)$$

$$\hat{y}(k) = g\left(\hat{x}(k)\right)$$

Donde los términos que llevan un " ̂ " representan las estimaciones.

Para el MPC, la estimación de perturbación del clima exterior $\hat{d}(k)$ sigue una función de distribución uniforme:

$$\hat{d}_j(k) \sim Unif\left(\mu_{\hat{d}_j}(k), \sigma_{\hat{d}_j}(k)\right), \text{ para } j = 1,2,3,4$$

Donde el promedio de la perturbación estimada coincide con la perturbación real.

$$\mu_{\hat{d}_j}(k) = d_j(k)$$

Y la desviación típica $\sigma_{\hat{d}_j}(k)$ está expresada:

$$\sigma_{\hat{d}_j}(k) = \frac{1}{12}\left(\hat{d}_{j,max}(k) - \hat{d}_{j,min}(k)\right)^2$$

Donde

$$\hat{d}_{j,max}(k) = (1+\gamma)\cdot d_j(k)$$
$$\hat{d}_{j,min}(k) = (1-\gamma)\cdot d_j(k)$$

Y se supone que $d_j(k) \geq 0$ y, $\gamma$ es un escalar con valor entre cero y uno que indica el grado de incertidumbre.

La suposición de que $\hat{d}_j(k)$ se distribuye uniformemente se puede justificar por el hecho de que esta distribución tiene un soporte compacto, que es realista para las previsiones meteorológicas a plazos relativamente cortos. La predicción de irradiancia solar $\hat{d}_1(k)$se ilustra gráficamente en la *Figura. 8* como ejemplo. En esta gráfica, $d_1(k)$ es la irradiancia solar verdadera que se define como el valor medio de la predicción $\hat{d}_j(k)$. Los valores mínimos $\hat{d}_{j,min}(k)$ y máximos $\hat{d}_{j,max}(k)$ determinan la varianza de la irradiancia solar uniformemente distribuida.

Cada señal individual de $\hat{d}(k)$ se modela de este modo y se suponen que los patrones generales sobre las previsiones meteorológicas futuras son conocidos.

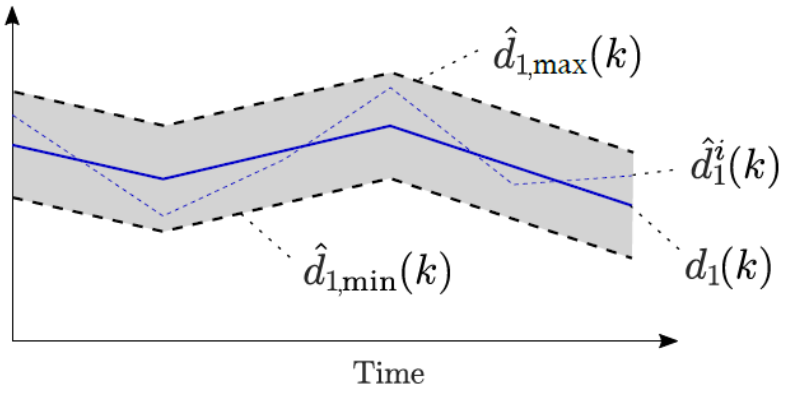



*Figura 8: Ilustración de la incertidumbre de perturbación (Fuente: [5])*

Un reto es que podemos extraer infinitas trayectorias diferentes en cada realización de las perturbaciones. Como muestra en la *Figura 8*, $\hat{d}_1^i(k)$ es una de estas trayectorias y todas las demás se encuentran en la zona gris. Por lo tanto, desde un punto de vista computacional es imposible tener en cuenta todas las trayectorias en el controlador. Una solución es considerar sólo los peores casos con el controlador (por ejemplo, $\hat{d}_{1,min}(k)$y $\hat{d}_{1,max}(k)$). Sin embargo, no sabemos que un valor máximo de perturbación resultará una salida máxima del modelo $\hat{y}(k)$, debido a que $f(\cdot)$ y $g(\cdot)$ son funciones no lineales. La opción que se toma en este trabajo es extraer $N_s$ muestras aleatorias (trayectoria) de cada distribución uniforme y luego tener en cuenta todas estas trayectorias en el controlador.

### 3.2.1 Problema de optimización

El problema de optimización que está formulado en esta sección se usa el modelo de invernadero definido en el capítulo 2.1.

Se asume que, para cada instante de tiempo, el estado $x(k)$ es medible, es lo mismo decir que $\hat{x}(k_0) = x(k_0)$.

Para cada paso de tiempo $k_0$, se resuelve el problema de optimización siguiente:

$$\min_{u(k)} \sum_{i=1}^{N_s} \sum_{k=k_0}^{k_0+N_p} V\left(u(k), \hat{y}^i(k)\right)$$

Donde $N_p$ es el horizonte de predicción que determina hasta qué lejos se propaga hacia delante en el tiempo. El valor $N_s$ es un numero real que representa el numero de trayectorias que se cogen de la distribución normal. Y, siendo $V\left(u(k), \hat{y}^i(k)\right)$ la función de coste que se define como sigue:

$$V\left(u(k), \hat{y}^i(k)\right) = -q_{\hat{y}_1} \cdot \hat{y}_1{}^i(N_p) + \sum_{j=1}^{3} q_{u_j} \cdot u_j(k)$$

Donde $q_{\hat{y}_1}$ y $q_{u_j}$ son valores reales ajustables.

Y las restricciones del problema de optimización son:

$$u_{min} \leq u(k) \leq u_{max}$$

$$|u(k) - u(k-1)| \leq \delta u$$

$$\hat{y}_{min}(k) \leq \hat{y}^i(k) \leq \hat{y}_{max}(k) \text{ , para } i = 1, \dots, N_s \text{ y } k = k_0, \dots, k_0 + N_p$$

Donde $u_{min}$ y $u_{max}$ son el límite inferior y superior de las señales de control:

$$u_{min} = (0\ 0\ 0)^T$$

$$u_{max} = (1.2\ \ 7.5\ \ 150)^T$$

$$\delta u = \frac{1}{10} u_{max}$$

Y el límite inferior y superior de las salidas:

$$\hat{y}_{min}(k) = \left(0\ \ 0\ \ f_{\hat{y}_{3,min}}(k)\ \ 0\right)^T$$

$$\hat{y}_{max}(k) = \left(\infty\ \ 1.6\ \ f_{\hat{y}_{3,max}}(k)\ \ 70\right)^T$$

Siendo la temperatura mínima $\hat{y}_{3,min}$ y máxima $\hat{y}_{3,max}$ de la salida se limita dependiendo de si es por la noche o por el día, como se muestra en la *Figura 9*. Uno de los resultados de limitar la temperatura en el tiempo es que el controlador no necesita trabajar tanto por la noche para mantener una temperatura a un nivel relativamente alto, cuando la temperatura exterior es relativamente baja. Como el sol calienta el invernadero durante el día, se necesita menos calefacción adicional durante este periodo para conseguir las temperaturas relativamente altas dentro del invernadero y, por tanto, se puede ahorrar energía.

$$f_{\hat{y}_{3,min}}(k) = \begin{cases} 15, & si\ \ d_1(k_0) < 10 \\ 20, & otros\ casos \end{cases}$$

$$f_{\hat{y}_{3,max}}(k) = \begin{cases} 20, & si\ \ d_1(k_0) < 10 \\ 25, & otros\ casos \end{cases}$$

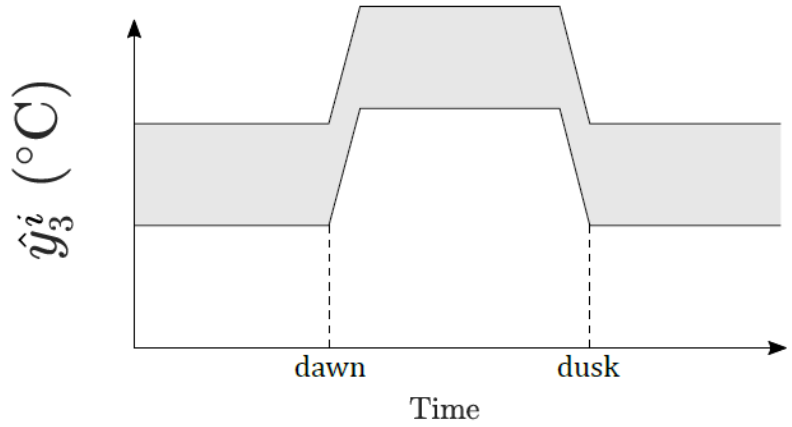



*Figura 9: rango de temperatura permitida a lo largo del día (Fuente: [5])*

Los resultados del comportamiento del control predictivo se muestran en el capítulo 6.

# 4 Aprendizaje por refuerzo

## 4.1 Introducción del aprendizaje por refuerzo

El aprendizaje por refuerzo (RL) es una rama de aprendizaje automático. A diferencia con los problemas de aprendizaje supervisado y aprendizaje no supervisado, la principal característica del RL es aprender a partir de las interacciones con el entorno. Mediante la interacción, el agente aprende continuamente según los refuerzos recibidos del entorno.

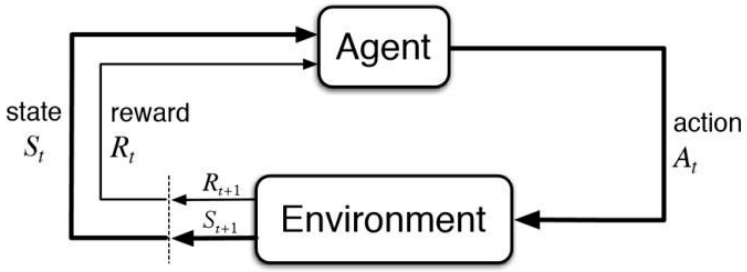



*Figura 10: Interacción Agente-Entorno (Fuente:[8])*

Los términos fundamentales del aprendizaje por refuerzo se forman por acción, estado y refuerzo. La acción $(A)$ representa el conjunto de acciones posibles que puede ejercer el agente sobre el entorno, el estado $(S)$ representa el conjunto de observaciones posibles percibidas por el agente desde el entorno, y el refuerzo $(R)$ representa todos refuerzos posibles recibidos por el agente después de interactuar con el entorno.

El problema central del aprendizaje por refuerzo es la selección de la acción $a$ $(a \in A)$ del agente en cuando se encuentra en el estado $s$ $(s \in S)$, esto recibe el nombre de política $(\pi)$.

La política puede ser determinista o estocástica. Cuando la política es determinista, significa que la política selecciona la acción directamente en función de estado $s$. Cuando la política es estocástica, la política $\pi(a \mid s)$ selecciona la acción en función de la distribución de probabilidad de cada acción bajo la condición de que el estado es $s$.

- Para la política determinista: $\pi(s) = a$
- Para la política estocástica: $\pi(a|s) = \Pr(A = a \mid S = s)$

Después de ejercer una acción $a$, el estado pasa de un estado viejo $s$ a un estado nuevo $s'$, esto se denomina transición del estado. La transición del estado puede tener estocasticidad, y esta estocasticidad viene del entorno:

$$p(s'|s,a) = \Pr(S' = s'|S = s, A = a)$$

En el mismo instante de la transición, el agente recibe un refuerzo $R$, que es un valor real que indica la bondad de la acción que ejerce el agente en el instante anterior. El objetivo del problema del aprendizaje por refuerzo es conseguir el mayor refuerzo acumulado, lo cual también se denomina beneficio $G$, que se define como:

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots + \gamma^n R_{t+n}$$

Donde $\gamma$ es un factor de descuento, que su valor se encuentra dentro del intervalo $[0, 1]$, y representa la importancia con la que se valoran los refuerzos futuros.

Debido a la incertidumbre del entorno, no es posible estimar con precisión los refuerzos futuros con mucha anticipación. Por lo tanto, cuanto mayor sea $n$, el valor de refuerzo correspondientes $\gamma^n R_{t+n}$ se acercará más a cero.

Bajo esta circunstancia, en el instante de tiempo $t$, los refuerzos $R_t, R_{t+1}, \dots, R_n$ no son determinísticos, ya que el refuerzo $R_i$ depende del estado $S_i$ y acción $A_i$, y ambos pueden ser muestreados aleatoriamente en base a funciones de distribución de probabilidad. Dado que $R_t, R_{t+1}, \dots, R_n$ son estocásticos, el beneficio $G_t$ también será estocástico, puesto que $G_t$ depende de $R_t, R_{t+1}, \dots, R_n$.

Para poder evaluar la bondad del agente al elegir acción $a$ en estado $s$, bajo la política $\pi$ se utiliza la función de acción-valor, que se expresa como:

$$Q_\pi(s_t, a_t) = \mathbb{E}_{S_{t+1}, A_{t+1}, \dots, S_n, A_n}(G_t \mid S_t = s_t, A_t = a_t)$$

Para la función de acción-valor, se considera que $s_t$ y $a_t$ son valores observados, y $S_{t+1}, A_{t+1}, \dots, S_n, A_n$ son estocásticos. Al calcular la esperanza matemática del $G_t$ bajo la condición de estado $S_t = s_t$ y acción $A_t = a_t$, se elimina la dependencia con $S_{t+1}, A_{t+1}, \dots, S_n, A_n$. De esta forma, $Q_\pi(s_t, a_t)$ solo se depende del estado $s_t$, acción $a_t$, política $\pi$ y función de transición del estado $p$, con eso, se elimina la estocasticidad de $G_t$.

Por otra parte, se puede calcular la esperanza matemática de la función de acción-valor para conocer la bondad de la situación en estado $s$, bajo la política $\pi$. Esto se denomina función de estado-valor, que se define como:

$$V_\pi(s_t) = \mathbb{E}_A\big(Q_\pi(s_t, A)\big)$$

Según la definición, para la política estocástica la acción sigue la distribución de probabilidad $A \sim \pi(\cdot \mid s_t)$. De esta forma, la función de estado-valor también se puede expresar como:

- Para acción discreta: $V_\pi(s_t) = \sum_a \pi(a|s_t)\, Q_\pi(s_t, a)$
- Para acción continúa: $V_\pi(s_t) = \int \pi(a|s_t)\, Q_\pi(s_t, a)\, da$

Así mismo, la forma de evaluar la bondad de la política $\pi$ es calcular la esperanza matemática de la función de estado-valor:

$$\mathbb{E}_S\big(V_\pi(S)\big)$$

## 4.2 Tipos de aprendizaje por refuerzo profundo

### 4.2.1 Aprendizaje por refuerzo basado en valor [9]

La función de acción–valor $Q_\pi(s_t, a_t)$ nos indica la bondad de realizar la acción $a_t$ en estado $s_t$ y bajo política $\pi$, cuando mayor sea su valor, mejor será la acción realizada. Para ello, introducimos la función de acción-valor óptima $Q^*$, que se define como sigue:

$$Q^*(s_t, a_t) = \max_\pi Q_\pi(s_t, a_t)$$

La $Q^*$ indica que, el resultado de elegir acción $a_t$ en estado $s_t$ por la función $Q_\pi(s_t, a_t)$ siempre tendrá un valor menor o igual que $Q^*(s_t, a_t)$ para cualquiera política que elegimos. Es lo mismo decir que, bajo el estado $s_t$, $Q^*$ valora la bondad de realizar la acción $a_t$ independientemente de la política, por lo que $Q^*$ puede guiar al agente para tomar las mejores acciones en cada estado.

Evidentemente, la mejor acción $a^*$ que debe tomar el agente es la acción que maximiza $Q^*$:

$$a^* = \operatorname*{argmax}_a Q^*(s_t, a_t)$$

Desafortunadamente, no existe una formulación matemática para determinar esta función de acción-valor óptima. Una alternativa es usar Deep Q-Network (DQN) para crear una red neuronal $Q(s_t, a_t; w)$ que aproxima $Q^*(s_t, a_t)$.

Donde $w$ son parámetros de la red neuronal, que en principio del problema $w$ pueden ser valores aleatorios, y a medida que se progresa en el aprendizaje, se actualizan los valores de $w$ mediante el método de diferencia temporal (TD).

El proceso de método de diferencia temporal consiste lo siguiente:

1. En instante $t$, el modelo hace una predicción sobre $Q(s_t, a_t; w_t)$.
2. Llega el instante $t+1$, se observa el refuerzo obtenido $r_t$ y el nuevo estado $s_{t+1}$. Al tener el estado $s_{t+1}$, se puede calcular la acción $a_{t+1}$ usando DQN:
$$Q(s_{t+1}, a_{t+1}; w_t) = \max_a Q(s_{t+1}, a; w_t)$$
3. Calcular TD objetivo $y_t$.
$$y_t = r_t + \gamma \cdot Q(s_{t+1}, a_{t+1}; w)$$
4. Calcular la función de coste $L_t$, donde $(Q(s_t, a_t; w) - y_t)$ es el error TD.
$$L_t = \frac{1}{2}(Q(s_t, a_t; w) - y_t)^2$$
5. Actualizar parámetros $w_{t+1}$ usando descenso de gradiente con el fin de reducir la función de coste $L$:
$$w_{t+1} = w_t - \alpha \cdot \frac{\partial L_t}{\partial w}\Big|_{w=w_t}$$

### 4.2.2 Aprendizaje por refuerzo basado en política [9]

La función de política $\pi(a|s)$ es una función de distribución de probabilidad que coge los estados $s$ como entrada y da la probabilidad de cada acción como salida.

Una manera de hacerlo es presentar en forma de tabla que registra la probabilidad de cada acción para cada estado. Sin embargo, si el número de estados o acciones posibles es muy grande o incluso infinito, no es factible representarlo en forma de tabla.

La alternativa es diseñar una función que aproxima la función de política $\pi$. Una posible forma es usar redes neuronales para diseñar una red de política $\pi(a|s\,;\theta)$ para apriximar la función de política $\pi(a|s)$.

Donde $\theta$ representa parámetros de la red neuronal que inicialmente son valores aleatorios, y se actualizan sus valores mediante el aprendizaje.

Del apartado anterior, la función de estado-valor para acción discreta viene definida como:
$$V_\pi(s_t) = \mathbb{E}_A\big(Q_\pi(s_t, A)\big) = \sum_a \pi(a|s_t)\, Q_\pi(s_t, a)$$

Con la red neuronal, se aproxima la función de política $\pi(a|s)$ por la red política $\pi(a|s\,;\theta)$. De esta forma, la función de estado-valor se convierte en:
$$V(s_t;\theta) = \sum_a \pi(a|s\,;\theta)\, Q_\pi(s_t, a)$$

Se conoce que la función de estado-valor se utiliza para evaluar la bondad del estado $s_t$. Dado el estado $s_t$ , cuando mejor sea este estado, mayor será $V(s_t;\theta)$. Por lo tanto, se puede ajustar los parámetros $\theta$ para maximizar $V(s_t;\theta)$.

Basado en esta idea, se define la función objetivo como la esperanza de $V(S;\theta)$, y se la denomina como $J(\theta)$:

$$J(\theta) = \mathbb{E}_S\big(V(S;\theta)\big)$$

Así pues, la única variable que queda es $\theta$, y la función $J(\theta)$ representa la evaluación sobre la red de política. Cuando mayor es $J(\theta)$, mejor es la red de política.

Por lo tanto, el aprendizaje por refuerzo basado en política consiste en aprender valor de $\theta$ para maximizar $J(\theta)$. Y el aprendizaje de $\theta$ se basa en método de gradiente ascendiente de política, que en consiste lo siguiente:

En cada instante, el agente observa un nuevo estado $s_t$, $s_t$ se obtiene mediante muestreo aleatorio de la distribución de probabilidad. Una vez observado $s_t$, calcula la derivada parcial de $V(S;\theta)$ respecto $\theta$, y se actualiza $\theta$ por ascenso de gradiente:

$$\theta_{t+1} = \theta_t + \beta \frac{\partial\, V(s;\theta)}{\partial\, \theta}$$

Donde $\beta$ es el parámetro de aprendizaje que se ajusta manualmente. Y el término $\frac{\partial\, V(S;\theta)}{\partial\, \theta}$ recibe el nombre de gradiente de política, que se representa de las formas siguientes:

- Forma 1: $\frac{\partial\, V(S;\theta)}{\partial\, \theta} = \sum_a \frac{\partial\, \pi(a|s;\theta)}{\partial\, \theta} Q_\pi(s,a)$
- Forma 2: $\frac{\partial\, V(S;\theta)}{\partial\, \theta} = \mathbb{E}_{A\sim\pi(\cdot|s;\theta)}\left(\frac{\partial \log \pi(A|s;\theta)}{\partial\, \theta} Q_\pi(s,A)\right)$

Las dos formas son equivalentes, donde la primera forma solo funciona para acciones discretas mientras que la segunda funciona tanto para acciones discretas como continuas.

No obstante, $Q_\pi$ es una función desconocida, se la puede estimar con formas siguientes:

- Opción 1: método de REINFORCE
  - Progresar el problema hasta el final y generar la trayectoria:
    $$s_1, a_1, r_1, s_2, a_2, r_2, \dots, s_T, a_T, r_T$$
  - Calcular el beneficio
    $$g_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$$
  - Dado que $Q_\pi(s_t,a_t) = \mathbb{E}(G_t)$, se puede usar el valor de $G_t$ observado $(g_t)$ para aproximar $Q_\pi(s_t,a_t)$.

- Opción 2: Aproximar $Q_\pi$ usando una red neuronal, de esta manera, habrán dos redes neuronales. Una sirve para aproximar la función de política $\pi$ y otra para aproximar la función de acción-valor $Q_\pi$. Esto recibe el nombre del método de actor-crítico.

### 4.2.3 Método de Actor-crítico [9]

El método de actor-crítico consiste en unir el aprendizaje por refuerzo basado en valor y en política. Donde el actor trata de una red de política que controla las acciones del agente y

el crítico se basa en la red de valor que evalúa la bondad de la acción ejercida por la red del actor.

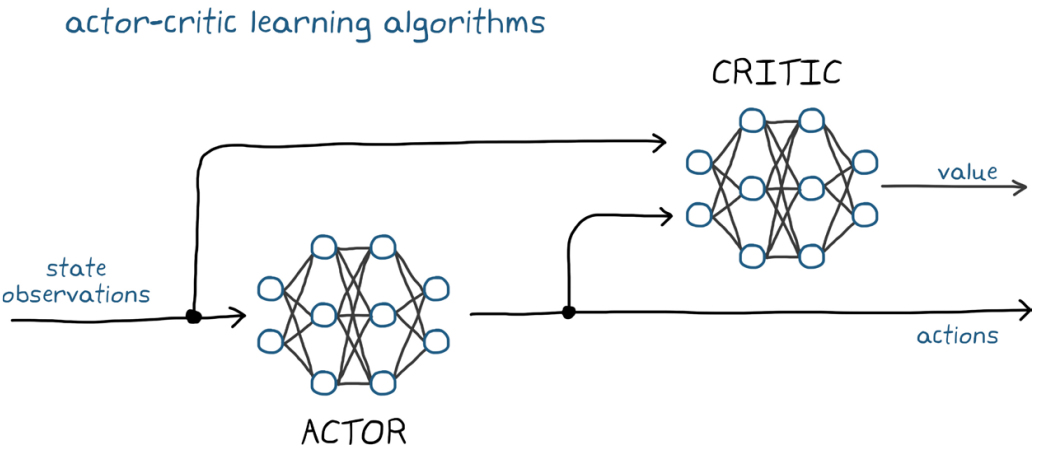



*Figura 11: Estructura del método de actor-crítico (Fuente: [10])*

**Construcción de redes de valor y política**

$$V_\pi(s) = \sum_a \pi(a|s) \cdot Q_\pi(s,a)$$

La función de estado valor $V_\pi(s)$ es la esperanza de la función de valor de acción $Q_\pi$, donde la función de política $\pi(a|s)$ controla al agente para tomar acciones, y la función $Q_\pi(s,a)$ evalúa la bondad de acción. No obstante, no conocemos ninguna de estas dos funciones, pero podemos crear ambas funciones con redes neuronales y luego hacer que las redes aprendan a aproximar a $Q_\pi$ y $\pi(a|s)$ utilizando el método Actor Critico.

La red de política (actor): $\pi(a|s;\theta)$ es un aproximador de $\pi(a|s)$ ; donde $\theta$ son parámetros de la red.

La red de valor (crítico): $q(s,a;w)$ es un aproximador de $Q_\pi(s,a)$, donde $w$ son los parámetros de la red.

El actor puede realizar las acciones, pero no sabe cómo mejorarlas, lo que requiere que el crítico le dé un refuerzo para que el actor sepa qué tipo de acción tiene un refuerzo alto y qué tipo de acción tiene refuerzo bajo. De esta manera, el agente (actor + critico) pueda mejorarse a sí misma, para que refuerzo recibido sea cada vez más alto.

Tal que:

$$V_\pi(s) = \sum_a \pi(a|s) \cdot Q_\pi(s,a) \cong \sum_a \pi(a|s;\theta) \cdot q(s,a;w)$$

**Estructura de la red de actor**

- Entrada: Estado $s$
- Salida: Funciones de distribución de posibles acciones.
- $A$ es un conjunto de acciones, $A = \{A_1, A_2, A_3 \dots\}$
- $\sum_{a \in A} \pi(a|s;\theta) = 1$

**Estructura de la red del crítico**

- Entradas: Estado $s$ y acción $a$
- Salida: Aproximación de función de estado valor (Salida escalar)

La estructura típica de la red del crítico es usar una capa convolucional y una capa totalmente conectada para extraer las características de las entradas, y se obtiene dos vectores (uno de estado y otro de acción). Luego se unen los dos vectores para obtener un vector de grado más alto. Finalmente, genera un número real con una capa totalmente conectada, que representa un refuerzo dado por el crítico. Este refuerzo muestra si es bueno o malo realizar una acción $a$ bajo en el estado $s$. Esta red de valor puede compartir parámetros convolucionales con la red de política, o puede ser completamente independiente de la red de política.

**Método Actor-Crítico**

El Método Actor-Crítico consiste entrenar tanto la red de políticas como la red de valor a la vez mediante el uso de redes neuronales para aproximar la función de valor de estado.

$$V(s;\theta,w) = \sum_a \pi(a|s;\theta) \cdot q(s,a;w)$$

El entrenamiento de las redes consiste en la actualización de los parámetros $\theta$ y $w$:

• Actualización de la red de política $\pi(a|s;\theta)$ sirve para aumentar el valor de $V(s;\theta,w)$

- La señal de supervisión es proporcionada por la red de valor. El actor mejora continuamente sus acciones en función del refuerzo dado por el crítico.

• Actualizar la red de valor $q(s,a;w)$ es hacer que el refuerzo sea más preciso.

- La señal de supervisión proviene solamente de los refuerzos. En el principio del entrenamiento, el crítico da refuerzos totalmente aleatorios, pero aumentará el nivel de puntuación de acuerdo con el refuerzo dado por el entorno, de modo que esté cerca de la puntuación real.

Pasos la actualización:

1. Observación de estado $S_t$
2. Tomar $S_t$ como entrada, y toma una muestra de acción en base a la red de políticas $\pi(a|s;\theta)$
3. Implementar la acción y observar el nuevo estado $s_{t+1}$, así como el refuerzo $r_t$
4. Utilizar el refuerzo $r_t$ para actualizar $w$ de la red de valor a través del algoritmo de diferencia temporal para que el crítico sea más preciso
5. Utilizar el gradiente de política para actualizar $\theta$ de la red de política para mejorar las acciones del actor.

**Método de diferencia temporal (TD) para actualización de red de valor**

1. Dar puntuaciones a acción $a_t$, $a_{t+1}$: calcular $q(s_t,a_t;w_t)$ y $q(s_{t+1},a_{t+1};w_t)$
2. TD objetivo: $y_t = r_t + \gamma \cdot q(s_{t+1},a_{t+1};w_t)$, que esta valoración es relativamente más cierta que $q(s_t,a_t;w_t)$
3. La función de coste $L$ es la diferencia entre el valor totalmente predicho $q$ y el valor

que contiene alguno valor verdadero $y_t$.

$$L(w) = \frac{1}{2}(q(s_t, a_t; w_t) - y_t)^2$$

4. Calcular el descenso de gradiente:

$$w_{t+1} = w_t - \alpha \cdot \frac{\partial L(w)}{\partial (w)}$$

**Gradiente de política para actualizar la red de política**

$$V(s; \theta, w) = \sum_a \pi(a|s; \theta) \cdot q(s, a; w)$$

La función de estado valor $V$ es equivalente a una puntuación media de todas las acciones realizadas por actor.

**Gradiente de política**: Función $V(s; \theta, w)$ con respecto a la derivada del parámetro $\theta$

- $g(a, \theta) = \frac{\partial \log(\pi(a|s; \theta))}{\partial \theta} \cdot q(s, a; w)$ donde $q$ es equivalente a la puntuación del crítico.
- $\frac{\partial V(s;\theta,w_t)}{\partial \theta} = E_A[g(A, \theta)]$ El gradiente de política es equivalente a la esperanza de la función $g$, la cual se puede obtener con la aproximación de Monte Carlo.

**Algoritmo de Monte Carlo:**

Muestreo de la red de políticas para obtener la acción: $a \sim \pi(\cdot\,|s_t; \theta_t)$, de esta forma $g(a, \theta)$ sigue una estimación imparcial (La media del estimador es igual al valor verdadero).

Con un gradiente estocástico $g$, se puede hacer un aumento del gradiente: $\theta_{t+1} = \theta_t + \beta \cdot g(a, \theta_t)$ para mejorar el comportamiento de actor.

## 4.3 Métodos de aprendizaje por refuerzo del MATLAB

Como el presente proyecto se realiza mediante simulaciones con MATLAB&SIMULINK, al crear el agente RL podemos utilizar los algoritmos que vienen con el paquete de aprendizaje por refuerzo de MATLAB. La versión 2022a de MATLAB contiene los algoritmos que se detallan en la *Tabla 2* [11].

| Agente | Tipo | Espacio de acción |
|---|---|---|
| Q-Learning (Q) | Basado en valor | Discreto |
| Deep Q-Network (DQN) | Basado en valor | Discreto |
| SARSA | Basado en valor | Discreto |
| Policy Gradient (PG) | Basado en política | Discreto o continuo |
| Actor-Critic (AC) | Actor-Crítico | Discreto o continuo |
| Proximal Policy Optimization (PPO) | Actor-Crítico | Discreto o continuo |
| Trust Region Policy Optimization (TRPO) | Actor-Crítico | Discreto o continuo |

| | | |
|---|---|---|
| Deep Deterministic Policy Gradient (DDPG) | Actor-Crítico | Continuo |
| Twin-Delayed Deep Deterministic Policy Gradient (TD3) | Actor-Crítico | Continuo |
| Soft Actor-Critic (SAC) | Actor-Crítico | Continuo |

*Tabla 2: Tipos de algoritmos de aprendizaje por refuerzo*

A la hora de elegir un agente, una buena práctica es empezar con un algoritmo más sencillo (y más rápido de entrenar) que sea compatible con los espacios de acción y observación. Para ello, la estrategia es probar algoritmos progresivamente más complicados si los más sencillos no funcionan como se desea.

En el presente trabajo, tanto el espacio de acción como observación son continuos. Para este caso, podemos aplicar método DDPG, TD3, PPO, SAC o TRPO en el nuestro problema. De los cuales, DDPG es el agente compatible más sencillo, seguido de TD3, PPO y SAC, a los que sigue TRPO. En general:

- TD3 es una versión mejorada y más compleja de DDPG.
- PPO tiene actualizaciones más estables, pero requiere más entrenamiento.
- SAC es una versión mejorada y más compleja de DDPG que genera políticas estocásticas.
- TRPO es una versión más compleja de PPO que es más robusta para entornos deterministas con menos observaciones.

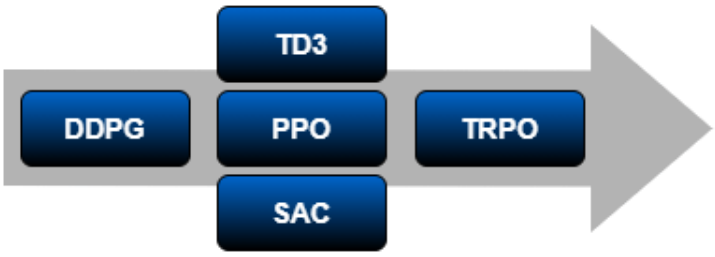



*Figura 12: El orden en que se toma los algoritmos (Fuente: [11])*

Por ello, en este trabajo hemos escogido el algoritmo DDPG para crear el agente.

### 4.3.1 DDPG

El método de aprendizaje de refuerzo profundo que aplicamos para el presente trabajo es *Deep Deterministic Policy Gradient* (DDPG), que se trata de un algoritmo sin modelo (model-free), en línea (online) y sin política (off-policy) basado en actor-crítico que busca una política óptima que maximiza el beneficio acumulado esperado a largo plazo[12].

Una de las razones por las que el aprendizaje por refuerzo es tan potente es que un agente no necesita tener ningún conocimiento sobre el entorno, pero sí puede aprender a través de interactuar con él. Esto recibe el nombre de aprendizaje por refuerzo sin modelo.

En términos más rigurosos, la palabra sin modelo significa que algoritmo no se esfuerza por aprender la dinámica subyacente que rige la interacción del agente con el entorno. En el caso de que el entorno tenga un espacio de estados discreto y el agente también tenga un número discreto de acciones entre las que elegir, el modelo de la dinámica del entorno es la matriz de transición de un paso: $p(s_{t+1}|s_t, a_t) = \Pr(S_{t+1}|\, S_t, A_t)$ . Esta matriz estocástica da todas las probabilidades de llegar a un estado deseado a partir del estado

y la acción actuales. Sin embargo, para los problemas con espacios de estado y acción de dimensiones continuas y/o elevadas, como el caso del presente trabajo, que los estados y acciones son continuas (dimensión infinita), el coste para calcular esta matriz es extremadamente alto [13].

En lugar de calcular la matriz de transición, los algoritmos sin modelo estiman directamente la política óptima o la función de valor a través de algoritmos como la iteración de políticas o la iteración de valores. Esto es mucho más eficiente desde el punto de vista computacional. No obstante, los métodos sin modelo suelen requerir un mayor número de episodios para el entrenamiento, ya que necesita explorar el espacio de estado entero para asegurar como obtener un mayor refuerzo.

Por otra parte, los algoritmos que se caracterizan por no calcular la política generalmente emplean una política de conducta (*behavior policy*) separada que es independiente de la política que se está mejorando, donde la política de conducta se utiliza para simular las trayectorias. Una ventaja clave de esta separación es que la política de conducta puede operar muestreando todas las acciones, mientras que la política de estimación puede ser determinista.

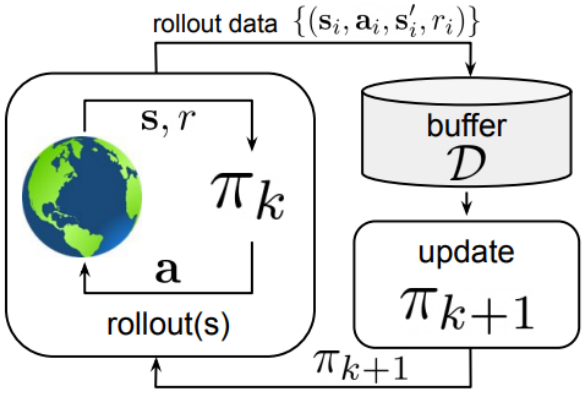



*Figura 13: Esquema de aprendizaje sin política (Fuente: [14])*

En los algoritmos sin política, las experiencias del agente se añaden a un buffer de experiencias, y cada nueva política $\pi_k$ recoge datos adicionales, de manera que el buffer de experiencias se compone de las transiciones $\{(s_0, a_0, s_1, r_0); (s_1, a_1, s_2, r_1) \ldots\}$,y todos estos datos se utilizan para entrenar una nueva política actualizada $\pi_{k+1}$ (*Figura 13*). El agente interactúa con el entorno para obtener las transiciones [14].

El aprendizaje sin política permite utilizar muestras más antiguas (recogidas con las políticas anteriores) en el cálculo. Para actualizar la política, las experiencias se muestrean a partir de un buffer que comprende las experiencias/interacciones que se recogen de sus propias políticas predecesoras. Esto mejora la eficiencia de las muestras, ya que no es necesario recogerlas cada vez que se cambia una política.

#### *4.3.1.1 Funciones de actor y crítico* [15]

Para estimar la política y la función de valor, el agente DDPG mantiene cuatro aproximadores de funciones:

- Actor $\pi(S;\theta)$: El actor, con parámetros $\theta$, toma la observación $S$ y devuelve la acción correspondiente que maximiza el beneficio a largo plazo.
- Actor objetivo $\pi_t(S;\theta_t)$: Para mejorar la estabilidad de la optimización, el agente actualiza periódicamente los parámetros del actor objetivo $\theta_t$ utilizando los últimos valores de los parámetros del actor.

- Crítico $Q(S, A; \phi)$: El crítico, con parámetros $\phi$, toma la observación $S$ y la acción $A$ como entradas y devuelve la correspondiente expectativa del beneficio a largo plazo.
- Crítico objetivo $Q_t(S, A; \phi_t)$: Para mejorar la estabilidad de la optimización, el agente actualiza periódicamente los parámetros del crítico objetivo $\phi_t$ utilizando los últimos valores de los parámetros del crítico.

Tanto $Q(S, A; \phi)$ como $Q_t(S, A; \phi_t)$ tienen la misma estructura y parametrización, y tanto $\pi(S; \theta)$ como $\pi_t(S; \theta_t)$ tienen la misma estructura y parametrización.

Durante el entrenamiento, el agente afina los valores de los parámetros en $\theta$. Después del entrenamiento, los parámetros permanecen en su valor sintonizado y el aproximador de función de actor entrenado se almacena en $\pi(S)$.

#### *4.3.1.2 Algoritmo de entrenamiento de DDPG* [15]

- Inicializar el crítico $Q(S, A; \omega)$ con valores de parámetros aleatorios ω, e inicializa los parámetros del crítico objetivo $\omega_t$ con los mismos valores: $\omega_t = \omega$.
- Inicializar el actor $\pi(S; \theta)$ con valores de parámetros aleatorios $\theta$, e inicializar los parámetros del actor objetivo $\theta_t$ con los mismos valores: $\theta_t = \theta$.
- Para cada paso del tiempo de entrenamiento, se realizan las siguientes operaciones:
  1. Para el estado actual $S$, selecciona una acción $A = \pi(s; \theta) + N$, donde $N$ es ruido estocástico del modelo de ruido.
  2. Ejecutar la acción $A$. Observa el refuerzo $R$ y el siguiente estado $S'$.
  3. Almacenar la experiencia $(S, A, R, S')$ en el búfer de experiencia.
  4. Muestrear un mini-lote aleatorio de $M$ experiencias $(S_i, A_i, R_i S_i')$ del búfer de experiencia.
  5. Si $S_i'$ es un estado terminal, establecer la función de valor destino $y_i$ en $R_i$. De lo contrario, configurarlo en

     $$y_i = R_i + \gamma \cdot Q_t(S_i', \pi_t(S_i'; \theta_t);\ w_t)$$

     El objetivo de la función de valor es la suma de refuerzo de experiencia $R_i$ y el refuerzo futuro descontado.

     Para calcular el refuerzo acumulado, el agente primero calcula una siguiente acción pasando el siguiente estado $S_i'$ de la experiencia muestreada al actor objetivo. El agente encuentra el refuerzo acumulado pasando la siguiente acción al crítico objetivo.

  6. Actualizar los parámetros del crítico minimizando el coste $L$ en todas las experiencias muestreadas.

     $$L = \frac{1}{M} \sum_{i=1}^{M} \left(y_i - Q(S_i, A_i; w)\right)^2$$

  7. Actualizar los parámetros del actor utilizando el siguiente gradiente de política para maximizar el refuerzo con descuento esperado.

     $$\nabla_\theta J \approx \frac{1}{M} \sum_{i=1}^{M} G_{ai} G_{\pi i}$$

     $$G_{ai} = \nabla_A Q(S_i, A; w) \quad \text{donde } A = \pi(S_i; \theta)$$

     $$G_{\pi i} = \nabla_\theta \pi(S_i; \theta)$$

Donde $G_{ai}$ es el gradiente de la salida del crítico con respecto a la acción calculada por la red de actores, y $G_{\pi i}$ es el gradiente de la salida del actor con respecto a los parámetros del actor. Ambos gradientes se evalúan para el estado $S_i$.

8. Actualizar los parámetros del actor de destino y del crítico en función del método de actualización de destino.

# 5 Agente DDPG

En esta sección, describiremos cómo se crea el agente DDPG. Usamos el código MATLAB para crear el agente, y gracias a la Toolbox de aprendizaje por refuerzo de MATLAB, el código es particularmente sencillo. Por ello, en esta sección, aspiramos a explicar el significado de cada código como los parámetros que aparecen.

## 5.1 Creación del agente

Para crear un agente DDPG con actor y críticos en MATLAB, por defecto basados en las especificaciones de observación y acción del entorno. Entonces, se siguie los siguientes pasos.

1. Crear especificaciones de observación para el entorno.
2. Crear especificaciones de acción para el entorno.
3. Si es necesario, especificar el número de neuronas en cada capa de aprendizaje o si se va a utilizar una capa LSTM. Para ello, crear un objeto de opción de inicialización del agente utilizando *rlAgentInitializationOptions*.
4. Si es necesario, especificar las opciones del agente utilizando la función *rlDDPGAgentOptions*.
5. Crear el agente utilizando la función *rlDDPGAgent*.

**Especificaciones del espacio de observaciones**

En este trabajo, hemos creamos decenas de versiones de agente. En las versiones con mejor rendimiento, utilizamos al menos ocho observaciones siguientes.

En primer lugar, tenemos que observar el estado del invernadero, que corresponde a cuatro variables medibles del invernadero, que son la masa seca de lechuga, concentración de dióxido de carbono, temperatura y humedad relativa interior del invernadero. Estos cuatro elementos son observaciones básicas esenciales, sin las cuales no podríamos controlar el clima del invernadero.

Después, añadimos la observación de las tres acciones de control del momento anterior. La razón de observar las acciones del momento anterior es que el estado del invernadero tarda un cierto tiempo en cambiar después de realizar las acciones. Por lo que cuando diseñamos la función de refuerzo nos fijamos más las acciones del momento anterior. Del mismo modo, tenemos que introducir estas acciones como las entradas de la red neuronal.

Además, hay que avisar al agente cuando hay irradiancia solar. Esto se debe a que los requisitos climáticos del invernadero son diferentes cuando hay irradiancia y cuando no la hay. Por lo tanto, usar esto como entrada le dará al agente una mejor comprensión del entorno.

A parte de estas ocho observaciones, los otros datos no son imprescindibles, como la temperatura exterior o la concentración de $CO_2$ externa. Esto se debe a que no afectan directamente al crecimiento de las lechugas, sino al clima interior, que ya conocemos por otras observaciones.

Por otro lado, debido a la diferencia de escalas de las entradas, las observaciones con entradas más grandes pueden no ser aprendidas eficazmente durante el proceso de modelización, mientras que los datos normalizados pueden ser escalados uniformemente a un intervalo, evitando así el problema del sesgo de aprendizaje para cada escala. Y los datos normalizados pueden mejorar la eficiencia del entrenamiento, acelerar la convergencia y mejorar la estabilidad del modelo.

Por lo tanto, normalizamos todas las entradas para que su valor mínimo se establezca en cero y su valor máximo en uno.

```
% Create observation specification
numObs=8;
obsInfo = rlNumericSpec([numObs
1],'LowerLimit',[0;0;0;0;0;0;0;0],'UpperLimit',[1;1;1;1;1;1;1;1]);
obsInfo.Name='observations';
obsInfo.Description = '4 mesurements + 3 actions + 1 timeinf';
```

### Especificaciones del espacio de acciones

El sistema de red neuronal tiene tres salidas, que corresponde a la inyección de $CO_2$ al invernadero, ventilación y calefacción. Para estas acciones tomamos el mismo límite máximo y límite mínimo que usa el MPC con el propósito de evitar que el sistema tome acciones que no tienen sentido, así como limitar espacios de exploración.

También hacemos la normalización a las acciones de salida.

```
% Create action specification
numAct=ops.nu;
actInfo = rlNumericSpec([numAct 1],'LowerLimit',[0;0;0],'UpperLimit',[1;1;1]);
actInfo.Name='controllable input';
```

Llegados a este punto, tenemos definido tanto las entradas como las salidas del sistema de red neuronal (el conjunto de la red de actor y de crítico). Creamos el interfaz de interacciones del entorno para el agente.

```
% Create the enviroment interface
env = rlSimulinkEnv(mdl,mdl+"/RL Agent",obsInfo,actInfo);
```

### Creación de la red del crítico

La red del crítico tiene dos entradas separadas, por una parte, entran las observaciones (*Figura 11*). Las observaciones conectan con una capa completamente conectada de $L_1$ neuronas, y este número de neuronas $L_1$ lo usamos para todas las capas de la red del crítico. Por otra parte, la red del crítico también recoge las acciones tomadas por la red del actor del mismo instante del tiempo como entradas, ya que el crítico tiene que poder observar como de buenas son estas acciones tomadas para poder dar una evaluación.

Después de cada capa completamente conectada, utilizamos la función de activación ReLU para agregar no linealidad al modelo. La función de activación ReLU es lineal para valores positivos y cero para valores negativos. La forma de ReLU se muestra en *Figura 14*, y su expresión matemática es:

$$ReLu(x) = \max(0, x)$$

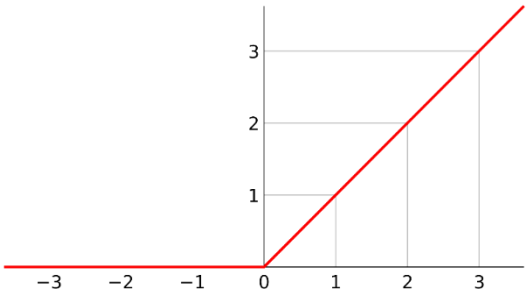



*Figura 14: Ilustración de función ReLu (Fuente: Wikipedia)*

Dado que no existe una forma rigurosa de definir el número de capas, neuronas y tipos de las funciones de activación. Después de estudiar los ejemplos de casos parecidos y las que se sugieren en los ejemplos de MATLAB para agente DDPG, las redes que hemos definido (crítico y actor) tienen la misma estructura (mismas capas y funciones de activación) que [29].

Por otro lado, hemos experimentado con cambios en el número de neuronas en este trabajo. Después de una serie de pruebas encontramos que el agente entrena y converge de forma correcta con entre 10 y 50 neuronas por capa.

La entrada de acción de red del crítico converge con la entrada de observación después de pasar a través de una capa totalmente conectada y, después de pasar una serie de capas totalmente conectadas, produce una salida unidimensional. Lo que representa una evaluación de las acciones tomadas bajo los estados observados (*Figura 15*).

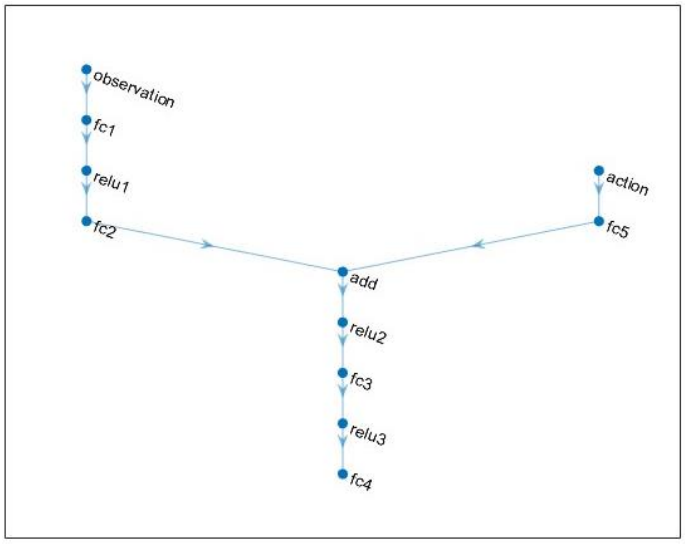



*Figura 15: Estructura de la red del crítico*

Y todo esto se define en el siguiente código:

```
% Create critic Network

statePath = [
featureInputLayer(numObs,'Normalization','none','Name','observation')
    fullyConnectedLayer(L1,'Name','fc1')
    reluLayer('Name','relu1')
    fullyConnectedLayer(L1,'Name','fc2')
    additionLayer(2,'Name','add')
```

```
    reluLayer('Name','relu2')
    fullyConnectedLayer(L1,'Name','fc3')
    reluLayer('Name','relu3')
    fullyConnectedLayer(1,'Name','fc4')];

actionPath = [
    featureInputLayer(numAct,'Normalization','none','Name','action')
    fullyConnectedLayer(L1, 'Name', 'fc5')];

criticNetwork = layerGraph(statePath);
criticNetwork = addLayers(criticNetwork, actionPath);

criticNetwork = connectLayers(criticNetwork,'fc5','add/in2');
criticNetwork = dlnetwork(criticNetwork);
```

### Creación de la red del actor

La red del actor solo tiene una capa de entrada, donde entran las observaciones del sistema. Después de pasar 4 capas completamente conectadas, se conecta con una capa de salida de 3 nodos, que representa 3 acciones de control (*Figura 16*).

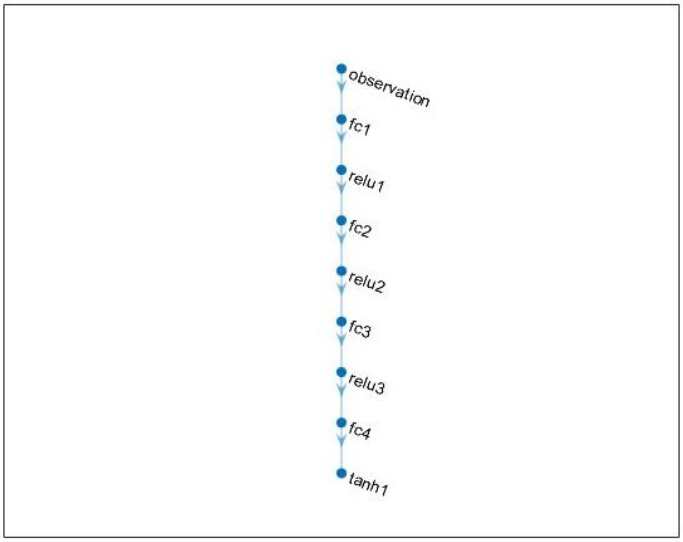



*Figura 16: Estructura de la red del actor*

A diferencia de las funciones de activación ReLu que utilizamos en las capas ocultas. En la salida de la red del actor aplicamos la función tangente hiperbólica (*Figura 17*) como la función de activación, y su expresión matemática es:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

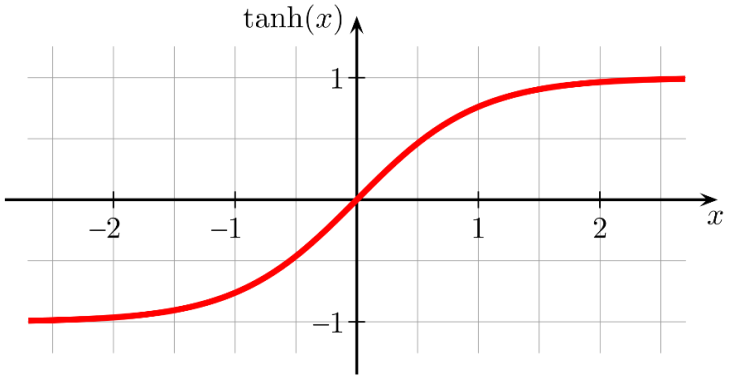



*Figura 17: Función de activación tangente hiperbólica (Fuente: Wikipedia)*

Y el código para crear la red del actor es:

```
actorNetwork = [
    featureInputLayer(numObs,'Normalization','none','Name','observation')
    fullyConnectedLayer(L2,'Name','fc1')
    reluLayer('Name','relu1')
    fullyConnectedLayer(L2,'Name','fc2')
    reluLayer('Name','relu2')
    fullyConnectedLayer(L2,'Name','fc3')
    reluLayer('Name','relu3')
    fullyConnectedLayer(numAct,'Name','fc4')
    tanhLayer('Name','tanh1')
    ];
actorNetwork = dlnetwork(actorNetwork);
```

## Especificación para aprendizaje de las redes

### Tasa de aprendizaje

Este es una variable clave para el aprendizaje de la red, si la tasa de aprendizaje es demasiado baja, el entrenamiento puede llevar mucho tiempo para conseguir un aprendizaje aceptable. Si la tasa de aprendizaje es demasiado alta, entonces el entrenamiento puede alcanzar un resultado subóptimo o no converge. Mediante una serie de pruebas que hicimos para este modelo, creemos que $[10^{-5}, 10^{-3}]$ es un rango adecuado para que el agente aprende bien y no llevar mucho tiempo en entrenamiento, pero en función de los cambios realizados en el modelo y la función de refuerzo, también haremos pequeñas modificaciones a este valor según ritmo de aprendizaje o para afinar el modelo.

### Umbral de gradiente (Gradient Threshold)

El umbral de gradiente se especifica en un escalar positivo o infinito. Si la norma $L_2$ del gradiente de un parámetro aprendible es mayor que el umbral de gradiente entonces se recorta el gradiente para que la norma $L_2$ sea igual al umbral de gradiente.

El recorte de gradiente ayuda a evitar la explosión del gradiente estabilizando el entrenamiento a tasas de aprendizaje más altas y en presencia de valores atípicos. El recorte de gradiente permite entrenar las redes más rápidamente y no suele afectar a la precisión de la tarea aprendida. [16]

Para el presente trabajo, utilizamos valores de 0.1 a 10 para el umbral de gradiente.

### Regularización

Uno de los problemas que se dan en el aprendizaje profundo es la aparición de sobreajuste (overfitting). El sobreajuste es el efecto de ajustar demasiado la función a los datos de entrenamiento, lo que puede extraer relaciones no causales y destruir la generalización de las redes neuronales. La *Figura 18* muestra un ejemplo de sobreajuste:

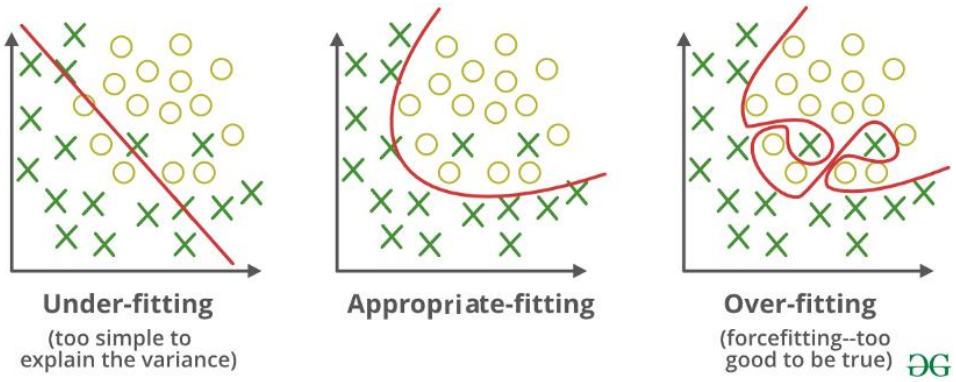



*Figura 18: Ejemplo del subajuste (izquierdo), apropiado (medio) y sobreajuste (derecho) (Fuente: [17])*

El factor de regularización se utiliza para añadir un término de regularización para los pesos a la función de coste $L(w)$ es una forma de reducir el sobreajuste. El término de regularización también se denomina decaimiento del peso. La función de coste con el término de regularización tiene la forma siguiente [18]:

$$L_R(w) = L(w) + \lambda \cdot \Omega(\mathrm{w})$$

Donde $w$ es el vector de peso, $\lambda$ es el factor de regularización que se especifica como un escalar no negativo y la función de regularización es $\Omega(\mathrm{w}) = \frac{1}{2}w^T w$

En el siguiente código se especifican los parámetros para aprendizaje de las redes:

```
% Specify options for the critic optimizer.
criticOptions = rlOptimizerOptions('LearnRate',1e-
3,'GradientThreshold',1,'L2RegularizationFactor',1e-4);

%Create the critic representation using the specified neural network and options.
critic = rlQValueFunction(criticNetwork,obsInfo,actInfo,...
    'ObservationInputNames','observation','ActionInputNames','action');

% Specify options for the actor optimizer.
actorOptions = rlOptimizerOptions('LearnRate',1e-
3,'GradientThreshold',1,'L2RegularizationFactor',1e-4);

actor = rlContinuousDeterministicActor(actorNetwork,obsInfo,actInfo);
```

### Especificación para agente DDPG

La configuración del agente DDPG incluye una serie de ajustes como el modelo de ruido, el tamaño del buffer, etc. Para ello, presentaremos las configuraciones que hemos cambiado para este agente, y el resto de las configuraciones se ajustarán a la configuración por defecto del modelo.

Para crear el agente DDPG, especificamos las opciones del agente DDPG mediante la función *rlDDPGAgentOptions* de MATLAB. En primer lugar, el periodo para que el agente tome una acción es 1, lo que significa por cada paso de simulación (1 paso de simulación equivalente el paso de 15 minutos de tiempo del invernadero) el agente toma las nuevas acciones.

**Tamaño de búfer**

Búfer es una zona que almacena las experiencias previas, durante el entrenamiento, el agente calcula las actualizaciones utilizando un mini-lote de experiencias muestreadas aleatoriamente de este búfer. Para que el algoritmo tenga un comportamiento estable, el tamaño de búfer debe ser lo suficientemente grande como para contener una amplia gama de experiencias, pero no siempre es bueno mantener todas experiencias. Si sólo utiliza los datos más recientes, tendrá sobreajuste y, si utiliza demasiada experiencia, puede ralentizar el aprendizaje [19].

El tamaño de búfer de experimento que cogemos para el presente trabajo es de $10^5$ a $10^6$, y el tamaño de mini-lote de 64.

**Factor de descuento**

El objetivo del aprendizaje por refuerzo es conseguir el mayor refuerzo acumulado. Para ello, tenemos que usar un factor de descuento para reducir los refuerzos futuros predichos. Dado que el agente difícilmente puede predecir con mucha exactitud los eventos que pasarán en el futuro, eso provoca que las predicciones de los refuerzos futuros serán menos fiables, de modo que para entonces el refuerzo podría de dejar de existir.


*Figura 19: Incertidumbres sobre refuerzos futuros (Fuente: [10])*

Precisamente porque tenemos el efecto del clima externo con incertidumbre en nuestro modelo de invernadero, en este caso, es más beneficioso prestar más atención en los refuerzos a tiempo cercano. Esto se consigue fijar un valor de factor de descuento relativamente bajo. Para este trabajo, escogemos valores de 0.8 a 0.95 para el factor de descuento.

El Código para especificar las opciones y crear el agente DDPG se muestra en siguiente:

```
% Specify the DDPG agent options
agentOptions = rlDDPGAgentOptions(...
    'SampleTime',1,...
    'ActorOptimizerOptions',actorOptions,...
    'CriticOptimizerOptions',criticOptions,...
    'ExperienceBufferLength',1e6,...
    'DiscountFactor', 0.8);
```

```
% Create the DDPG agent using the specified actor representation, critic
representation, and agent options.
agent = rlDDPGAgent(actor,critic,agentOptions);
```

### Modelo de ruido a las acciones

El método DDPG entrena una política determinista. Dado que la política es determinista, si el agente explorara dentro de la política, al principio probablemente no probaría una variedad de acciones lo suficientemente amplia como para encontrar señales de aprendizaje útiles. Para que las políticas DDPG exploren mejor, añadimos ruido a sus acciones en el momento del entrenamiento. [19]

El modelo de ruido que utiliza por agente DDPG para la exploración es *Ornstein-Uhlenbeck* [20]. Que consiste lo siguiente:

En cada paso de tiempo de muestreo $k$, el valor del ruido $v(k)$ se actualiza utilizando la siguiente fórmula, donde $T_s$ es el tiempo de muestreo del agente, y el valor inicial $v(1)$ está definido por el parámetro *InitialAction*.

$$v(k+1) = v(k) + MeanAttractionConstant.* \big(Mean - v(k)\big).* T_S + StandardDeviation(k).* radn\big(size(Mean)\big).* \sqrt{T_s}$$

Para cada periodo de muestreo, la desviación estándar decae de forma siguiente:

$$decayedStandardDeviation = StandardDeviation(k).* (1 - StandardDeviationDecayRate)$$

$$StandardDeviation(k+1) = \max\left(decayedStandardDeviation, StandardDeviationMin\right)$$

Para las variables que muestran en las ecuaciones anterior tomamos los valores de la *Tabla 3.*

| Variable | Descripción | Valor |
|---|---|---|
| *InitialAction* | Valor inicial de acción para modelo de ruido | 0 |
| Mean | Media del modelo de ruido | 0 |
| MeanAttractionConstant | Constante que especifica la rapidez con la que la salida del modelo de ruido es atraída por la media | 0.15 |
| StandardDesviationDecayRate | Tasa de decaimiento de la desviación estándar | 1e-4 |
| StandardDesviation | Desviación estándar del modelo de ruido | 0.1 |
| StandardDesviationMin | Desviación estándar mínima | 0 |

*Tabla 3: Parámetros del modelo de ruido*

### Especificación de las opciones de entrenamiento

En la especificación de las opciones de entrenamiento, especificamos el número máximo de episodios para entrenar al agente a 1000, como por defecto. Este valor será modificado según los ajustes del modelo de simulación. Y el entrenamiento solo termina después de alcanzar este número máximo de episodios.

```
% Train Agent
maxepisodes = 1000;
maxsteps = nDays*96;
trainingOpts = rlTrainingOptions(
    'MaxEpisodes',maxepisodes,...
    'MaxStepsPerEpisode',maxsteps,...
    'Verbose',false,...
    'Plots','training-progress',...
    'StopTrainingCriteria','EpisodeCount',...
    'StopTrainingValue', maxepisodes);
```

### Modelo de ruido para generar curvas meteorológicas para entrenamiento

Para garantizar que el agente pueda ser controlado de forma estable bajo diferentes condiciones meteorológicas externas y para evitar la aparición de sobreajuste en el aprendizaje del agente, damos al agente unos datos meteorológico externo y climas interiores iniciales diferentes para cada simulación. Para ello, añadimos un ruido a los datos meteorológicos nominales. Este modelo de ruido realiza una modificación de la meteorología exterior antes de iniciar un nuevo episodio.

La aplicación concreta es la siguiente. Haremos una serie de muestras aleatorias sobre una distribución uniforme con una media de 1 y una desviación de 0,35. Y multiplicar los valores muestreados uno a uno con los datos meteorológicos nominales para generar nuevos datos, y se obtiene nuevos datos para hacer el entrenamiento (*Figura 20)*

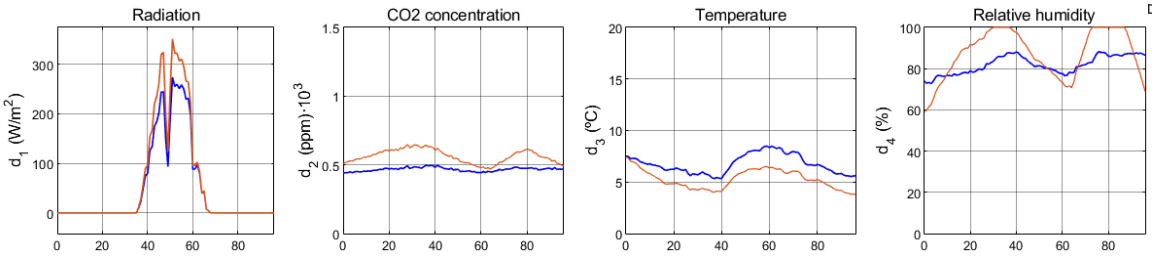



*Figura 20: Comparación de las curvas meteorológicos nominales (azul) con las curvas generadas para entrenamiento (naranja)*

## 5.2 Diseño de la función de refuerzo y valoración del agente

Una vez configurado el entorno y las especificaciones del agente, debemos considerar qué queremos que el agente logre y qué refuerzo recibirá cuando lo logra.

Esto requiere diseñar una función de refuerzo que permita al algoritmo de aprendizaje entender cuándo la política se vuelve mejor y, en última instancia, converger al resultado que queremos buscar.

En el aprendizaje por refuerzo, no hay restricciones para crear funciones de refuerzo. Podemos utilizar refuerzos dispersos, o reforzar después de cada paso de tiempo, o sólo

reforzar cuando un segmento está completamente terminado después de un período de tiempo muy largo. El refuerzo puede calcularse con funciones no lineales o incluir miles de parámetros. Depende totalmente del enfoque que se adopte para formar al agente de forma eficaz.

Dado que no hay restricciones sobre cómo se puede crear la función de refuerzo, podemos encontrar situaciones en las que el refuerzo es escaso. Lo que significa el objetivo que queremos lograr no se consigue hasta después de una larga secuencia de acciones correctas.

Este nuestro caso del modelo de invernadero, como la lechuga no crece en las horas que no haya irradiancia solar, si en función de refuerzo sólo tiene en cuenta a la masa de la lechuga, será muy difícil que el agente aprenda qué hacer durante el tiempo en que la lechuga no crezca. Eso provoca que, durante un largo período de tiempo, el agente prueba diferentes acciones y pase por muchos estados diferentes sin obtener ningún refuerzo, y por lo tanto no aprenda nada en este proceso. Ya que la probabilidad de que el agente sea capaz de generar aleatoriamente la secuencia exacta de acciones para obtener un mayor refuerzo futuro es muy pequeña.

De hecho, hicimos una serie de pruebas sobre como diseñar la función de refuerzo. En las primeras versiones, sólo añadimos la masa de la lechuga en la función de refuerzo,

$$reward(t) = y_1(t)$$

Después de un entrenamiento de 500 episodios (*Figura 22*) obtenemos los resultados como muestra en la *Figura 21*. Comparamos las medidas obtenidas con las medidas del caso de sin aplicar ninguna acción de control *Figura 3*. Se observa que, con el agente entrenado, sí que puede conseguir un aumento de producción de lechuga, la masa seca de lechuga aumenta de 4,3 g/m$^2$ a 4,6 g/m$^2$. En la comparación de los resultados obtenidos, observamos que agente solo aumenta la concentración de dióxido de carbono interior, mientras que el control de la temperatura interior y la humedad relativa es insignificante.

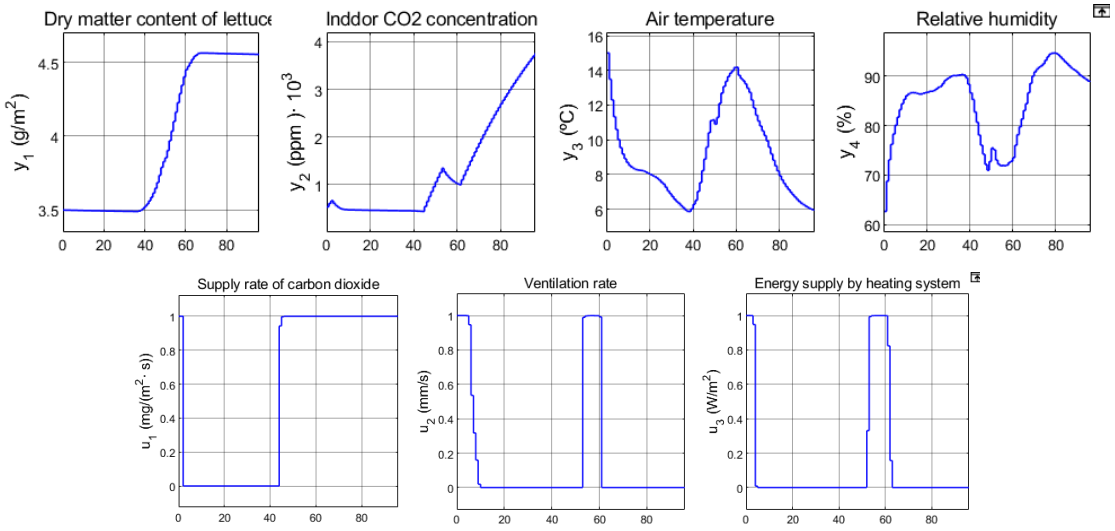



*Figura 21: Resultados de simulación; la primera fila representa las medidas del invernadero y la segunda fila representa las acciones tomadas*

Por otro lado, en esta versión no hemos normalizado las acciones del agente y, el límite inferior y superior para las acciones que definimos coinciden con las restricciones del MPC. Esto significa que el valor de $u_1$ puede ser de 0 a 1.2 mg·m$^{-2}$·s$^{-1}$; $u_2$ de 0 a 7,5 mm·s$^{-1}$ y $u_3$ de 0 a 150 W·m$^{-2}$. Sin embargo, por razones desconocidas, como se puede ver en la *Figura 21*, todas estas salidas del agente se mantienen entre cero y uno. Para corregir esto, en

las versiones posteriores haremos normalización a las acciones del agente para asegurar que esté entre cero y uno. Y luego las desnormaliza para obtener el valor correcto antes de aplicar estas acciones al entorno.

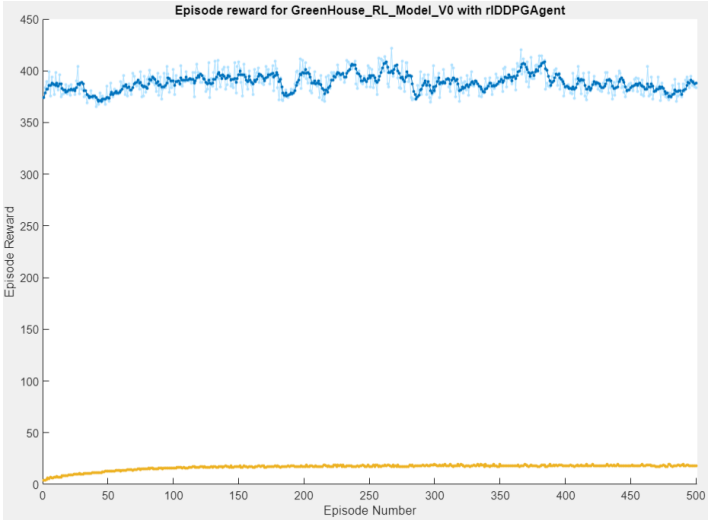



*Figura 22: Curva de aprendizaje*

En la *Figura* 22, la línea azul claro es el refuerzo acumulado obtenido del episodio actual, el azul oscuro es la media de los últimos refuerzos, y la línea naranja es el episodio Q0, que representa el valor calculado por el crítico al principio de cada episodio. Episodio Q0 es un indicador que nos dice lo bien que se ha entrenado el crítico. Si tenemos el crítico perfecto que puede predecir con precisión el refuerzo esperado a largo plazo basándose en las observaciones del principio del episodio, este valor debería coincidir con el refuerzo real obtenido durante ese mismo episodio. En general, no es necesario que esto ocurra para los modelos actor-crítico. Debido que el actor puede converger antes que el crítico y en ese momento estaría bien dejar de entrenar el agente.

A juzgar por la línea azul oscuro, parece que ese agente no está aprendiendo. Ya que los refuerzos que recibe el agente no mejoran a partir del primer episodio. Esto también demuestra que, después de 500 episodios de entrenamiento, las acciones tomadas por el agente no son mejores que las acciones aleatorias del primer episodio.

Después de esto, consideramos que, debido a que la producción de lechuga casi siempre está aumentando, la función de refuerzo utilizada anteriormente puede no ser una buena evaluación para el nivel de crecimiento de lechuga. Por lo que añadimos un bloque de retardo de medidas en el modelo, que se utiliza para guardar las medidas del paso anterior. Estas medidas guardadas se agregan a las observaciones de agente y la función de refuerzo.

La nueva función de refuerzo es la masa lechuga seca actual menos la masa de lechuga seca del estado anterior, lo que puede darnos una mejor idea de qué tan rápido está creciendo las lechugas Para ello, la nueva función de refuerzo es:

$$r(t) = y_1(t) - y_1(t-1)$$

Sin embargo, los resultados seguían siendo insatisfactorios. Después de esto, se ajustaron los parámetros de la red neuronal, como el número de neuronas por capa, la tasa de aprendizaje, etc. Se hicieron una serie de ajustes, pero los resultados seguían siendo malos. En este punto, el agente entrenado era completamente incapaz de controlar el clima interior.

En este punto, nos dimos cuenta de que teníamos que rediseñar la función de refuerzo, y era necesario dispersar los refuerzos, es decir, proporcionar refuerzos intermedios más pequeños para guiar al agente por el camino correcto.

En primer lugar, para comprobar si nuestra red neuronal puede controlar el clima interior, establecemos el objetivo de solo mantener la temperatura del aire interior del invernadero a un valor de referencia $T_{ref}$. Para poder controlar mejor la temperatura, establecemos un refuerzo que no solo da recompensas al nivel de temperatura interior del invernadero, sino también da recompensas a la acción de calentamiento ($a_3$). Porque una gran parte del cambio en la temperatura interior es causada por la temperatura exterior. Debido a que agente no tiene datos sobre el clima exterior (en las primeras versiones, no hemos dado ninguna observación sobre el clima exterior al agente), y si no se agrega refuerzo para las acciones, el valor de salida del refuerzo estará dependiendo mucho del cambio de temperatura exterior, aunque las acciones tomadas son correctas. Para evitar esto, establecemos una gran proporción de la recompensa por las acciones correctas.

$$T_{reward}(t) = -20\left|T_{ref} - y_3(t)\right|$$

Siendo $T_{ref} = 20\,°C$ y $y_3(t)$ es la temperatura real dentro del invernadero en instante $t$.

$$a_{3_{reward}}(t) = \begin{cases} -10 & si:\ y_3(t) < 15℃\ \ y\ \ a_3(t-1) < 50\dfrac{W}{m^2} \\ -10 & si:\ y_3(t) > 25℃\ \ y\ \ a_3(t-1) > 50\dfrac{W}{m^2} \\ 5 & otros\ casos \end{cases}$$

El resultado obtenido por esta función de refuerzo se muestra en la *Figura 23*. En esta versión, observamos que el agente puede controlar significativamente la temperatura interior y mantenerla entre 19 y 21 ºC, de modo que, el agente reduce el suministro de calefacción y aumenta la ventilación durante las horas en que la irradiancia está disponible y, toma acciones contrarias en ausencia de irradiancia solar.

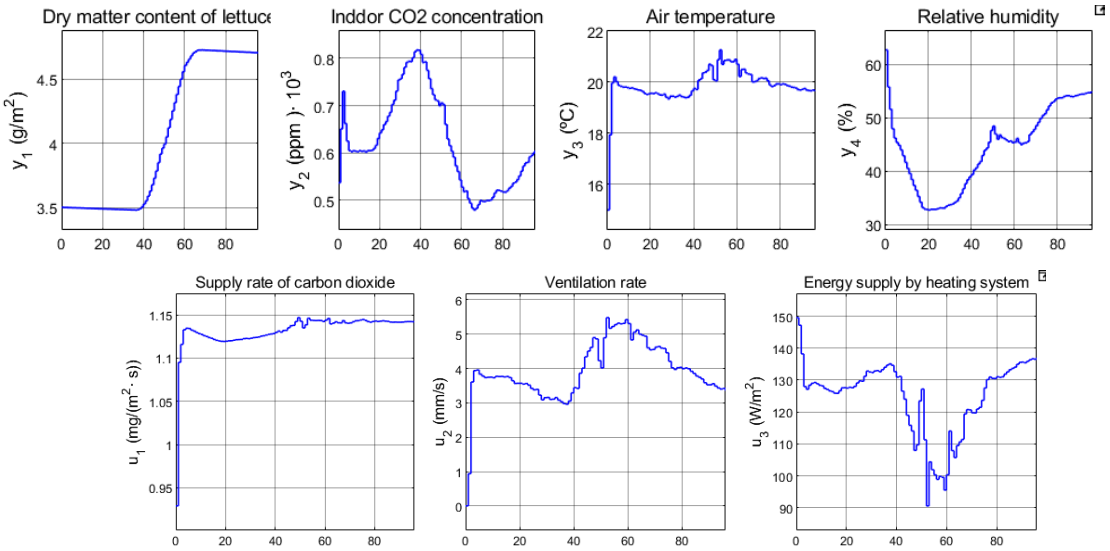



*Figura 23: Resultados de simulación*

Por otro lado, como se puede ver en la curva de aprendizaje (*Figura 24*), este agente es muy bueno en entrenar esta tarea. Porque con solo 20 episodio de entrenamiento, ya

consigue un refuerzo acumulado bueno. Entonces, también creemos que los parámetros que hemos ajustado para la red neuronal son correctos.

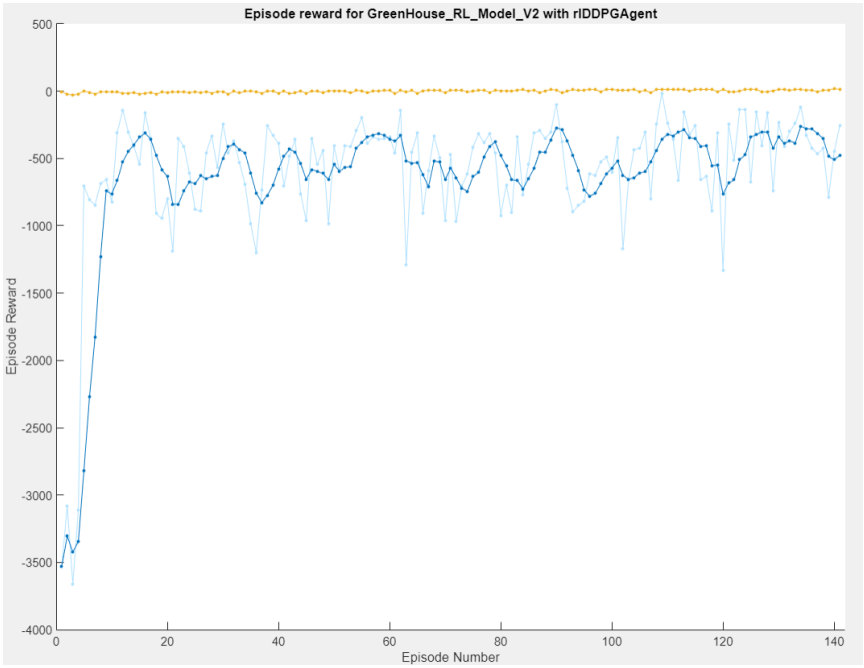



*Figura 24: Curva de aprendizaje*

Después de otra serie de simulaciones, decidimos hacer algunos cambios en el modelo RL por las siguientes razones.

El crecimiento de la lechuga no solo depende por los efectos de las acciones que se toma por controlador. La incertidumbre del clima exterior también tiene un papel muy importante, ya que participa directamente en el sistema de climatización interno. Además de esto, la irradiancia solar también afecta directamente el crecimiento de las lechugas, lo que conlleva a la falta de esta parte de información para que el agente tome mejores acciones.

Con este fin, agregamos una información sobre la irradiancia solar al modelo. Como se representa en la *Figura 25,* Esta información se emite un valor de 1 desde las 8:00h hasta las 18:00h y 0 para el resto de las horas, ya que la irradiancia solar sólo está disponible entre las 8:00h a las 18:00h. El objetivo de establecer esta variable es que el agente sepa cuándo estará disponible la irradiancia para poder aplicar diferentes estrategias de control cuando esté la irradiancia solar y cuando no lo esté.

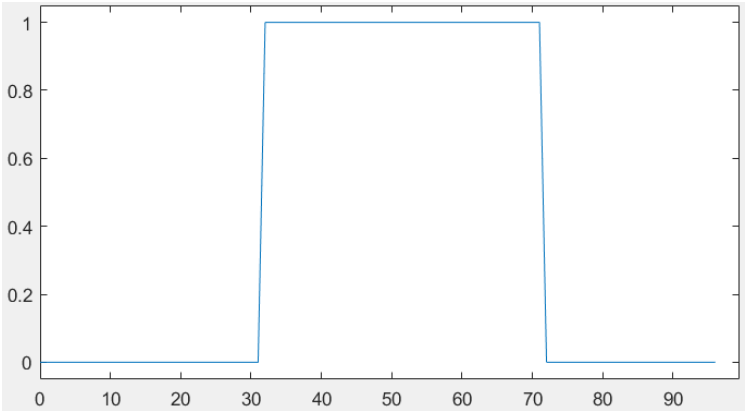



*Figura 25: Información horaria de la radiación solar proporcionada al agente*

Luego, hemos probado de premiar al agente por aumentar la producción de lechuga y también damos refuerzos por controlar la temperatura y humedad relativa a un valor de referencia.

No damos penalización a las acciones realizadas (uso de recursos) por el agente porque en este caso nos interesa observar qué acciones realizará el agente para obtener un mayor refuerzo acumulado y si fueran razonables.

El valor de referencia de la humedad lo fijamos en el 65%. En cambio, el valor de referencia de la temperatura son variables, $T_{ref}$ =18 ºC en 8:00h a 18:00h y, $T_{ref}$ =10 ºC en el resto de las horas.

Dadas las condiciones anteriores, comprobamos que el agente es capaz de mantener la temperatura y la humedad en los valores de referencia dados (*Figura 26*). Al mismo tiempo, se tiende a emitir un nivel muy alto de $CO_2$ para obtener una mayor producción. Y, en este caso, la concentración de $CO_2$ del invernadero supera el valor máximo de $CO_2$ definido en el modelo MPC (1.6 ppm·$10^3$).

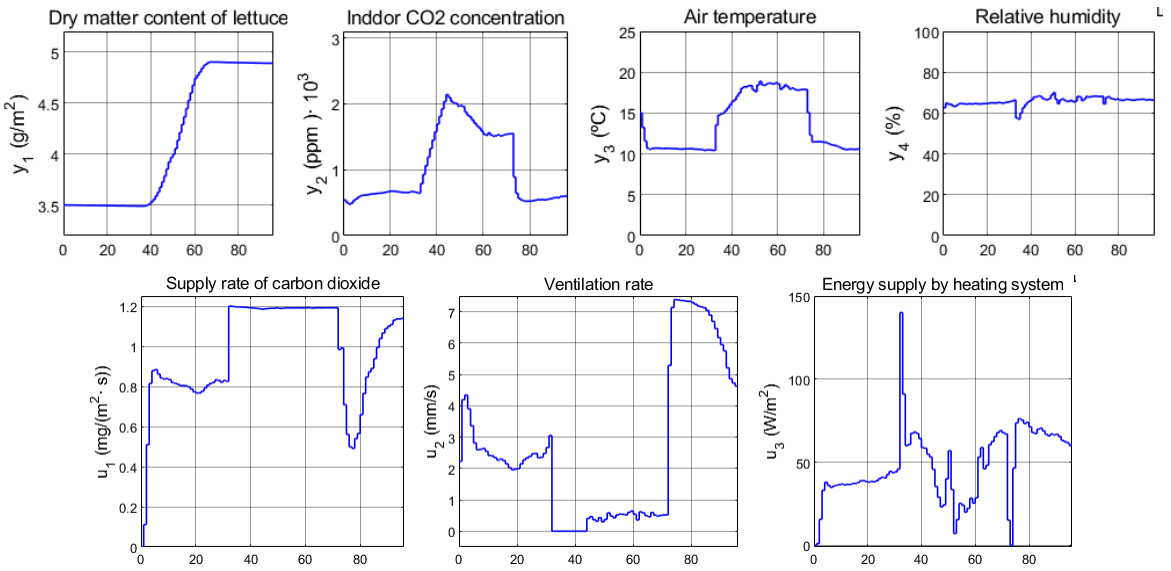



*Figura 26: Resultados de simulación*

A partir del caso anterior, hemos añadido el control de la concentración de $CO_2$. Además, hemos modificado el control de las salidas del invernadero fijando un límite máximo y un mínimo para la concentración de $CO_2$, la temperatura y la humedad relativa interior del invernadero (*Figura 27*, *Figura 28* y *Figura 29*). Esto se implementa de tal manera que cuando estas medidas son mayores o menores que los límites dados, se aplica una mayor penalización, y a la inversa, mantener los resultados dentro de los límites recibe un refuerzo más pequeño.

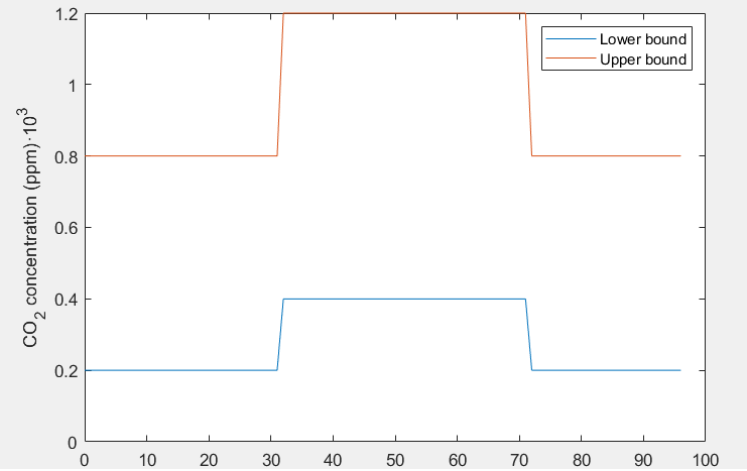



*Figura 27: Rango de CO2 establecido*

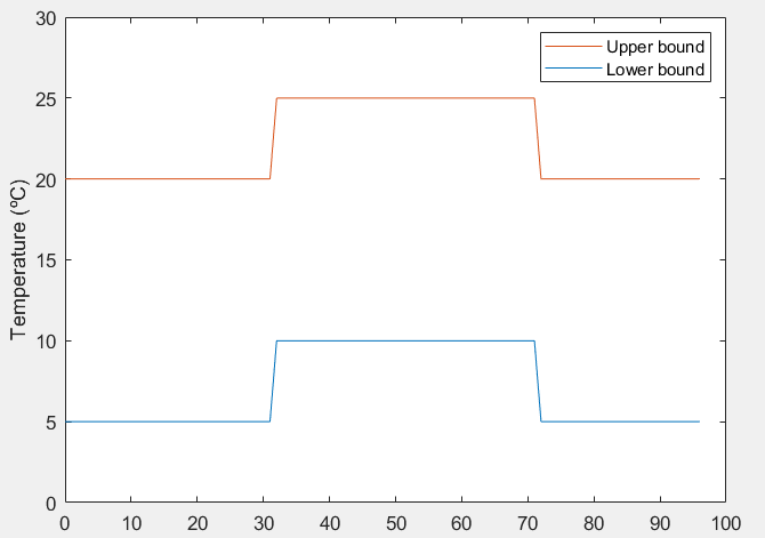



*Figura 28: Rango de temperatura establecido*

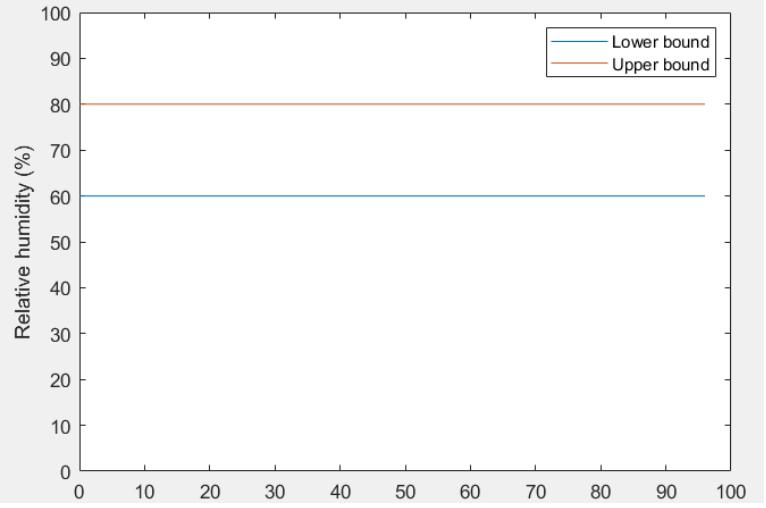



*Figura 29: Rango de humedad relativa establecido*

Después de eso, hicimos una serie de ajustes sobre el rango de los límites superior e inferior, el valor de refuerzo por mantener dentro del rango y por salir de rango y la forma de darlo (si dar un valor de refuerzo constante por salir del rango o un valor dependiendo de la desviación, etc.). Además, añadimos penalizaciones por el uso de recursos en la función de refuerzo. En *Figura 30* muestra un ejemplo de resultados obtenidos con esta nueva función de refuerzo.

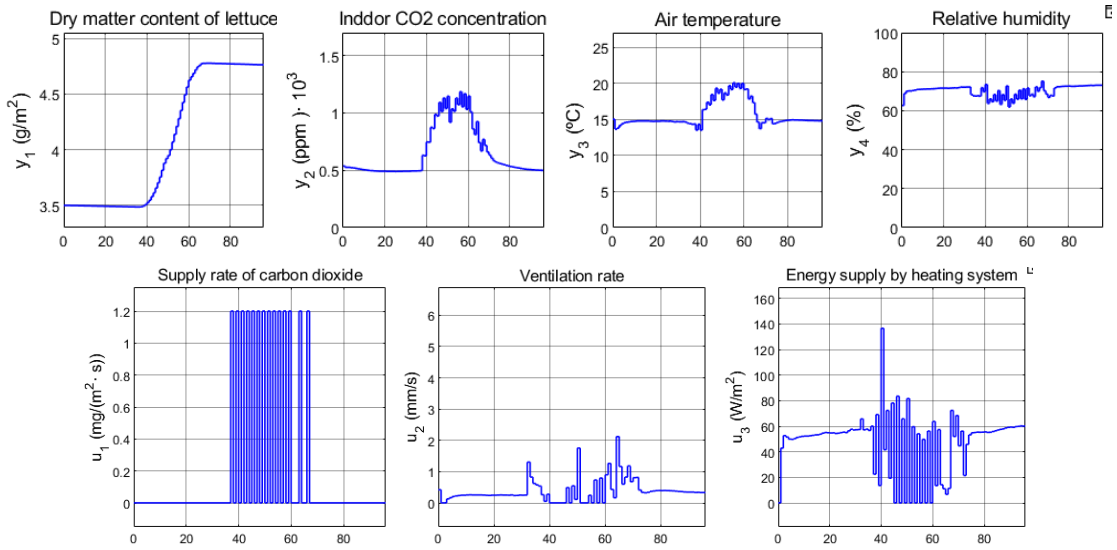



*Figura 30: Resultados de simulación*

A partir de los resultados (*Figura 30*), el agente es capaz de controlar el clima interior a un buen nivel. En ausencia de radiación, no se inyecta el dióxido de carbono al invernadero y la temperatura interior se controla a 15 ºC. Y en presencia de la radiación solar, la concentración de $CO_2$ en el interior aumenta a 1000 ppm y la temperatura a 20 ºC. Por otra parte, la humedad relativa se mantiene alrededor de 70% durante todo el tiempo. En comparación con las versiones anteriores mencionadas, ésta consume muchos menos recursos y también consigue un buen aumento de la producción de la lechuga.

Sin embargo, esta versión tiene un defecto en las acciones de salida del agente, que se puede ver que las acciones varían significativamente entre cada paso de simulación, lo que creemos que tiene un impacto en el refuerzo. Como utilizamos las acciones del instante anterior como observación para calcular el refuerzo, si el cambio de acción entre cada instante y el siguiente es muy grande, entonces dará lugar a una gran diferencia de refuerzo entre cada instante y el siguiente. Esto no es bueno para predecir los refuerzos futuros.

Por otro lado, un gran cambio de acción también haría que el actuador se encendiera y apagara con frecuencia, lo que podría acelerar el desgaste del actuador. Por lo tanto, tenemos que utilizar algunos medios para limitar este cambio de acción.

En primer lugar, utilizamos la función de refuerzo para limitar este cambio de acción. Esto se implementa de tal manera que cuando la variación de acción es más allá del diez por ciento de ese valor máximo de salida ($u_j(t) - u_j(t-1) > 0.1\, u_{j,max}$), se aplica una penalización relativamente grande.

Sin embargo, después de varios intentos, se obtiene resultados insatisfactorios. El agente no sólo tiene dificultades para aprender a reducir la cantidad de variación de la acción, sino que también afecta al aprendizaje del agente del control climático del invernadero.

Por esta razón, la solución que hemos adaptado es utilizar una función de saturación a la salida de las acciones tomadas por agente para limitar esta variación entre acciones. Y los resultados de simulación se muestra en *Figura 31*.

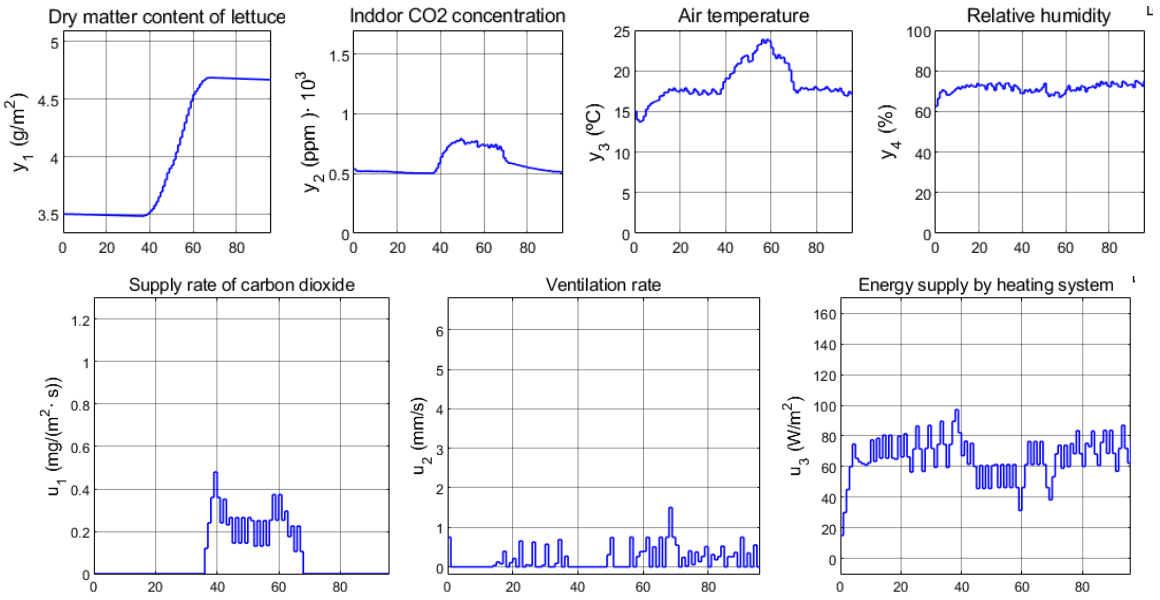



*Figura 31: Resultados de simulación*

En este punto, el diseño de la función de recompensa se declara completo. En resumen, el diseño de la función de refuerzo es un gran reto para nosotros. Dado que los refuerzos escasos dificultan el aprendizaje del agente y, si remodelamos la función de refuerzo y damos abundosos refuerzos al agente, es muy probable que hay un atajo escondido dentro de estos refuerzos, y el agente lo tomará.

Por ejemplo, si queremos que el agente mantenga la temperatura a un nivel determinado y el refuerzo que recibe el agente por mantener la temperatura es relativamente alto, el agente utilizará mucho la calefacción cuando la temperatura esté por debajo de aquella temperatura dada y pondrá el sistema de ventilación a un nivel alto cuando la temperatura esté por encima. Esto claramente no es razonable para nosotros, pero para el agente esta puede ser la solución que le proporciona el mayor refuerzo acumulado.

Por lo tanto, al diseñar la función de refuerzo, debemos tener especial cuidado con los refuerzos que recibe el agente en cada estado y acción. Esto suele requerir mucha simulación de prueba y error para verificarlo.

## 5.3 Validación del agente RL

Llegados a este punto, ya hemos configurado el entorno, creado el agente y después de que el algoritmo de aprendizaje haya convergido en la solución, el método sigue pudiendo tener defectos.

Matemáticamente, la política consiste en una red neuronal, que puede contener miles de pesos, umbrales y funciones de activación no lineales. Estos valores se combinan con la estructura de la red para formar una función compleja que asigna las observaciones de alto nivel a las acciones de bajo nivel.[10]

Para nosotros, esta función es como una "caja negra". Es posible que tengamos una comprensión intuitiva de cómo funciona esta función y de las características ocultas que ha identificado la red, pero no entendemos el razonamiento matemático que hay detrás de un determinado valor de peso o de umbral. En consecuencia, si la política no cumple las especificaciones o el entorno operativo cambia, es posible que no sepamos cómo adaptar la política para resolver el problema.[10]

Además, si en algún momento el sistema no funciona como se espera, o si la política obtenida no es del todo correcta y no somos capaces de modificar la parte problemática de la política, hay que rediseñar el modelo de agente o de entorno y volver a entrenarlo. Una nueva ronda de diseño, entrenamiento y valoración puede llevar mucho tiempo. Por lo tanto, debemos probar la política final con mucho cuidado.

Sin embargo, verificar que una política cumple la especificación utilizando una red neuronal no suele ser fácil. Por la razón de que, es difícil utilizar una política aprendida para predecir el comportamiento del sistema en un estado basándose en su comportamiento en otro estado. Simplemente porque un pequeño cambio en alguna entrada de la red neuronal puede activar un conjunto de neuronas muy diferente y producir resultados inesperados, y no hay forma de saberlo a menos que se realicen pruebas.

Tras numerosas simulaciones en diferentes condiciones meteorológicas externas dentro del rango siguiente (es el mismo rango de clima externa que usamos durante el proceso de entrenamiento),

$$0.65 \cdot d_{j,nominal}(t) \leq d_{j,test}(t) \leq 1.35 \cdot d_{j,nominal}(t) \text{ para } j = 1,2,3,4$$

comprobamos que el agente tiene un comportamiento muy estable.

Además, para probar la capacidad de mantener el control estable en condiciones meteorológicas externas no encontradas durante el proceso de entrenamiento, fijamos el ruido del clima en 0.6, es decir:

$$0.4 \cdot d_{j,nominal}(t) \leq d_{j,test}(t) \leq 1.6 \cdot d_{j,nominal}(t) \text{ para } j = 1,2,3,4$$

A continuación, presentamos dos ejemplos con un nivel de ruido de 0,6:

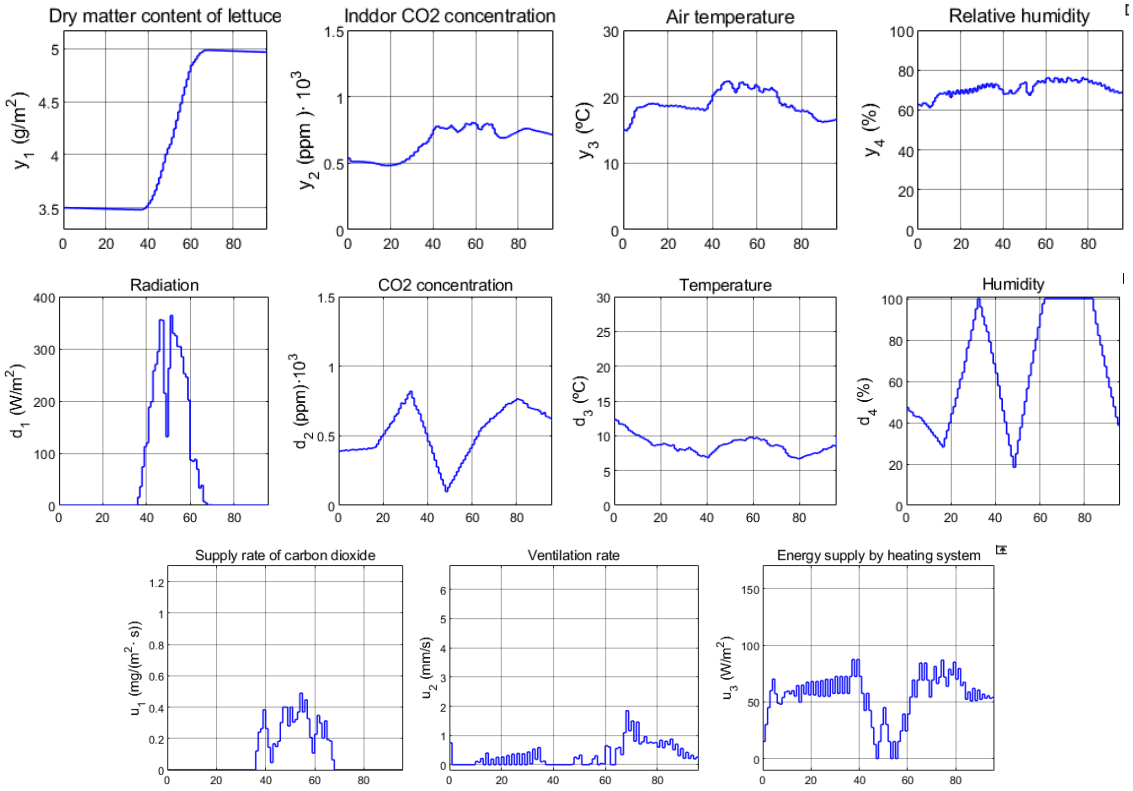



*Figura 32: Resultados de simulación (caso 1); la primera fila representa las medidas de invernadero, la segunda fila representa las curvas meteorológicas y la tercera fila representa acciones tomadas por agente*

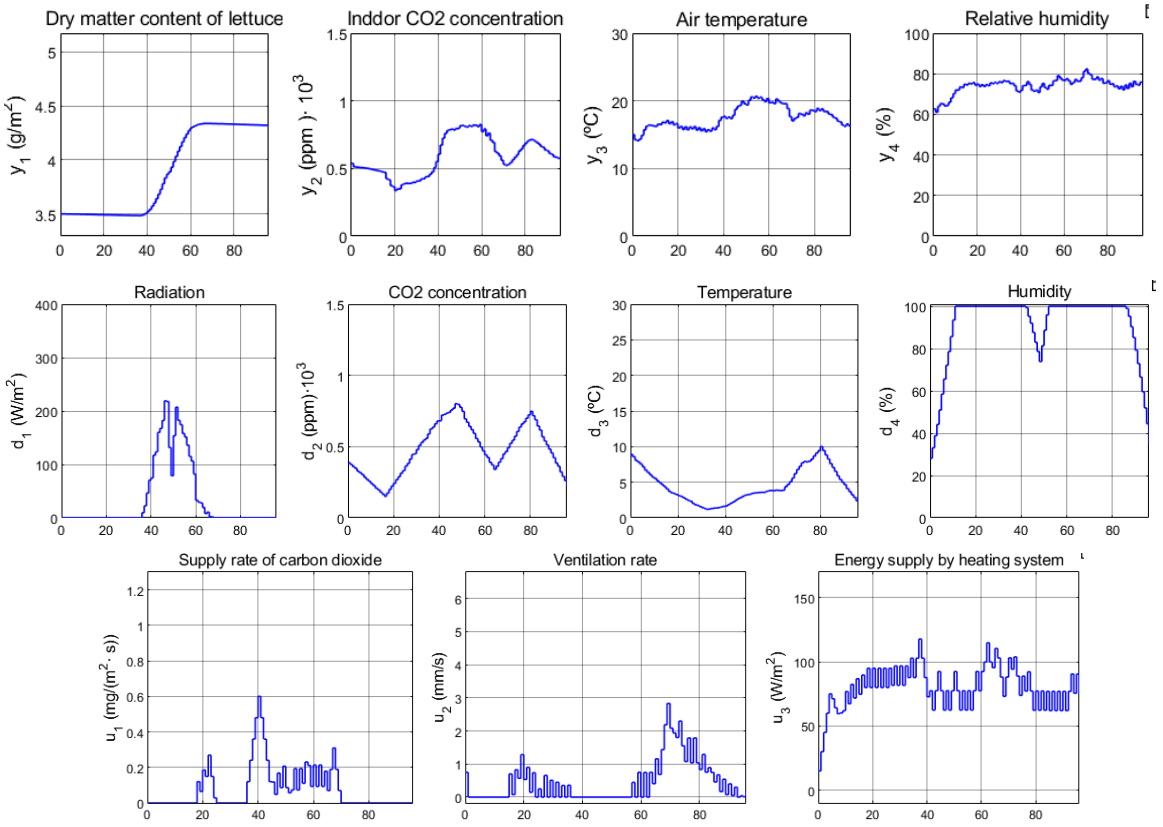



*Figura 33: Resultados de simulación (caso 2)*

Como se puede ver en la *Figura 32* y *Figura 33*, en ambos casos representa una gran diferencia de su curva meteorológica con respecto a la curva meteorológica nominal. En ambos casos la variación de concentración $CO_2$ exterior supera las 500 ppm entre los valores más altos y bajos durante el día. Además, la humedad relativa en el exterior varía considerablemente, desde un mínimo del 20% hasta un máximo del 100%.

Sorprendentemente, el agente es capaz de mantener el clima interior a un nivel adecuado incluso con clima exterior tan "malo". El agente es capaz de mantener la humedad interior en torno al 70% en todo momento, mantener la temperatura entre 15 y 25 ºC y una concentración superior a 500 ppm.

Aunque no podemos demostrar que éste sea el estado óptimo de control, sí podemos garantizar que el uso del agente mantendrá el clima interior en un nivel confortable.

# 6 Comparación MPC con agente DDPG y resultados

Para comparar el rendimiento del MPC con el agente DDPG, introducimos una función de beneficio económico[21]:

$$J = \Phi\left(y_1(t_f)\right) - \int_{t_b}^{t_f} \left(c_q u_q(t) + c_{CO_2} u_{CO_2}(t)\right) dt$$

donde $\Phi\left(y_1(t_f)\right)$ es el ingreso bruto obtenido en el momento de la cosecha $t_f$ al vender el producto cosechado en la subasta, y $c_q u_q(t) + c_{CO_2} u_{CO_2}(t)$ son los costes de funcionamiento del equipo de climatización en $Hfl \cdot m^{-2} \cdot s^{-1}$. El precio de subasta de la lechuga sigue una relación lineal $\Phi\left(y_1(t_f)\right) = c_{pri,1} + c_{pri,2} y_1(t_f)$, parametrizado por $c_{pri,1}$ en $Hfl \cdot m^{-2}$ y $c_{pri,2}$ en $Hfl \cdot kg^{-1} \cdot m^{-2}$, entre el precio de subasta y el peso de cosecha de la lechuga $x_d$ en $kg \cdot m^{-2}$.

Se supone que los costes de funcionamiento del equipo de acondicionamiento climático están relacionados linealmente con la cantidad de energía $u_q$ en $W \cdot m^{-2}$ y la cantidad de dióxido de carbono introducida en el sistema $u_c$ en $kg \cdot m^{-2} \cdot s^{-1}$. Estos costes de funcionamiento se parametrizan mediante el precio de la energía $c_q$ en $Hfl \cdot J^{-1}$ y el precio de dióxido de carbono $c_{CO_2}$ en $Hfl \cdot kg^{-1}$, sus valores se encuentran en la *Tabla 4*.

| Parámetro | Valor |
|---|---|
| $c_{CO_2}$ | $42 \cdot 10^{-2}\ Hfl/kg$ |
| $c_q$ | $6.35 \cdot 10^{-9}\ Hfl/J$ |
| $c_{pri,1}$ | $1.8\ Hfl/m^2$ |
| $c_{pri,2}$ | $16\ Hfl/kg$ |

*Tabla 4: Parámetros de la función de beneficio económico*

Para el MPC, basta introducir directamente esta función al modelo, por lo que esta función de beneficio económico $J$ es lo mismo que la función de coste $V$ que utiliza MPC, de modo que:

$$V = -J$$

Por otra parte, en lo que respecta a las restricciones de MPC, utilizamos los mismos límites que en el **apartado 3.2.1**, salvo que los límites de temperatura mínima y máxima se han reducido en 5 ºC. Y los ajustes del controlador se muestra en *Tabla 5.*

| Parámetro | Descripción | Valor |
|---|---|---|
| $h$ | Periodo de muestreo | $15\ minutes$ |
| $N_p$ | Periodo de predicción | $6\ hours$ |
| $N_s$ | Número de trayectorias | 1 |
| $\gamma$ | Tamaño de incertidumbre | $0;\ 0.15;\ 0.3$ |

*Tabla 5: Parámetros para ajuste del controlador MPC*

En caso del agente DDPG, la función de refuerzo es más complejo.

En primero, hemos modificado la función de beneficio económico para adaptarla al modelo RL. Hemos eliminado las partes constantes ($c_{pri,1}$ y $c_{pri,2} \cdot y_1(t_b)$), ya que las constantes que no tienen ningún efecto en la búsqueda del control óptimo (búsqueda del máximo absoluto). Además, para los recursos utilizados, $u_q$ y $u_{co_2}$, hemos utilizado los valores del momento anterior, porque el agente no puede conocer las acciones de control del momento actual para calcular el refuerzo actual. Y la nueva función de beneficio económico instantáneo tiene la forma siguiente:

$$J_{RL}(t) = c_{pri,2}\big(y_1(t) - y_1(t-1)\big) - \Big(c_q \cdot u_3(t-1) + c_{co_2} \cdot u_1(t-1)\Big)h$$

A parte de esto, también establecemos refuerzos por mantener la temperatura dentro del límites superiores $T_{max}(t)$ e inferiores $T_{min}(t)$ establecido, para que el agente aprenda a controlar de forma más rápida.

$$T_{reward}(t) = \begin{cases} -c_1 \cdot \min\Big(\big(y_3(t) - T_{min}(t)\big)^2, \big(y_3(t) - T_{max}(t)\big)^2\Big) & si: y_3(t) < T_{min}(t) \ \ o \ y_3(t) > T_{max}(t) \\ c_2 \cdot \min\left(1, \dfrac{1}{\big(y_3(t) - T_{ref}(t)\big)^2}\right) & si: T_{min}(t) \le y_3(t) \le T_{max}(t) \ \ y \ \ y_3(t) \neq T_{ref}(t) \\ c_2 & si: \ \ y_3(t) = T_{ref}(t) \end{cases}$$

Y también le premia por mantener la concentración de $CO_2$ dentro del rango:

$$CO_{2_{reward}}(t) = \begin{cases} -c_3 \cdot \min\Big(\big(y_2(t) - CO_{2_{min}}(t)\big)^2, \big(y_2(t) - CO_{2_{max}}(t)\big)^2\Big) & si: y_2(t) < T_{min}(t) \ \ o \ y_2(t) > CO_{2_{max}}(t) \\ c_4 & otros \ casos \end{cases}$$

Donde los términos $c_1, c_2, c_3$ y $c_4$ son parámetros ajustados manualmente que muestra en *Tabla 6*, los valores de $T_{max}(t)$ y $T_{min}(t)$ se puede observar en la *Figura 34*, y los valores de $CO_{2_{max}}(t)$ y $CO_{2_{min}}(t)$ se encuentra en la *Figura 35*.

Por otra parte, la temperatura de referencia $T_{ref}$ está expresada en:

$$T_{ref}(t) = \frac{T_{max}(t) - T_{min}(t)}{2}$$

| Parámetro | Valor |
|---|---|
| $c_1$ | 0.001 |
| $c_2$ | 0.0005 |
| $c_3$ | 0.1 |
| $c_4$ | 0.0003 |

*Tabla 6: Parámetros de la función de refuerzo*

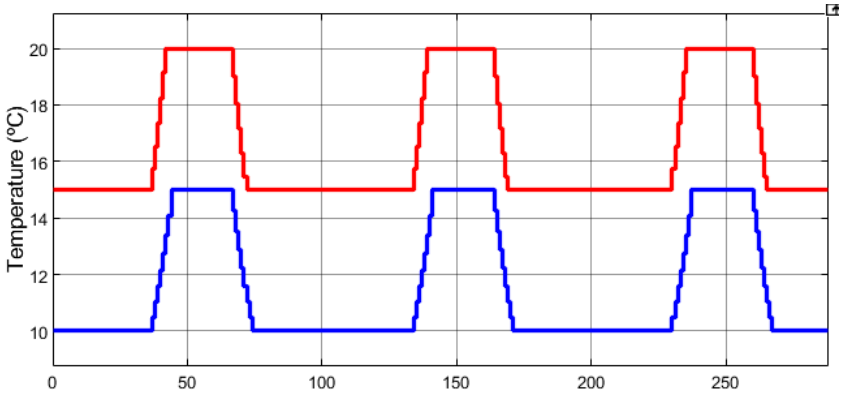



*Figura 34: Límite superior (rojo) e inferior (azul) de temperatura*

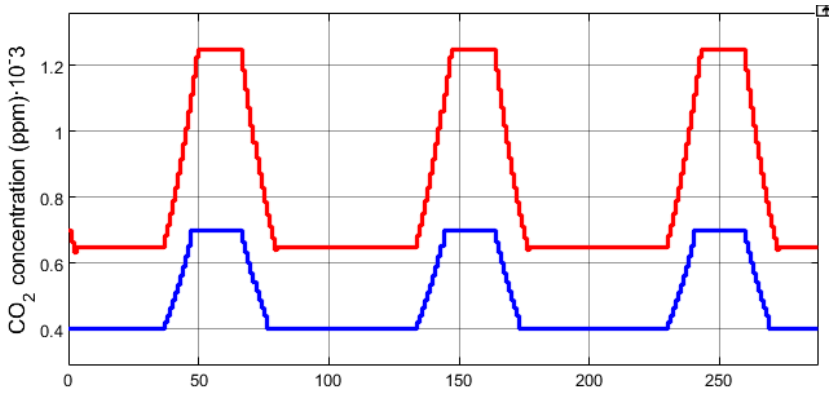
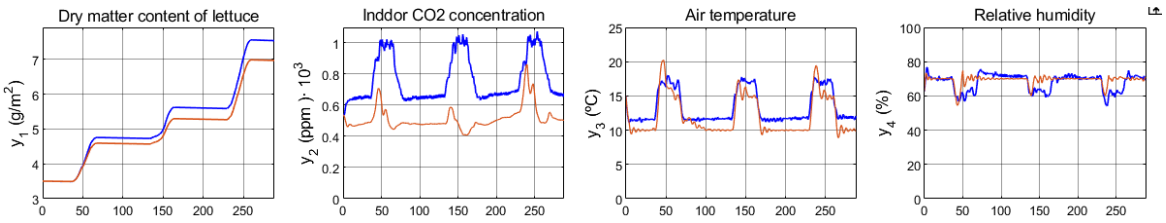



*Figura 35: Límite superior (rojo) e inferior (azul) de $CO_2$*

Por consecuencia, la función de refuerzo del agente DDPG está compuesta por:

$$reward(t) = J_{RL}(t) + T_{reward}(t) + CO_{2_{reward}}(t)$$

## 6.1 Resultado de simulación



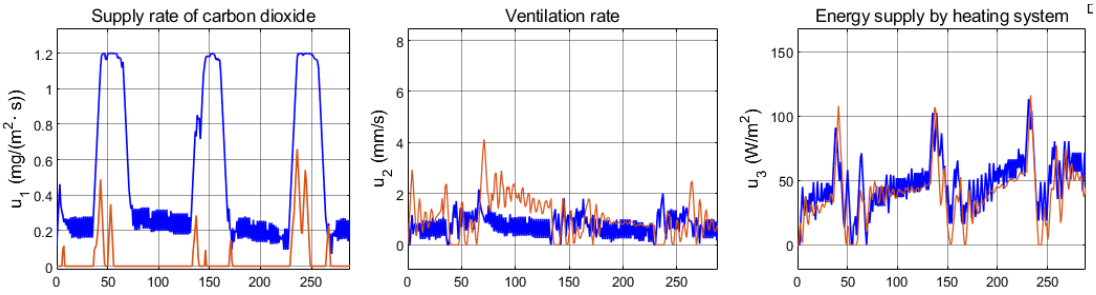



*Figura 36: Resultados de simulación; Las curvas naranjas son resultados de MPC y azules de agente DDPG*

Como se puede observar en la *Figura 36*, comparando los resultados obtenidos por el MPC con los obtenidos por el agente DDPG, encontramos que ambos mantienen los niveles de humedad relativa interior de forma muy similar y ambos mantienen la temperatura en el nivel mínimo. El agente es claramente más conservador que el MPC porque la función de refuerzo penaliza cuando la temperatura desciende por debajo de $T_{min}$, por lo que el agente mantiene la temperatura en un nivel ligeramente superior a la temperatura mínima para evitar a recibir esta penalización.

En términos de control de la concentración de $CO_2$, el agente controla claramente la concentración de $CO_2$ a un nivel mucho más alto que el MPC. En términos de producción, el agente logra una mayor producción de lechugas, pero en término de beneficio económico, el MPC logra un mayor retorno económico (1.843 Hfl·m$^{-2}$) que el agente (1.788 Hfl·m$^{-2}$). Esto se debe a que la rentabilidad económica del aumento de la producción de las lechugas es inferior al coste del consumo de recuso para la función de retorno económico con los parámetros establecidos en *Tabla 4*. Por ello, los niveles de temperatura se mantienen bajos en ambos casos. Y, en el caso del $CO_2$, dado que el aumento de la inyección de $CO_2$ puede aumentar significativamente la producción y, por tanto, la rentabilidad económica, es posible que en esta versión el agente solo haya explorado los espacios con el que aumenta el $CO_2$ y haya llegado a una solución subóptima.

Para resolver el problema, podemos hacer que el agente continúe con el entrenamiento para explorar un espacio de estado más amplio, o bien podemos remodelar la función de refuerzo y redefinir los límite superiores e inferiores de $CO_2$. Aunque con esto permite a agente encuentre la solución óptima en este problema, pero si el precio de subasta de lechuga o el coste de $CO_2$ cambia, este agente no se adaptará para encontrar nueva solución óptima, a no ser que tome todos estos parámetros variables como observación, rediseñar el agente y entrenar de nuevo.

# 7 Resume del presupuesto

En este capítulo se listarán en tablas las unidades y costos de cada elemento tanto equipo, software como mano de obra que se han aplicado en el desarrollo del proyecto.

| Recursos humanos | | | |
|---|---|---|---|
| | **Unidad** | **Coste Unitario** | **Coste total** |
| Búsqueda y lectura de bibliografía | 50 h | 15 €/h | 750 € |
| Desarrollo de simulación | 100 h | 15 €/h | 1500 € |
| Obtención y análisis de resultado | 100 h | 15 €/h | 1500 € |
| Elaboración de la memoria | 50 h | 15 €/h | 750 € |
| | | **Total:** | 4500 € |

| Equipo y Software | | | |
|---|---|---|---|
| | **Unidad** | **Coste Unitario** | **Coste total** |
| Ordenador portátil | 1 Ud. | 700 €/Ud. | 700 € |
| MATLAB and Simulink Student Suite | 1 Ud. | 69 €/Ud. | 69 € |
| Deep Learning Toolbox | 1 Ud. | 20 €/Ud. | 20 € |
| Reinforcement Learning Toolbox | 1 Ud. | 20 €/Ud. | 20 € |
| Horas de simulación | 200 h | 2 €/h | 400 € |
| | | **Total:** | 1209 € |

| | **Coste** |
|---|---|
| Recursos humanos | 4500 € |
| Equipo y Software | 1209 € |
| **Total:** | 5709 € |

# 8 Impacto ambiental

En el presente apartado, se analiza el impacto ambiental a lo largo del desarrollo del proyecto, así como la mejora de aspecto ambiental que puede llevar la realización del proyecto.

Durante el proceso de desarrollar el proyecto, se utilizó una computadora portátil con una potencia máxima de 130 Vatios. Para ello, se calcula la cantidad de dióxido de carbono generado por la energía consumida por la computadora.

Según los datos publicado por la red española de CNMC de 2021 [22], en España, se genera una media de 250 gramos de dióxido de carbono por cada kilovatio-hora de electricidad consumida en el año 2020.

| **Potencia de portátil (W)** | **Horas de trabajo (h)** | **Consumo total (kWh)** | **Factor de emisión de CO2 (kg $CO_2$/kWh)** | **Emisión total (kg $CO_2$)** |
|---|---|---|---|---|
| 130 | 400 | 52 | 0,25 | 13 |

*Tabla 7: Calculo de emisión de $CO_2$*

Como se puede observar en la tabla, durante el desarrollo del proyecto, se ha producido un total de 13 kg de $CO_2$. Se puede considerar que durante el desarrollo del proyecto tiene un impacto muy bajo al medio ambiente.

Por otra parte, el invernadero es un gran consumidor de recursos y el agente que ha sido entrenado correctamente es capaz de utilizar el clima exterior para regular el clima interior con el fin de mantener el clima interior en condición óptima para crecimiento de los cultivos, lo que reduce efectivamente el consumo energético y, por tanto, reduce el riesgo de mal uso de los recursos.

Dado que el entorno externo y el coste de las energías tienen un impacto significativo en el crecimiento y el control óptimo de la lechuga, no es posible calcular directamente el ahorro energético del agente en comparación con otros controles convencionales. Sin embargo, los resultados del capítulo 6 muestran que el agente tiene un buen control de la temperatura y el suministro de calor. Por ello, suponemos que, con el agente podemos ahorrar un 5% del suministro de calor en comparación con el control convencional, y que éste requiere 1kWh de energía por cada metro cuadrado de tierra cultivada al día de valor media. Por otro lado, suponemos que la duración de la lechuga es 65 días desde la plantación hasta la recolección, para ello, el agente puede ahorrar 3,25 kWh de energía por metro cuadrado de cultivo en un ciclo de cultivo, lo que, si el tamaño del invernadero es de 1.000 metros cuadrados, supondría un ahorro de 3,25 MWh de energía.

Aunque el agente es una caja negra para nosotros, una vez que el agente ha sido entrenado, no es nada más que una red neuronal que sólo contiene operaciones matemáticas básicas y no necesita ser actualizada, lo que hace que el agente sea muy rápido en comparación con el MPC para computar y procesar los problemas, lo que también ayuda a reducir el coste de computación y el consumo de energía.

# 9 Conclusiones y trabajos futuros

## 9.1 Conclusión

Al examinar los objetivos fijados al principio del trabajo, podemos decir se cumplen, excepto en el caso del beneficio económico, que requiere un análisis más profundo y se da a continuación.

Hemos completado el diseño del agente y hemos probado la estabilidad del mismo para mantener el clima interior en un nivel confortable a pesar de tener un clima exterior extremo.

En cuanto a los resultados, el agente no ha obtenido tan buenos resultados como el modelo MPC en cuanto a la optimización de la energía, y aunque se ha obtenido más producción, la rentabilidad económica no ha sido tan buena como en el caso del MPC.

Para el modelo de invernadero, es difícil evaluar el controlador en términos de salida debido al gran número de salidas, es decir, es difícil diseñar una función de refuerzo que se pueda juzgar con precisión. Aunque podríamos utilizar directamente la función de beneficio económica como función de refuerzo para el agente, este no es un método fiable por las siguientes razones:

En el aprendizaje por refuerzo no podemos aplicar directamente restricciones a los estados, lo que lleva a que, si no diseñamos un rango pequeño y razonable para estos estados en la función de refuerzo, el agente podrá explorar aquellos estados que son completamente irracionales, como subir la temperatura por encima de los 40 ºC, encender el ventilador al máximo, etc.

La segunda es que, si hay un atajo escondido en la función de refuerzo, el agente irá por ese atajo. Por ejemplo, en el caso de comparación del beneficio económico, si se pone el ventilador al máximo sin inyectar nada de dióxido de carbono y sin encender la calefacción, se obtiene el mayor beneficio económico, que alcanza 1,894 Hfl·$m^2$. Sin embargo, esto hace que la temperatura y la humedad relativa interior se mantengan a un nivel muy peligroso para las lechugas.

Además, con un agente entrenado, es difícil saber si el agente ha convergido a un control óptimo o a un control subóptimo. La solución sólo puede ser dejar que se compare con otros controladores de control óptimo.

En resumen, es muy difícil diseñar un agente perfecto, y el proceso de diseño suele ser un proceso iterativo. Sin embargo, una vez que se obtiene un agente que se pueda considerar correcto, es muy potente, y se pueden obtener resultados de control con la aplicación de muy pocos recursos informáticos y tiempo de cálculo.

## 9.2 Trabajos futuros

Durante el desarrollo del proyecto, hemos identificado una serie de áreas de mejora y de continuación del trabajo:

- Una de las propuestas de continuación de este trabajo sería rediseñar el agente e integrar predicciones del clima futuro como observaciones. En este trabajo, solo ingresamos el clima exterior del momento actual al agente, y la predicción de los estados futuros del agente se basan únicamente en el clima interior y exterior actual y las acciones realizadas. En futuros trabajos, se podría intentar incluir predicciones del

clima exterior futuro como entrada, permitiendo así al agente hacer mejores predicciones de los refuerzos futuros.

- Otra propuesta es utilizar los parámetros de la función de beneficio económico como observaciones del agente y dejar que el agente aprenda a realizar los cambios correspondientes en el uso de la energía cuando estos parámetros cambien.
- También se propone aplicar este problema a los algoritmos de aprendizaje profundo por refuerzos más complejos, como TD3, PPO o SAC, y comparar su rendimiento.

# 10 Referencias

[1] (2019). Creciendo a un ritmo menor, se espera que la población mundial alcanzará 9.700 millones en 2050 y un máximo de casi 11.000 millones alrededor de 2100. ONU.

[2] (2009). La agricultura mundial en la perspectiva del año 2050. Roma: FAO.

[3] (2019). El cambio climático y la tierra. IPCC.

[4] Huang, A. (9 de Agosto de 2016). Transforming the agricultural industry. Obtenido de https://www.ibm.com/blogs/internet-of-things/agricultural-industry/

[5] Boersma, S. (s.f.). Sample-based Nonlinear Model Predictive Control in a Lettuce Greenhouse with Uncertain Weather Forecasting.

[6] F. L. K. Kempkes, J. Janse, and S. Hemming. Greenhouse concept with high insulating double glass with coatings and new climate control strategies; from design to results from tomato experiments. Acta Horticulturae, vol. 1037, pp. 83-92, 2014.].

[7] E. F. Camacho and C. Bordons, Model predictive control. Springer Science & Business Media, 2013.

[8] Reinforcement Learning: An Introduction, R. Sutton, and A. Barto. The MIT Press, Second edition, (2018)

[9] Wang Shusen. (2021). Deep Reinforcement Learning. Obtenido de *https://github.com/wangshusen/DRL/blob/master/Notes_CN/DRL.pdf*

[10] MATLAB & Simulink (2022). Reinfocement Learning With MATLAB. Obtenido de *https://www.mathworks.com/content/dam/mathworks/ebook/gated/reinforcement-learning-ebook-all-chapters.pdf*

[11] MATLAB & Simulink (2022). Reinforcement Leaning Agents. Obtenido de https://es.mathworks.com/help/reinforcement-learning/ug/create-agents-for-reinforcement-learning.html

[12] MATLAB & Simulink (2022). Deep Deterministic Policy Gradient Agents. Obtenido de https://es.mathworks.com/help/reinforcement-learning/ug/ddpg-agents.html

[13] OpenAI (2018). Kinds of RL Algorithms. Obtenido de https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html

[14] *S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. arXiv preprint arXiv:2005.01643, 2020*

[15] MATLAB & Simulink (2022). Deep Deterministic Policy Gradient Agents. Obtenido de https://es.mathworks.com/help/reinforcement-learning/ug/ddpg-agents.html

[16] MATLAB & Simulink (2022). Options for training deep learning neural network. Obtenido de https://es.mathworks.com/help/deeplearning/ref/trainingoptions.html#bu59f0q_sep_mw_b46b4fad-056e-48b4-92c7-6f902d19128d

[17] Rubiales Alberto. (2020). ¿Qué es Underfitting y Overfitting? Obtenido de https://rubialesalberto.medium.com/qu%C3%A9-es-underfitting-y-overfitting-c73d51ffd3f9

[18] MATLAB & Simulink (2022). trainingOptions. Obtenido de https://es.mathworks.com/help/deeplearning/ref/trainingoptions.html#bu59f0q_sep_bu83y_3-1_head

[19] OpenAI (2018). Deep Deterministic Policy Gradient. Obtenido de https://spinningup.openai.com/en/latest/algorithms/ddpg.html#deep-deterministic-policy-gradient

[20] MATLAB & Simulink (2022). rlDDPGAgentOptions. Obtenido de https://es.mathworks.com/help/reinforcement-learning/ref/rlddpgagentoptions.html

[21]Van Henten, E. J., & Bontsema, J. (2009). Time-scale decomposition of an optimal control problem in greenhouse climate management. *Control Engineering Practice*, *17*(1), 88-96.

[22] gencat. (2022). Factor de emisión de la energía eléctrica: el mix eléctrico. Obtenido de https://canviclimatic.gencat.cat/es/actua/factors_demissio_associats_a_lenergia/

[23] Ban, B. (2017). Control of nonlinear, complex and black-boxed greenhouse system with reinforcement learning.

[24] (2019). El estado mundial de la agricultura y la alimentación. FAO.

[25] Mourik, S. v. (2021). Introductory overview: Systems and control methods for operational management support in agricultural production systems. Environmental Modelling and Software.

[26] Subeesh, A. (2021). Automation and digitization of agriculture using artificial. Artificial Intelligence in Agriculture, 278-291.

[27] Shangtong Zhang, Richard S. Sutton (2018) A Deeper Look at Experience Replay

[28] Mathworks. Understanding Model Predictive Control, Part 1: Why Use MPC? Obtenido de https://es.mathworks.com/videos/understanding-model-predictive-control-part-1-why-use-mpc--1526484715269.html

[29] Mathworks. (2022) Train DDPG Agent for Adaptive Cruise Control. Obtenido de https://es.mathworks.com/help/reinforcement-learning/ug/train-ddpg-agent-for-adaptive-cruise-control.html

[30] Mathworks. (2022) Water Distribution System Scheduling Using Reinforcement Learning. Obtenido de https://es.mathworks.com/help/reinforcement-learning/ug/water-distribution-scheduling-system.html