Trabajo Fin de Máster
Máster Universitario en Ingeniería Industrial

Control predictivo de plantas termosolares mediante la aplicación de técnicas de Deep Reinforcement Learning

Autor: Fernando Manuel Borrego Prado

Tutor: José Ramón Domínguez Frejo

Sara Ruiz Moreno

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022

Trabajo Fin de Máster
Máster Universitario en Ingeniería Industrial

Control predictivo de plantas termosolares mediante la aplicación de técnicas de Deep Reinforcement Learning

Autor:

Fernando Manuel Borrego Prado

Tutor:

José Ramón Domínguez Frejo

Contrato Juan de la Cierva - Incorporación

Sara Ruiz Moreno

Personal Investigador en Formación

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022

Trabajo Fin de Máster: Control predictivo de plantas termosolares mediante la aplicación de técnicas de Deep Reinforcement Learning

Autor: Fernando Manuel Borrego Prado
Tutor: José Ramón Domínguez Frejo
Sara Ruiz Moreno

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

A mis padres y a mi hermana
A mis amigos
A Emilia

Agradecimientos

El presente Trabajo Fin de Máster pone fin a esta etapa de mi vida como estudiante universitario y da comienzo, en su lugar, a una aún más apasionante como ingeniero. El mérito de este paso, sin embargo, no solo me pertenece a mí, sino que recae sobre todas aquellas personas que me han brindado su apoyo y cariño durante todos estos años. Por este motivo, no podía dejar pasar la ocasión sin dedicarles mis más sinceros agradecimientos.

En primer lugar, me gustaría agradecerle a mi familia la confianza depositada en mí durante todo este tiempo. Vuestro apoyo y fe incondicional son los motivos que me han hecho llegar a donde estoy hoy. Vosotros sois quienes me han dado la fuerza y confianza necesarias para seguir adelante incluso en los momentos más duros.

También me gustaría darle las gracias a Emilia, compañera durante todo este camino. Gracias por no haber permitido que me perdiera en la aventura que han supuesto estos últimos años y gracias, especialmente, por todos esos consejos que me han ayudado a sacar aún más provecho de ella. Es gracias a ti que a día de hoy sigo disfrutando lo que hago.

A mis amigos, por haber compartido el peso de estos seis largos años y haberlo hecho mucho más llevadero. Habéis hecho de esta etapa de mi vida una experiencia inolvidable. No tengo ninguna duda de que sin vosotros no hubiera significado lo mismo.

Por último, me gustaría darle las gracias a los otros dos artífices de este proyecto: José Ramón y Sara. Gracias a ambos por la paciencia y comprensión de este último año y gracias, en especial, por haberme permitido adentrarme en un mundo que tanta admiración me causaba. Ha sido todo un placer recibir vuestra guía y consejos durante todo este proceso.

Muchas gracias a todos.

Fernando Manuel Borrego Prado
Sevilla, 2022

Resumen

Las tecnologías de generación renovable han experimentado, durante las últimas décadas, un crecimiento sin precedentes en su popularidad. Este crecimiento, motivado por el abaratamiento de los costes y por una necesidad de reducir el uso de aquellas tecnologías causantes, entre otros ejemplos, del cambio climático, ha incentivado el estudio de la tecnología de concentración solar como una alternativa interesante a los medios de generación tradicionales gracias a su capacidad de almacenamiento energético.

El control de este tipo de instalaciones, a menudo, se fundamenta en estrategias de control predictivo basado en modelo (MPC). Sin embargo, debido al excesivo coste computacional derivado de necesitar resolver un problema de optimización a cada instante de muestreo y a la necesidad de un conocimiento tan preciso de las dinámicas del sistema a controlar, esta estrategia de control puede resultar, en ocasiones, ineficiente.

Este trabajo pretende, por tanto, presentar una estrategia de control alternativa a estos controladores MPC utilizados para el control de plantas solares térmicas de concentradores cilindro-parabólicos. Para ello, se propone una estrategia de control basada en técnicas de aprendizaje por refuerzo que permita, además de reproducir los buenos resultados proporcionados por un controlador predictivo, solventar aquellos problemas derivados del mismo. Se desarrolla, para tal fin, un agente de control que, mediante un algoritmo DDPG, dé forma a una ley de control óptima que le permita maximizar la potencia térmica producida en la instalación. Este algoritmo se complementa, además, con el desarrollo de un simulador de la antigua planta experimental ACUREX mediante la librería *OpenAI Gym*, de tal forma que se puedan comprobar los resultados obtenidos.

Abstract

Renewable generation technologies have experienced an unprecedented growth in popularity over the last few decades. This growth, driven by lower costs and a need to reduce the use of those technologies that cause -among others- climate change, has encouraged the study of concentrating solar power technology as an appealing alternative to traditional generation technologies due to its energy storage capacity.

The control of this type of installations is often based on Model Predictive Control (MPC) strategies. However, due to the excessive computational cost of needing to solve an optimization problem at each sampling time and the need for such a precise knowledge of the dynamics of the system to be controlled, this control strategy can sometimes be inefficient.

Therefore, this work aims to present an alternative control strategy to these MPC controllers for controlling parabolic trough collector solar plants. To this end, a control strategy based on reinforced learning techniques is proposed in order to not only reproduce the good results provided by a predictive controller, but also to solve the problems arising from it. Thus, it is developed a control agent which, by means of a DDPG learning algorithm, shapes an optimal control law that maximizes the thermal power produced by the plant. This algorithm is further complemented with the development of an ACUREX simulator using the library *OpenAI Gym*, which allows to check the results achieved.

Índice Abreviado

Índice

Notación

$=$	Igual a.
$\approx$	Aproximadamente igual a.
$\sim$	Similar a, se comporta de acuerdo a.
$\in$	Pertenece a.
$\rightarrow$	Tienda a.
$\forall$	Para todo.
Δ	Incremento.
$\|\|v\|\|$	Norma del vector v.
$\sum_{i=a}^{b}$	Sumatorio desde i igual a a hasta b.
$\log(x)$	Logaritmo natural de x.
$\text{máx}_x f$	Valor máximo de $f(x)$ variando x.
$\text{mín}_x f$	Valor mínimo de $f(x)$ variando x.
$\arg\text{máx}_x f$	Valor de x para el cual $f(x)$ toma su valor máximo.
$\frac{dx}{dy}$	Derivada de x respecto a y.
$\frac{\partial x}{\partial y}$	Derivada parcial de x respecto a y.
$\nabla_{\bar{x}}$	Función gradiente respecto al vector $\bar{x}$.
$\mathbb{P}[X=x\|Y=y]$	Probabilidad de que la variable aleatoria X tome el valor x sujeta a que Y vale y.
$\mathbb{E}[X\|Y=y]$	Esperanza matemática de la variable aleatoria X sujeta a que Y vale y.
μ	Media estadística de una función de distribución de probabilidad.
σ	Desviación típica de una función de distribución de probabilidad.
$\mathcal{N}(\mu,\sigma)$	Función de distribución normal con media μ y desviación típica σ.

Model Predictive Control.

J	Función de coste.
N_p	Horizonte de predicción.
N_u	Horizonte de control.
N_1	Inicio del horizonte de predicción.
N_2	Final del horizonte de predicción.
$u(t)$	Acción de control en el instante t.
$u(i\|t)$	Acción de control en el instante i, estimada en el instante t.
$w(t)$	Referencia de seguimiento en el instante t.
$y(t)$	Variable a controlar en el instante t.
$\hat{y}(i\|t)$	Estimación de la variable a controlar en el instante i, estimada en el instante t.
$\delta(j)$	Peso del error de seguimiento en el instante $t+j$ en la función de coste J.
$\lambda(j)$	Peso del *slew rate* en el instante $t+j-1$ en la función de coste J.

Modelo de parámetros concentrados: parámetros y variables.

A_f	Área de paso del fluido a través de la tubería de un colector [m^2].
A_e	Superficie del colector perdida por efecto de los extremos [m^2].
A_b	Superficie del colector perdida por bloqueos laterales [m^2].
A_s	Superficie del colector perdida por sombras entre colectores [m^2].
C	Capacidad térmica del lazo de colectores [J/°C].
c_{p_f}	Capacidad térmica específica del fluido caloportador [J/kg °C].
EoT	Ecuación del tiempo [min].
f	Distancia focal del colector cilindro-parabólico [m].
G	Dimensión transversal del colector (apertura) [m].
H_l	Coeficiente global de pérdidas térmicas [W/°C].
I_{rad}	Irradiancia solar normal directa recibida por el colector [W/m^2].
JD	Día juliano [días].
K_{opt}	Eficiencia óptica del colector [p.u.].
L_{loop}	Longitud del lazo de colectores [m].
L_{ref}	Longitud geográfica del meridiano central [°].
L_{loc}	Longitud geográfica del meridiano local de la instalación [°].
P_{cp}	Coeficiente de propiedades térmicas del fluido caloportador [J/m^3 °C].
r	Reflectividad de los espejos del colector [p.u.].
S	Superficie total del colector [m^2].
t	Tiempo [s].
t_{local}	Hora local estándar [h].
t_s	Hora solar [h].
T_{amb}	Temperatura ambiente del sistema [°C].
$T_{control}$	Periodo de control [s].
T_f	Temperatura del fluido [°C].
T_{in}	Temperatura del fluido caloportador a la entrada al lazo de colectores [°C].
T_{med}	Temperatura media a lo largo del lazo de colectores [°C].
T_{out}	Temperatura del fluido caloportador a la salida al lazo de colectores [°C].
T_{sample}	Periodo de muestreo [s].
T_{sim}	Periodo de simulación o tiempo de integración del modelo [s].
q	Caudal de fluido caloportador [l/s].
w	Ángulo diario [rad].
W	Dimensión longitudinal del colector [m].
W_{in}	Potencia térmica del fluido caloportador a la entrada del lazo de colectores [W].
W_{out}	Potencia térmica del fluido caloportador a la salida del lazo de colectores [W].
W_{th}	Potencia térmica neta del fluido caloportador [W].
$\alpha_{sol,A}$	Absortancia de la tubería receptora del colector [p.u.].
γ_A	Factor de interceptación del colector [p.u.].
δ_1	Declinación solar [rad].
δ_2	Ángulo horario [rad].
ε	Parámetro de ajuste de la función de recompensa: penalización por *slew rate* [p.u.].
η_o	Eficiencia geométrica de la posición de los espejos respecto a la radiación solar [p.u.].
ρ_f	Densidad del fluido caloportador [kg/m^3].
$\tau_{sol,c}$	Transmisividad de la cubierta de vidrio de la tubería receptora [p.u.].
ϕ	Ángulo de incidencia solar [rad].
ϕ_r	Ángulo de incidencia entre colectores adyacentes [°].
ψ	Parámetro de ajuste de la función de recompensa: penalización por salir del rango óptimo [p.u.].

Reinforcement Learning.

a, a'	Acciones realizadas por el agente.
a_i	Acción realizada por el agente en el instante i.
$a^*(s)$	Acción óptima realizable por el agente desde un estado s.
A_i	Variable aleatoria de la acción realizada por el agente en el instante i.
$\mathcal{A}$	Conjunto de las acciones realizables por el agente.
$A_\pi(a)$	Valor de ventaja de la acción a.
e_t	Experiencia compuesta por la transición $(S_t, A_t, R_{t+1}, S_{t+1})$.
$G_t(\tau)$	Recompensa acumulada obtenida por el agente en su trayectoria τ desde el instante t.
$H(\pi)$	Función de entropía de la política de actuación π.
J	Función objetivo.
$\mathcal{M}$	Conjunto descriptor de un Proceso de Decisión de Markov.
N_{buff}	Tamaño del *buffer* de experiencias.
o	Observación parcial de un estado del escenario.
$P(s', r\|s, a)$	Función de transición del escenario.
$P(s'\|s, a)$, $P^a_{ss'}$	Función de transición de estados del escenario.
$P(\tau\|\pi)$	Función de transición de estados aplicada a la trayectoria τ.
$Q_\theta(s, a)$	Función de aproximación de la función valor estado-acción según los parámetros θ.
$Q_\pi(s, a)$	Valor estado-acción del par estado-acción (s, a) bajo la política π.
$Q_{\pi^*}(s, a)$	Valor estado-acción óptimo del par estado-acción (s, a) bajo la política π^*.
r	Recompensa recibida por el agente desde el escenario.
R_i	Variable aleatoria de la recompensa recibida por el agente en el instante i.
$\mathcal{R}$	Conjunto de recompensas posibles.
$R(s, a)$	Función de recompensa del escenario.
s, s'	Estados del escenario.
s_i	Estado del escenario en el instante i.
S_i	Variable aleatoria del estado del escenario en el instante i.
$\mathcal{S}$	Conjunto de los estados posibles del escenario.
T	Longitud de la trayectoria τ.
$V_\theta(s)$	Función de aproximación de la función valor estado según los parámetros θ.
$V_\pi(s)$	Valor estado del estado s bajo la política π.
$V_{\pi^*}(s)$	Valor estado óptimo del estado s bajo la política π^*.
$y(r, s')$	Valor objetivo de la función valor estado-acción.
α	*Learning rate*, tamaño de paso del aprendizaje.
β	Coeficiente de importancia relativa entre recompensa y entropía de la política de acción.
γ	Factor de descuento de la función de recompensa acumulada.
θ	Conjunto de parámetros que definen una función de aproximación $f(\theta)$.
$\mu(s)$	Política de acción determinista del agente.
$\pi(a\|s)$, π	Política de acción estocástica del agente.
$\pi^*(a\|s)$, π^*	Política de acción estocástica óptima del agente.
$\pi(a\|s;\theta)$, $\pi_\theta(a\|s)$	Función de aproximación de la política del agente según el conjunto de parámetros θ.
τ	Trayectoria descrita por el agente a través del espacio de estados del escenario.
$\tau_{:t}$	Trayectoria descrita por el agente desde el instante inicial hasta un instante t.
$\tau_{t:}$	Trayectoria descrita por el agente desde el instante $t+1$ hasta el instante final T.

Deep Learning.

a_k^l	Datos de salida de la neurona k de la capa oculta l de una red neuronal.
b	Sesgo o *bias* de una neurona.
H	Número de neuronas en la capa oculta de una red neuronal.
$\mathcal{L}$	Función de pérdidas.
w_i	Pesos de ponderación de las variables de entrada a una neurona.
x_i	Variables de entrada a una neurona.
y	Variable de salida de una neurona.
α	*Learning rate*, tamaño de paso del aprendizaje.
ζ	Tamaño de paso de una actualización *Polyak*.
η	Umbral máximo admisible de la norma del vector gradiente de una red neuronal.
μ^l	Media estadística de los datos de salida de las neuronas de la capa l de una red neuronal.
σ^l	Desviación típica de los datos de salida de las neuronas de la capa l de una red neuronal.
$\sigma(\cdot)$	Función de activación de una neurona.

1 Introducción.

You cannot get through a single day without having an impact on the world around you. What you do makes a difference, and you have to decide what kind of difference you want to make.

JANE GOODALL

1.1 Motivación.

Las fuentes de energía renovable se han instaurado, en la actualidad, como una de las herramientas más poderosas en la lucha contra el cambio climático. El acelerado crecimiento del nivel del mar [79], el impacto sobre numerosos ecosistemas naturales [1, 72] y los peligros derivados de un aumento excesivo en las temperaturas ambientales [30] son algunos de los síntomas que han manifestado una necesidad real de buscar alternativas sostenibles en el ámbito de la producción energética.

Históricamente, y a pesar de que las *energías renovables*[1] llevan acompañando al ser humano desde sus inicios, no es hasta la crisis del petróleo de comienzos de la década de 1970 cuando estas tecnologías de generación se empiezan a considerar como posibles alternativas sustitutivas de las tecnologías de generación convencionales [67]. Hasta ese momento, la utilización a gran escala de la energía fósil lograda por los avances tecnológicos de la revolución industrial [9] había conseguido desbancar a las fuentes de energía renovable como medios competitivos en la producción energética. Sin embargo, la excesiva dependencia del petróleo que presentaba la industria por ese entonces, así como el gran aumento del precio del combustible ocasionado por el embargo de petróleo de la Organización de Países Árabes Exportadores de Petróleo (OPAEP) en 1973, hizo despertar una fuerte necesidad tecnológica de nuevas soluciones al problema energético. Bajo esta coyuntura, surgieron nuevas tecnologías para dar respuesta a dicha problemática y se asentaron, así, las bases del desarrollo tecnológico que se sucedería en los próximos años en torno a los medios de generación eléctrica.

En la actualidad, una continua sucesión de avances tecnológicos sigue perfilando estas nuevas formas de generación. Así, puede apreciarse cómo sigue presente la búsqueda tanto de nuevos medios de almacenamiento de energía [73] como de mayores rendimientos a través del estudio de nuevos materiales que permitan aumentar la eficiencia de los paneles solares [91], se sigue abogando por la reducción de costes mediante una recopilación de las mejores prácticas de explotación [21, 48], e incluso se siguen desarrollando nuevas tecnologías de generación como la eólica *off-shore* [34] y la mareomotriz [17]. Con todo esto, la generación eléctrica renovable no solo se ha convertido en una aliada imprescindible para combatir el cambio climático, sino que ha ofrecido una opción sostenible en el tiempo, crecientemente competitiva, diversa y políticamente favorable en el panorama de la generación energética [64], llegando incluso, en el caso de la energía eólica y fotovoltaica, a costes LCOE más bajos que en los medios de generación convencionales [24].

[1] Las energías renovables se definen en [78] como *"aquellas energías que se obtienen a partir de corrientes de energía continuas y recurrentes en el mundo natural"*. A partir de esta definición, tecnologías tan dispares como el aprovechamiento energético de la fuerza del viento para la propulsión de barcos de vela o la utilización del efecto fotoeléctrico para la conversión de radiación solar en electricidad quedarían recogidas bajo el marco de estas energías.

Este cúmulo de circunstancias ha ocasionado que las energías renovables hayan irrumpido en nuestros días con gran fuerza. Tanto es así que, desde la década de 1990, es posible apreciar una tendencia ascendente cada vez más clara en la adopción e implementación de tecnologías de generación energética renovable en detrimento de aquellas basadas en combustibles fósiles. Así, en la Figura 1.1, se observa cómo en los últimos diez años se ha producido una caída de un 62.4 % y un 43.0 % en la producción europea de energía a partir de combustibles como el gas natural o carbón respectivamente, mientras que la producción a partir de fuentes renovables ha llegado a crecer en un 39.2 % [23].

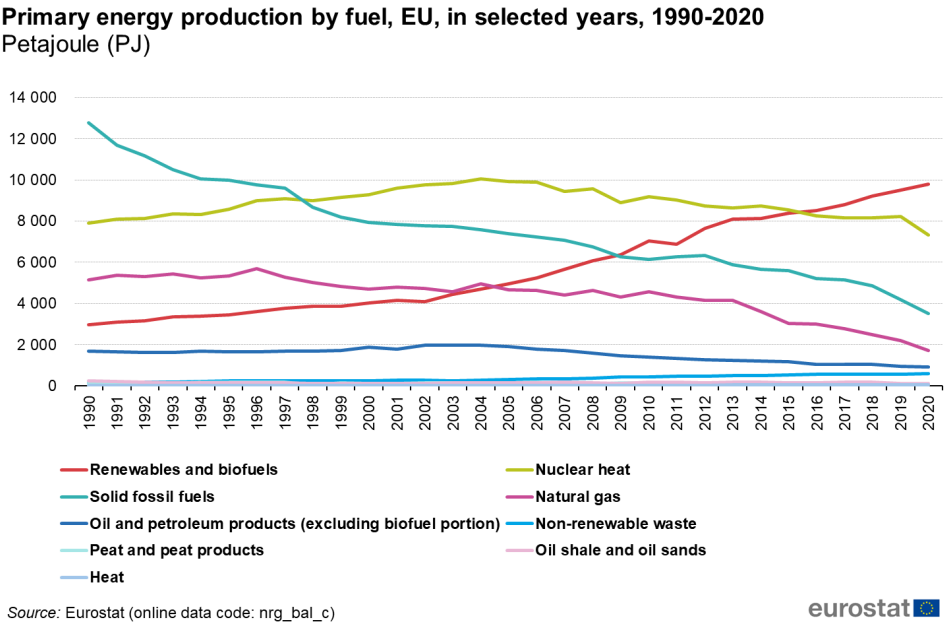


Figura 1.1 Producción de energía primaria en Europa (1990-2020) en función del combustible utilizado [23].

Esta creciente popularidad de las fuentes de energía renovable, junto a las previsiones de aumento de la demanda energética en los próximos años [27], genera unas perspectivas muy favorables para el desarrollo de estas nuevas tecnologías.

Uno de los puntos fuertes de estas nuevas formas de generación es la diversidad que ofrece en cuanto a la fuente de energía primaria utilizada. Así, pueden encontrarse tecnologías basadas en la energía cinética y potencial del agua, en la energía térmica irradiada por la Tierra, en la energía cinética ocasionada por el soplo del viento o incluso en la energía que es irradiada por el Sol. En concreto, puede observarse en la Figura 1.2 que estas dos últimas son las más populares: energía eólica y energía solar.

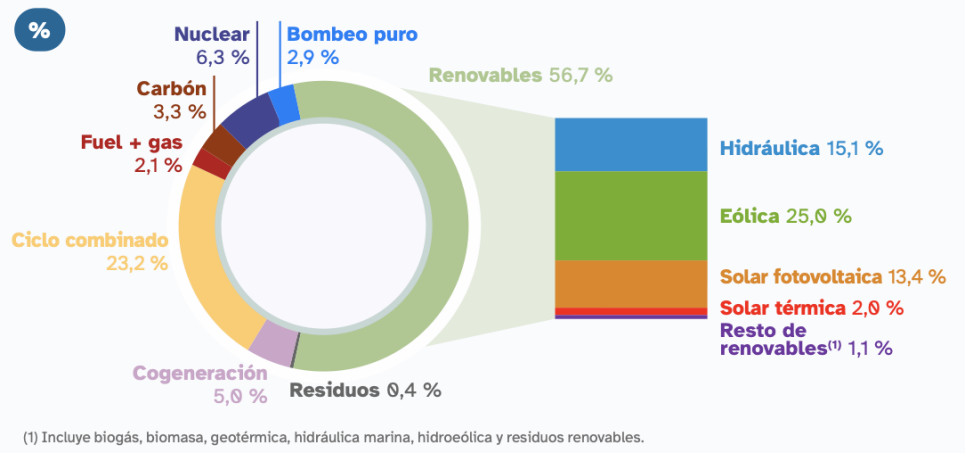


Figura 1.2 Estructura de la potencia eléctrica instalada en España (2021) [59].

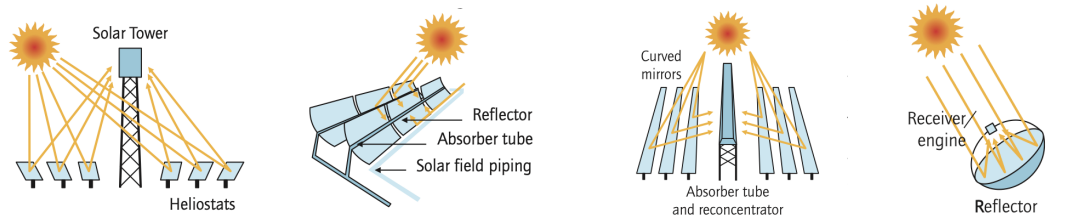
1.1.1 Tecnologías de concentración solar.

En concreto, la fuente de energía que más está aumentando su uso en los últimos años es la energía solar. La radiación electromagnética emitida por el Sol, y que llega a la Tierra en forma de luz y calor, puede ser recogida y convertida en otras formas de energía más útiles para el ser humano utilizando diferentes técnicas [18]:

- Efecto fotoeléctrico. El uso de paneles solares fotovoltaicos o, en inglés, *Photovoltaic* (PV) *Cells*, consigue convertir en corriente continua la radiación solar que incide sobre una lámina de material semiconductor. Los últimos avances, además, plantean la posibilidad de usar concentradores solares para lograr un aumento de la eficiencia obtenida durante la conversión energética [55].
- Efecto termoeléctrico. Existen sensores solares termoeléctricos mediante los cuales una diferencia de temperatura entre los extremos del dispositivo puede ser convertida en una diferencia de potencial eléctrico. No obstante, la eficiencia de estos sistemas aún sigue siendo muy baja como para ser considerada una tecnología competitiva.
- Celda solar Grätzel o célula solar sensibilizada por colorante. Estas celdas están constituidas por un sistema foto-electro-químico que aprovecha la excitación del colorante con la radiación solar para generar electricidad. Así, la liberación de electrones por parte de dicho colorante, y el movimiento de estos a través de un material semiconductor y una solución electrolítica, generan una corriente eléctrica que permite la generación de electricidad.
- Concentración solar. La tecnología de concentración solar o, en inglés, *Concentrated Solar Power* (CSP), se fundamenta en la concentración de la radiación solar en una superficie de menor tamaño. De esta forma, se permite una transmisión de calor, y de energía, más eficiente hacia un fluido caloportador. A partir de dicho fluido, será posible bien la obtención de energía eléctrica al hacerlo pasar a través de una turbina, o bien la obtención de energía térmica si se utiliza un intercambiador de calor.

A pesar de tan amplia variedad de tecnologías, dos de ellas se caracterizan por ser las más populares en el mercado: la tecnología fotovoltaica (PV) y la tecnología de concentración solar (CSP), existiendo incluso ciertos estudios que tratan la combinación de ambas [35]. De esta forma, y a pesar de que la tecnología fotovoltaica se ha instaurado a nivel mundial como el medio de generación renovable por excelencia gracias a su bajo coste, resultan interesantes las oportunidades ofrecidas por la tecnología termosolar en cuanto al almacenamiento de energía térmica. La integración de este tipo de almacenamiento -*Thermal Energy Storage* (TES)- a gran escala permite convertir la inestabilidad característica de la energía solar en un suministro de potencia más estable. Así, este tipo de plantas aportan la flexibilidad y seguridad necesarias para una futura penetración masiva de las energías renovables en el sistema eléctrico.

Existen, en la literatura [90], distintas tecnologías de generación CSP en función del sistema utilizado para concentrar la radiación solar. El alcance de este trabajo explora, específicamente, las posibilidades que brinda la tecnología de *Parabolic Trough Collector* (PTC), que consiste en una serie de colectores solares (o espejos) con forma cilindro-parabólica que concentran la radiación solar en un tubo situado en el eje focal de estos.



(a) Solar Power Tower, SPT. **(b)** Parabolic Trough Collector, PTC. **(c)** Linear Fresnel Reflector, LFR. **(d)** Parabolic Dish System, PDS.

Figura 1.3 Tecnologías CSP: a) SPT, b) PTC, c) LFR, d) PDS. [90].

El control de este tipo de sistemas, centrado en gestionar el caudal de fluido caloportador que atraviesa los diferentes colectores de la instalación, ha sido objeto de estudio en los últimos años. Así, se han ido desarrollando distintas estrategias de control, desde estrategias de control clásicas basadas en controladores PID en [11], hasta técnicas más modernas y potentes como el control predictivo, o *Model Predictive Control* (MPC), en [12] e incluso técnicas basadas en aprendizaje automático y redes neuronales en [60].

1.1.2 Machine Learning.

Los últimos avances en el campo de la inteligencia artificial han demostrado que existe cabida para este tipo de técnicas en prácticamente cualquier sector industrial. La detección de fallos estructurales en pavimentos o el control de calidad a partir de análisis de imágenes [4, 75], la detección temprana de enfermedades [5] o el seguimiento de animales [54] son algunas de las evidencias que dejan claras las infinitas posibilidades que ofrecen estas herramientas.

No obstante, uno de los terrenos aún más inexplorados por las técnicas de aprendizaje automático, o *Machine Learning* (ML), es su aplicación en el control de sistemas físicos. Un ejemplo de aproximación a este ámbito es el sugerido en [60], donde se desarrollan distintas redes neuronales que buscan servir de controladores sustitutivos a los más clásicos controladores predictivos. Sin embargo, aunque este enfoque consigue una mayor agilidad en la ejecución del algoritmo de control en tiempo real, sigue presentando el inconveniente de necesitar de una base de datos suficientemente amplia con la que aprender y, por tanto, de haber ejecutado el controlador predictivo previamente.

1.2 Objetivo.

La propuesta de este trabajo es, por tanto, el desarrollo de un controlador sustitutivo a los controladores predictivos en plantas termosolares. Para ello, se hará uso de una de las últimas tendencias en el aprendizaje automático: el *Deep Reinforcement Learning* (DRL). El DRL, o aprendizaje por refuerzo, es una rama del aprendizaje automático que considera el problema de un agente que es capaz de aprender a partir del *ensayo y error*. De esta forma, no es necesaria la existencia de una verdad básica o *ground truth* previo, sino que el agente es capaz de desarrollar estrategias de decisión a partir de una serie de interacciones con su entorno y una función de recompensa que indique cómo de buena o mala ha sido dicha interacción.

Así, mediante este acercamiento, se pretende reducir el coste computacional asociado a la resolución de problemas de optimización multivariables y no lineales que tiene lugar en los algoritmos de control predictivo. Se busca, con esto, dotar al sistema físico de controladores más ágiles y flexibles, capaces de inferir el comportamiento del sistema en base a la experimentación.

La idea tras este trabajo es, por consiguiente, formular una arquitectura de control donde el agente de control sustituya al control predictivo clásico.

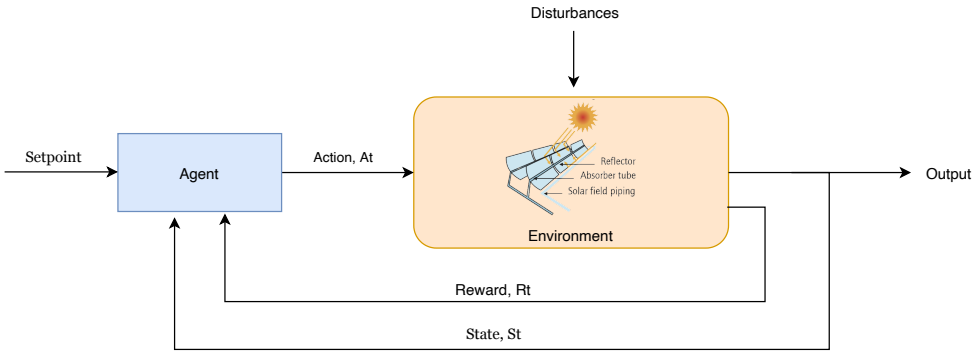


Figura 1.4 Implementación de la estructura de control del sistema.

Para llevar a cabo este objetivo, se utiliza como caso de estudio la antigua planta solar de experimentación ACUREX. Esta planta, que se situó en la Plataforma Solar de Almería (PSA), consistie en una planta de la década de 1980 basada en tecnología PTC que ha servido, durante los años, como modelo de referencia para las distintas investigaciones en el ámbito del control de plantas termosolares.

1.3 Estructura.

La estructura seguida en el presente trabajo, y en la que se trata de desarrollar todos los objetivos descritos anteriormente, es la siguiente:

- En el Capítulo 2 se presenta una introducción al estado del arte de las diferentes herramientas del aprendizaje automático. Entre ellas, se pone especial atención en aquellas técnicas enmarcadas bajo la corriente del aprendizaje por refuerzo y se exponen, pormenorizadamente, los conceptos teóricos y matemáticos necesarios para facilitar la posterior compresión del sistema de control modelado. Asimismo, se introduce el método de control predictivo basado en modelo como un controlador de referencia con el que comparar los resultados obtenidos por el agente de control desarrollado utilizando las técnicas de aprendizaje reforzado. Finalmente, se recoge una breve comparativa de ambos enfoques adoptados, destacando las diferencias y las ventajas y desventajas de cada uno de ellos.

- En el Capítulo 3 se presenta un modelo sencillo de la planta termosolar ACUREX basado en una descripción de parámetros concentrados. Así, en este capítulo se desarrolla el conjunto de ecuaciones que permite describir el comportamiento dinámico del sistema a controlar. Estas ecuaciones, además, se complementan con ciertas consideraciones prácticas que han de ser tenidas en cuenta para permitir la posterior implementación de las mismas en un entorno de simulación funcional. Se añade, por último, una relación de todos los parámetros utilizados en el modelo para facilitar al lector la réplica de los resultados obtenidos en posteriores capítulos.

- El Capítulo 4 desarrolla el algoritmo de aprendizaje reforzado utilizado como cerebro del agente de control. Este algoritmo es el encargado de hacer que el agente en cuestión sea capaz de elaborar una política de acción que emule el comportamiento del controlador tomado como referencia: un controlador predictivo o MPC. Se presenta en este capítulo, además, un entorno de simulación funcional de la planta termosolar ACUREX. Para ello, es introducida una librería de Python, *Gym*, que permite la estandarización de dicho modelo para su uso en algoritmos de aprendizaje por refuerzo.

- En el Capítulo 5 se presentan, finalmente, los resultados obtenidos del entrenamiento del agente de control y de la evaluación de su interacción con el entorno de simulación. Además, se establecen diferentes métricas que valoran el desempeño de esta nueva filosofía de control, comparando los resultados con aquellos obtenidos por un controlador predictivo MPC.

- El Capítulo 6 pone fin al trabajo recogiendo las conclusiones obtenidas a lo largo de todo su desarrollo, así como aportando distintas perspectivas sobre futuros trabajos que permitan ampliar este campo de investigación.

2 Deep Reinforcement Learning.

Machine intelligence is the last invention that humanity will ever need to make.

Nick Bostrom

Este capítulo presenta una breve introducción al *Deep Reinforcement Learning*, así como a los fundamentos de aquellos algoritmos de aprendizaje reforzado utilizados en el control de la planta termosolar abordada en este trabajo. El capítulo se completa, además, realizando una rápida comparación entre este tipo de métodos y las técnicas de control predictivo que serán utilizadas como referencia más adelante en el documento.

2.1 Contextualización.

Tras la reflexión "*Can machines think?*" presentada por el matemático inglés Alan Turing en [76], se despierta en la comunidad científica un interés por descubrir las capacidades de los ordenadores en la consecución de aquellas tareas denominadas, hasta el momento, inteligentes. Sin embargo, no sería hasta 1956 [32] cuando el estadounidense John McCarthy le da nombre a esta disciplina: *Artificial Intelligence* (AI). La inteligencia artificial se instauraría, de esta forma, como una rama de la computación cuyo objetivo es el de replicar el pensamiento humano en los ordenadores.

El término "inteligencia artificial", sin embargo, ha ido evolucionando y extendiéndose durante las últimas décadas. Tanto es así que, en la actualidad, es posible identificar multitud de diferentes técnicas y sistemas que presentan características propias del comportamiento humano, como la computación cognitiva, visión artificial, aprendizaje automático, redes neuronales, aprendizaje profundo o procesamiento del lenguaje natural, entre otras [37].

El aprendizaje automático o ML es una de las ramas de la AI que mayor importancia ha cobrado en los últimos años. Esta disciplina, enfocada en dotar a las máquinas de la capacidad de realizar tareas para las cuales no han sido explícitamente programadas, se centra en el desarrollo de diferentes algoritmos software capaces de identificar y aprender patrones en unos datos de entrada y de tomar decisiones en consecuencia. Estos algoritmos son, además, frecuentemente acompañados por técnicas de *Deep Learning* (DL), lo que les permite aumentar el tamaño del *dataset* analizado e identificar patrones en cantidades más masivas de datos.

Dentro de la rama del ML es posible, de igual forma, encontrar diferentes categorías de algoritmos según el tipo de dato utilizado para el aprendizaje [61]. Según la Figura 2.1, es posible distinguir entre:

- **Aprendizaje supervisado.** El aprendizaje supervisado es aquel constituido a partir de algoritmos que utilizan un conjunto de datos etiquetados para inducir de ellos una función objetivo. Estos datos constan de unos pares de entradas y salidas relacionadas entre sí, de forma que, de manera iterativa, el algoritmo trata de hallar la función que relaciona ambos pares. El proceso por el que tiene lugar este aprendizaje se denomina *training*.

- **Aprendizaje no supervisado.** El aprendizaje no supervisado, por su parte, no necesita de estos *datasets* etiquetados. Por el contrario, en este tipo de técnicas es el propio algoritmo el que intenta dar sentido a los datos extrayendo, de estos, características comunes, tendencias o estructuras implícitas del *dataset*.

- **Aprendizaje semi-supervisado.** Las técnicas de aprendizaje semi-supervisado se encuentran a medio camino entre las dos anteriores. Este tipo de técnicas operan tanto con *datasets* etiquetados como no etiquetados y su objetivo es el de apoyar al proceso de aprendizaje supervisado clásico de un algoritmo con grandes cantidades de datos no etiquetados. Esto permite aumentar el conjunto de datos disponibles y, con ello, la precisión lograda en el aprendizaje.

- **Aprendizaje reforzado.** Por último, el aprendizaje reforzado (o aprendizaje por refuerzo) se ocupa de determinar qué acciones debe escoger un agente de control para alcanzar una política de actuación óptima en un contexto o escenario determinado. El funcionamiento de este tipo de técnicas esta basado en la psicología conductista[1] y su objetivo último es el de maximizar la recompensa obtenida por el agente a lo largo de su interacción con el entorno.

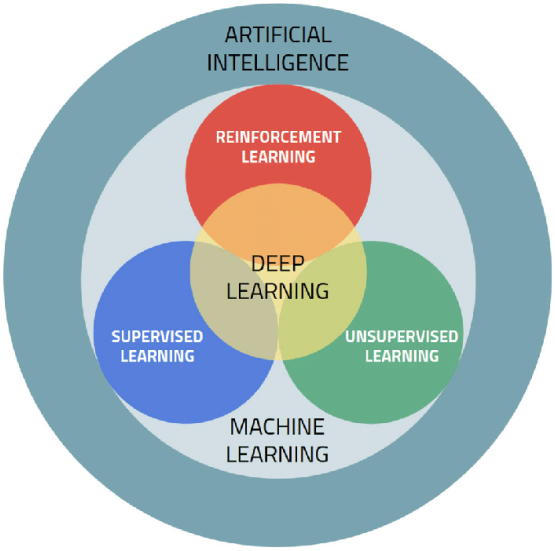


Figura 2.1 Relación entre disciplinas: AI, ML, DL y RL [36].

Por lo general, todos estos algoritmos configuran su estructura interna de tal forma que permitan alcanzar, a partir de un conjunto de datos de entrada, la salida deseada. Sin embargo, no es posible garantizar la efectividad y eficiencia de estos algoritmos en la resolución de un problema, sino que esta dependerá, principalmente, de la naturaleza de los datos y el objetivo perseguido en la solución. Esto implica, por tanto, que no existe algoritmo de *Machine Learning* alguno que sea universalmente mejor que el resto en todos los problemas (*No Free Lunch theorem*, [2]).

El enfoque de este trabajo, como ya se introdujo en el Capítulo 1, es explorar las posibilidades del DRL en el ámbito del control automático. En la literatura ya es posible encontrar diversos artículos que estudian la aplicación de estas técnicas a la robótica [39], al control de multirrotores [33] o incluso al control de procesos en la industria química [42, 71]. No obstante, la utilización de este tipo de técnicas dentro del ámbito del control de centrales de producción de potencia aún es un terreno inexplorado.

[1] El conductismo, surgido en 1913 de la mano de J.B. Watson a través de su artículo "*Psychology as the behaviorist views it*" [81], hace referencia al estudio experimental objetivo y natural de la conducta. En esta disciplina se utiliza el término "condicionamiento operante", término por el cual se define un proceso de aprendizaje en el que una acción es más factible de ser realizada si esta es seguida de algún estímulo deseable, mientras que si dicha acción es penalizada mediante estímulos no deseables, esta será menos factible de repetirse.

2.2 Reinforcement Learning.

El *Reinforcement Learning* (RL) es una disciplina dentro del *Machine Learning* que fundamenta su proceso de aprendizaje en un mecanismo de recompensas con el que, a través de un proceso de ensayo y error, el agente es capaz de discernir qué acciones son más favorables en un determinado contexto. De esta forma, y a diferencia de otras disciplinas como el aprendizaje supervisado, el agente no recibe directamente indicaciones sobre las acciones a tomar, sino que este es capaz de inducirlas a partir del *feedback* recibido por el escenario. El objetivo del aprendizaje de este agente será, por tanto, desarrollar una política de acción que permita maximizar la recompensa obtenida durante la interacción con el escenario.

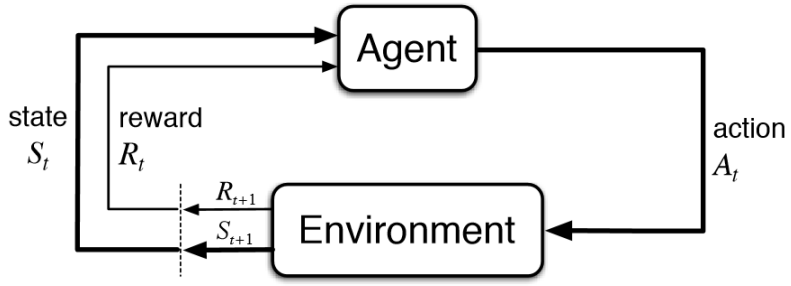


Figura 2.2 Interacciones principales entre agente y escenario [70].

La Figura 2.2 muestra las principales interacciones que tienen lugar entre el agente (sujeto que toma las decisiones) y el escenario (contexto dentro del cual se desenvuelve el agente y con el que interactúa) durante el proceso de aprendizaje. En este lazo de interacciones se pueden distinguir los siguientes elementos:

- **Estado**, $s \in \mathcal{S}$.

  El estado s de un escenario corresponde a una descripción completa del conjunto de variables que definen a dicho escenario. En ocasiones, puede resultar no ser identificable la totalidad de estas variables, sino tan solo una descripción parcial de las mismas. En estos casos, donde parte de la información se encuentra oculta para el agente, se habla de una descripción parcial del estado u observación o.

- **Acción**, $a \in \mathcal{A}$.

  El agente, conociendo el estado s que describe la situación del escenario, ha de tomar una acción a sobre este y observar su evolución. Esta acción a será la causante de llevar al escenario desde un estado s a un estado s'. A menudo, se utiliza el término trayectoria, τ, para describir la secuencia de estados y acciones que se toman en un escenario determinado. Esta trayectoria viene descrita por la siguiente expresión:

  $$\tau = (s_0, a_0, s_1, a_1, ...) \tag{2.1}$$

  La elección de la acción a realizar sobre el escenario, por su parte, es escogida a partir de las pautas proporcionadas por el cerebro del agente: la política de actuación. Esta política de actuación, o *policy*, se encarga de trazar una relación entre cada uno de los estados posibles del escenario y la acción óptima a realizar por el agente en cada caso. En función de si esta política, dado un estado de partida s, proporciona de forma unívoca una única acción (política determinista) o bien la probabilidad de que una determinada acción a sea escogida como óptima (política estocástica), se distingue entre:

  $$\begin{cases} \text{Política determinista:} & \mu(s) = a \\ \text{Política estocástica:} & \pi(a|s) = \mathbb{P}[A_t = a | S_t = s] \end{cases} \tag{2.2}$$

  Finalmente, también es posible distinguir, en función de si las indicaciones de esta política son utilizadas o no en el aprendizaje del algoritmo, entre algoritmos *on-policy* (aquellos que utilizan la salida de la política de actuación en el proceso de aprendizaje) y *off-policy* (aquellos que utilizan parámetros diferentes a la política de actuación) [82].

- **Recompensa**, $r \in \mathcal{R}$.

  La recompensa r, por último, no es más que una medida de cómo de acertada es la acción tomada por el agente en función del estado en el que se encuentra el escenario. En la práctica, dado que el objetivo del agente es maximizar la recompensa obtenida durante toda la interacción con el escenario, resulta muy útil definir una función de recompensa acumulada durante la trayectoria τ a partir de un instante t:

$$G_t(\tau) = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.3}$$

  donde $\gamma \in [0,1]$ corresponde al factor de descuento. Este factor de descuento resulta de gran importancia en las técnicas de aprendizaje por refuerzo ya que cumple una doble función: por un lado, permite dar una mayor importancia a las recompensas más cercanas en el tiempo, penalizando aquellas más lejanas debido a la mayor incertidumbre que presentan; y, por otro, permite garantizar la convergencia de la función $G_t(\tau)$.

2.2.1 Conocimiento del modelo.

Se denomina modelo al descriptor del escenario, esto es, al conjunto de ecuaciones que permiten describir el comportamiento y evolución de este. En un modelo se pueden distinguir dos partes fundamentales:

- **Función de transición**. La función de transición, $P(s', r|s, a)$, es aquella que determina la probabilidad de que un estado s de lugar a un estado s' y una recompensa r a consecuencia de la acción a .

$$P(s', r|s, a) = \mathbb{P}[S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a] \tag{2.4}$$

  A partir de esta función, se calcula la conocida como función de transición de estado:

$$\begin{aligned} P^a_{ss'} = P(s'|s, a) &= \mathbb{P}[S_{t+1} = s'|S_t = s, A_t = a] \\ &= \sum_{r \in \mathcal{R}} P(s', r|s, a) \end{aligned} \tag{2.5}$$

- **Función de recompensa**. La función de recompensa, por su parte, es aquella encargada de asignar una recompensa r a cada transición entre estados s y s' originada por la influencia de la acción a.

$$\begin{aligned} R(s,a) &= \mathbb{E}[R_{t+1}|S_t = s, A_t = a] \\ &= \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} P(s', r|s, a) \end{aligned} \tag{2.6}$$

Estas dos funciones, sin embargo, no siempre son conocidas en todos los modelos estudiados. Surgen, por tanto, dos enfoques diferentes a la hora de diseñar un agente: agentes basados en modelo o *model-based* y agentes *model-free*. Por un lado, los algoritmos *model-based* son aquellos que disponen de toda la información del modelo o, lo que es lo mismo, aquellos en los que se conocen a la perfección las funciones de transición y de recompensa que rigen su comportamiento. En este tipo de algoritmos es posible hacer uso de dichas funciones para desarrollar una política de actuación en la que, tras conocer la respuesta del sistema ante distintos estímulos de entrada, se escoja la acción que proporcione mejores resultados. Por el contrario, los algoritmos *model-free* no cuentan con este conocimiento y, por tanto, deben buscar formas alternativas de predecir el comportamiento del escenario. De manera más habitual, el enfoque utilizados para conseguir dicho objetivo es el de utilizar aproximadores que permitan modelar el comportamiento de las funciones valor estado y valor estado-acción del sistema.

Por último, y a pesar de que pudiera parecer que el desarrollo de algoritmos basados en modelo es más popular debido a las ventajas ofrecidas en cuanto a la previsibilidad del comportamiento del sistema, cabe destacar una predominancia de los algoritmos *model-free* en la literatura. Así, mientras que los algoritmos *model-based* presentan problemas de sobreajuste al modelo utilizado (además de necesitar disponer de modelos muy precisos), los algoritmos *model-free* son generalmente más eficientes y sencillos de implementar.

2.2.2 Valor estado y valor estado-acción.

Se le asocia, a cada estado s alcanzado por el escenario, un valor que representa la bondad del mismo en función de aquellas recompensas futuras que se esperan recibir si se sigue, desde dicho estado, la política π. A este valor se le denomina valor estado y es representado por $V_\pi(s)$.

$$V_\pi(s) = \mathbb{E}_{\tau\sim\pi}[G_t(\tau)|S_t = s] \tag{2.7}$$

De la misma forma, se puede definir el valor estado-acción. En este caso, el valor estado-acción proporciona información acerca de la bondad de una acción a. Así, a cada pareja estado-acción se le asigna un valor en función de las recompensas futuras que pudieran ser obtenidas si, tras partir del estado s y realizar la acción a, se continúa con la política π.

$$Q_\pi(s,a) = \mathbb{E}_{\tau\sim\pi}[G_t(\tau)|S_t = s, A_t = a] \tag{2.8}$$

Uniendo las ecuaciones (2.7) y (2.8) es posible encontrar una relación entre ambas magnitudes:

$$V_\pi(s) = \sum_{a\in\mathcal{A}} Q_\pi(s,a)\pi(a|s) \tag{2.9}$$

Finalmente, y a partir de las definiciones anteriores, se puede determinar cómo de buena resulta la acción a con respecto al resto de acciones posibles en $\mathcal{A}$. Para ello, se define la función de *advantage*:

$$A_\pi(a) = Q_\pi(s,a) - V_\pi(s) \tag{2.10}$$

2.2.3 Procesos de Decisión de Markov.

Atendiendo a la formulación utilizada para describir un problema de *Reinforcement Learning*, es posible encontrar una analogía con los denominados Procesos de Decisión de Markov. Un Proceso de Decisión de Markov, o *Markov Decision Process* (MDP), consiste en la representación matemática de un sistema de decisión secuencial estocástico donde la función de transición entre estados del sistema, así como el coste asociado a dicha transición, dependen únicamente del estado actual del sistema y la acción tomada [56].

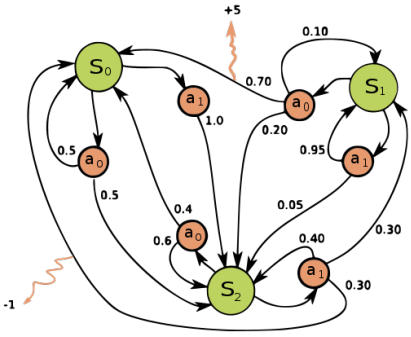


Figura 2.3 Representación de un MDP con tres estados (verde) y dos acciones por estado (naranja) [84].

Un MDP, por tanto, puede representarse mediante el siguiente conjunto de elementos:

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle \longrightarrow \begin{cases} \mathcal{S} \rightarrow \text{Conjunto de estados } s \text{ posibles, } s \in \mathcal{S}. \\ \mathcal{A} \rightarrow \text{Conjunto de acciones } a \text{ posibles, } a \in \mathcal{A}. \\ P \rightarrow \text{Función de transición de estado: } P(s'|s,a) = \sum_{r\in\mathcal{R}} P(s',r|s,a). \\ R \rightarrow \text{Función de recompensa: } R(s,a) = \sum_{r\in\mathcal{R}} r \sum_{s'\in\mathcal{S}} P(s',r|s,a). \\ \gamma \rightarrow \text{Factor de descuento, } \gamma \in [0,1]. \end{cases} \tag{2.11}$$

El objetivo de un MDP, al igual que el de un problema de *Reinforcement Learning*, es el de encontrar una política óptima de actuación π^* que permita maximizar las recompensas futuras acumuladas por el agente que toma las decisiones.

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau\sim\pi}[G_t(\tau)] \tag{2.12}$$

De esta forma, siguiendo como política de actuación la política π^*, el agente será capaz de identificar, dado un estado s, tanto aquellas acciones que le aportan un mayor beneficio al agente como aquellos estados s' que resultan más óptimos. Esto es:

$$\pi^* = \arg\max_{\pi} V_\pi(s) = \arg\max_{\pi} Q_\pi(s,a) \longrightarrow \begin{cases} V_{\pi^*}(s) = \max_{\pi} V_\pi(s) \\ Q_{\pi^*}(s,a) = \max_{\pi} Q_\pi(s,a) \end{cases} \tag{2.13}$$

Finalmente, es necesario tener en cuenta que no todos los problemas pueden modelarse como un Proceso de Decisión de Markov. Así, existen sistemas físicos reales donde las dinámicas propias del sistema imposibilitan una descripción tal que la evolución futura del sistema sea independiente de estados pasados del mismo. Una solución útil para este tipo de situaciones es utilizar históricos de aquellas variables más relevantes como parte del estado del sistema, de forma que el agente pueda intuir, de ellos, estas dinámicas existentes.

2.2.4 Ecuaciones de Bellman.

Las ecuaciones de Bellman son un conjunto de ecuaciones que facilitan la resolución de un MDP. Estas ecuaciones permiten establecer una formulación recursiva de los valores estado y estado-acción [82], de forma que *el valor estado del estado de partida se estime como la recompensa que se espera obtener por estar ahí, más el valor del estado alcanzado a continuación* [51].

Así, partiendo de la ecuación (2.7), la función valor estado puede entenderse como una suma recursiva de los valores estado de cada uno de los estados alcanzados por el agente.

$$\begin{aligned} V_\pi(s) &= \mathbb{E}_{\tau\sim\pi}[G_t(\tau)|S_t = s] \\ &= \mathbb{E}_{\tau\sim\pi}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ...|S_t = s] \\ &= \mathbb{E}_{\tau\sim\pi}[R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + ...)|S_t = s] \\ &= \mathbb{E}_{\tau\sim\pi}[R_{t+1} + \gamma G_{t+1}(\tau)|S_t = s] \\ &= \mathbb{E}_{\tau\sim\pi}[R_{t+1} + \gamma V_\pi(S_{t+1})|S_t = s] \end{aligned} \tag{2.14}$$

De la misma forma, y utilizando esta recursividad, el valor estado-acción queda representado tal que:

$$\begin{aligned} Q_\pi(s,a) &= \mathbb{E}_{\tau\sim\pi}[G_t(\tau)|S_t = s, A_t = a] \\ &= \mathbb{E}_{\tau\sim\pi}[R_{t+1} + \gamma V_\pi(S_{t+1})|S_t = s, A_t = a] \\ &= \mathbb{E}_{\tau\sim\pi}[R_{t+1} + \gamma Q_\pi(S_{t+1}, A_{t+1})|S_t = s, A_t = a] \end{aligned} \tag{2.15}$$

Finalmente, utilizando tanto la definición de $Q(s,a)$ como la relación de la ecuación (2.9), este proceso iterativo se puede descomponer en el siguiente conjunto de ecuaciones que relacionan $V_\pi(s)$ y $Q_\pi(s,a)$.

$$\begin{cases} V_\pi(s) = \sum_{a\in\mathcal{A}} \overbrace{\left(R(s,a) + \gamma \sum_{s'\in\mathcal{S}} P^a_{ss'} V_\pi(s')\right)}^{Q_\pi(s,a)} \pi(a|s) \\ Q_\pi(s,a) = R(s,a) + \gamma \sum_{s'\in\mathcal{S}} P^a_{ss'} \underbrace{\sum_{a'\in\mathcal{A}} \pi(a'|s') Q_\pi(s',a')}_{V_\pi(s')} \end{cases} \tag{2.16}$$

Resolviendo estas ecuaciones para una política óptima π^*, es posible determinar la trayectoria óptima que otorga una mayor recompensa al agente. No obstante, estas ecuaciones son difícilmente utilizables en la práctica debido a que, por un lado, es necesario conocer a la perfección el modelo del escenario con el que se interactúa y, por otro, a que, en aquellos problemas más complejos, se incurre en un alto coste computacional al almacenar memoria de todas las funciones valor mencionadas.

2.2.5 Algoritmos de resolución.

Los algoritmos utilizados para la resolución de un problema de RL se distinguen en función de si se persigue un enfoque *model-based* o uno *model-free*. En este trabajo, dado que los algoritmos más desarrollados en la literatura son los referidos a este segundo grupo, se opta por estudiar las opciones de implementar un agente de control que no utilice información alguna concerniente al modelo del sistema para el desarrollo de su política de actuación.

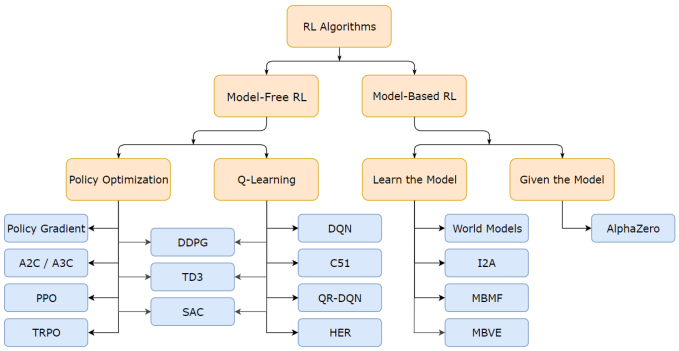


Figura 2.4 Algoritmos más populares en *Reinforcement Learning* [50].

Se identifican dos ramas principales dentro del estudio de los algoritmos de aprendizaje reforzado *model-free*: métodos basados en valor (o *Q-Learning*) y métodos basados en política (o *Policy Optimization*). No obstante, ambos métodos no son excluyentes, permitiendo a algunos algoritmos situarse en un punto intermedio.

Q-Learning.

Los métodos recogidos bajo el ámbito de los algoritmos *Q-Learning* son aquellos en los que el agente utiliza la función valor-estado $Q_\pi(s,a)$ para tomar la decisión en cuanto a qué acción realizar.

$$a^*(s) = \arg\max_a Q_\pi(s,a) \tag{2.17}$$

Sin embargo, dado que el problema en cuestión se trata de un problema *model-free*, no es posible conocer el valor real tomado por esta función $Q_\pi(s,a)$. Se necesita, por tanto, una estimación de la misma:

$$Q_\pi(S_t,A_t) = Q_\pi(S_t,A_t) + \alpha\Big(\underbrace{R_{t+1} + \gamma \max_{a'\in\mathcal{A}} Q_\pi(S_{t+1},a')}_{\text{(a)}} - \underbrace{Q_\pi(S_t,A_t)}_{\text{(b)}}\Big) \tag{2.18}$$

donde α refleja el tamaño del paso de aprendizaje (o *learning rate*).

La ecuación (2.18) permite llevar a cabo una estimación recursiva (partiendo de un valor inicial aleatorio) de la función $Q_\pi(s,a)$. Así, a cada paso dado por el agente, la estimación de la función estado-acción se corrige con la diferencia que existe entre (a) la recompensa acumulada que el agente realmente obtiene al realizar la acción A_t y alcanzar un nuevo estado S_{t+1} y (b) la recompensa acumulada que el agente había predicho que conseguiría. Esta técnica, denominada *Time Difference*, hace que no sea necesario modelar el sistema, sino que basta con la propia experiencia para estudiar cómo se comporta.

Un detalle a destacar es que, a diferencia de los métodos basados en política, este tipo de métodos suelen llevar a cabo un aprendizaje de manera *off-policy*, esto es, utilizando para el aprendizaje cualquier conjunto de datos obtenido por el agente durante todo el entrenamiento y no únicamente la interacción inmediatamente anterior. Esto quiere decir que, una vez dado el paso por el agente, este puede utilizar cualquier conjunto de datos $(S_t, A_t, R_{t+1}, S_{t+1})$.

Por último, es necesario puntualizar que, en realidad, los algoritmos *Q-Learning* son simplemente una familia de métodos más dentro del abanico de los métodos basados en valor. Sin embargo, dada la popularidad que dichos métodos han adquirido en la literatura científica, a menudo se utiliza el término *Q-Learning* para englobar a este tipo de métodos que no utilizan la política π de forma explícita en su aprendizaje.

Policy Optimization.

Los métodos basados en política son métodos que, por el contrario, tratan de calcular directamente una aproximación de la política de actuación π mediante una función $\pi_\theta(a|s) = \pi(a|s;\theta)$ parametrizada según los parámetros θ. Para ello, el algoritmo calcula estos parámetros θ de tal forma que se maximice la recompensa obtenida por el agente en su interacción con el entorno.

De esta forma, se aplica el algoritmo de ascenso del gradiente[2] a θ para maximizar la función objetivo $J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[G_t(\tau)]$.

$$\theta_{k+1} = \theta_k + \alpha\, \nabla_\theta J(\pi_\theta)|_{\theta_k} \tag{2.19}$$

Además, al igual que los algoritmos basados en valor podían utilizar los datos obtenidos en cualquier interacción con el escenario a lo largo de su trayectoria, los algoritmos basados en política únicamente pueden utilizar aquellos datos de la interacción inmediatamente anterior para actualizar su política (*on-policy*). Esto se debe a que, atendiendo a la ecuación (2.19), la estimación del gradiente de la recompensa a recibir $\nabla_\theta J(\pi_\theta)|_{\theta_k}$ se evalúa en base a la política actual π_{θ_k}.

Se requiere, por tanto, de una expresión de $\nabla_\theta J(\pi_\theta)$ para llevar a cabo este tipo de algoritmos. Para ello, se expande la expresión de la función objetivo utilizando los resultados expuestos en (2.2) y (2.5):

$$\nabla_\theta J(\pi_\theta) = \nabla_\theta \mathbb{E}_{\pi_\theta}[G(\tau)] = \nabla_\theta \int_\tau \underbrace{\prod_{s,a\in\tau} P(s'|s,a)\,\pi_\theta(a|s)}_{P(\tau|\pi_\theta)}\, G(\tau) = \nabla_\theta \int_\tau P(\tau|\pi_\theta)\, G(\tau) \tag{2.20}$$

donde se ha simplificado la notación utilizando como punto de partida el instante $t = 0$. Pese a esta asunción, los resultados son fácilmente generalizables a un instante de partida t genérico.

A continuación, se hace uso del *log-derivative trick* para reemplazar la dependencia de la función objetivo con la distribución de probabilidad $P(\tau|\pi_\theta)$ y utilizar, en su lugar, el logaritmo de dicha función. Este artificio matemático se realiza debido a que la función logarítmica aporta una mayor facilidad en su operativa, permitiendo sustituir multiplicaciones y divisiones por sumas y restas. Así, la ecuación resulta:

$$\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \nabla_\theta \int_\tau P(\tau|\pi_\theta)\, G(\tau) \\
&= \int_\tau \nabla_\theta P(\tau|\pi_\theta)\, G(\tau) \\
&= \int_\tau \nabla_\theta P(\tau|\pi_\theta) \frac{P(\tau|\pi_\theta)}{P(\tau|\pi_\theta)}\, G(\tau) \\
&= \int_\tau P(\tau|\pi_\theta) \frac{\nabla_\theta P(\tau|\pi_\theta)}{P(\tau|\pi_\theta)}\, G(\tau) \\
&= \int_\tau P(\tau|\pi_\theta) \nabla_\theta \log\big(P(\tau|\pi_\theta)\big)\, G(\tau)
\end{aligned} \tag{2.21}$$

Además, es posible simplificar la expresión $\log\big(P(\tau|\pi_\theta)\big)$ de la siguiente forma:

$$\begin{aligned}
\nabla_\theta \log\big(P(\tau|\pi_\theta)\big) &= \nabla_\theta \log\big(\prod_{s,a\in\tau} P(s'|s,a)\,\pi_\theta(a|s)\big) \\
&= \sum_{s,a\in\tau} \big(\nabla_\theta \log\big(P(s'|s,a)\big) + \nabla_\theta \log\big(\pi_\theta(a|s)\big)\big) \\
&= \sum_{s,a\in\tau} \nabla_\theta \log\big(\pi_\theta(a|s)\big)
\end{aligned} \tag{2.22}$$

donde se ha utilizado que la función de transición de estado $P(s'|s,a)$ no depende del parámetro θ para simplificar la ecuación.

[2] El algoritmo de ascenso del gradiente es un algoritmo de optimización utilizado para calcular el máximo de una función.

Uniendo los resultados de las ecuaciones (2.21) y (2.22), es posible concluir en una nueva expresión de la función objetivo:

$$\nabla_\theta J(\pi_\theta) = \int_\tau P(\tau|\pi_\theta) \sum_{s,a\in\tau} \nabla_\theta \log\big(\pi_\theta(a|s)\big)\ G(\tau) = \mathbb{E}_{\pi_\theta}[\sum_{s,a\in\tau} \nabla_\theta \log\big(\pi_\theta(a|s)\big)\ G(\tau)] \tag{2.23}$$

Esta ecuación representa el gradiente de la función objetivo como la recompensa total acumulada al final de la trayectoria τ ponderada por la probabilidad (en forma logarítmica) de que todas las acciones a en τ resulten elegidas [52]. Esto, sin embargo, no parece lo más conveniente, ya que se está recompensando a las acciones posteriores a un instante t_k, $a_{t_{k^+}}$, con las recompensas derivadas de las acciones realizadas en instantes anteriores a este t_k, $a_{t_{k^-}}$ (puesto que $G(\tau)$ incluye las recompensas desde t_0 hasta t_N).

Se propone, como alternativa, una función objetivo donde, para cada acción a_t, únicamente se consideran las recompensas obtenidas como consecuencia de dicha acción. Esto es:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{T} \nabla_\theta \log\big(\pi_\theta(a_t|s_t)\big)\ G_t(\tau)] \tag{2.24}$$

donde T es la longitud de la trayectoria τ. De esta forma, las acciones solo son premiadas con las recompensas que se obtienen a raíz de su influencia, y no con las obtenidas anteriormente.

En la literatura, sin embargo, se suele hacer uso de una reformulación de (2.24) en función de la función valor estado-acción [49]. Para llegar a dicha expresión, basta con aplicar la ley de las esperanzas iteradas [83] y dividir la trayectoria τ en $\tau_{:t} = (s_0,a_0,s_1,a_1,...,s_t,a_t)$ y $\tau_{t:} = (s_{t+1},a_{t+1},s_{t+2},a_{t+2},...,s_T,a_T)$.

$$\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{T} \nabla_\theta \log\big(\pi_\theta(a_t|s_t)\big)\ G_t(\tau)] \\
&= \sum_{t=0}^{T} \mathbb{E}_{\tau\sim\pi_\theta}[\nabla_\theta \log\big(\pi_\theta(a_t|s_t)\big)\ G_t(\tau)] \\
&= \sum_{t=0}^{T} \mathbb{E}_{\tau_{:t}\sim\pi_\theta}\Big[\mathbb{E}_{\tau_{t:}\sim\pi_\theta}\big[\nabla_\theta \log\big(\pi_\theta(a_t|s_t)\big)\ G_t(\tau)\big|\ \tau_{:t}\big]\Big] \\
&= \sum_{t=0}^{T} \mathbb{E}_{\tau_{:t}\sim\pi_\theta}\Big[\nabla_\theta \log\big(\pi_\theta(a_t|s_t)\big)\ \mathbb{E}_{\tau_{t:}\sim\pi_\theta}[G_t(\tau)|\ \tau_{:t}\big]\Big]
\end{aligned} \tag{2.25}$$

donde $\mathbb{E}_{\tau_{t:}\sim\pi_\theta}[G_t(\tau)|\ \tau_{:t}]$ puede identificarse, según la ecuación (2.8), por $Q_{\pi_\theta}(s,a)$. Resulta, por tanto, la siguiente expresión:

$$\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \sum_{t=0}^{T} \mathbb{E}_{\tau_{:t}\sim\pi_\theta}\Big[\nabla_\theta \log\big(\pi_\theta(a_t|s_t)\big)\ Q_{\pi_\theta}(s_t,a_t)\Big] \\
&= \mathbb{E}_{\tau_{:t}\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log\big(\pi_\theta(a_t|s_t)\big)\ Q_{\pi_\theta}(s_t,a_t)\right]
\end{aligned} \tag{2.26}$$

Con esto, y sabiendo que la función $Q_{\pi_\theta}(s_t,a_t)$ representa la recompensa acumulada esperable a partir de un instante t cualquiera, se concluye en:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{s,a\in\tau} \nabla_\theta \log\big(\pi_\theta(a|s)\big)\ Q_{\pi_\theta}(s,a)\right] \tag{2.27}$$

A este resultado se le conoce en la literatura como *Policy Gradient Theorem* y permite hallar el gradiente de la función objetivo de una forma cómoda para la aplicación de la ecuación (2.19) y, por tanto, para el cálculo de una política de actuación π_θ que aproxime a la política óptima π^*.

2.3 Deep Reinforcement Learning.

Las técnicas de *Reinforcement Learning* más tradicionales se fundamentan, en el caso de los métodos basados en valor, en estrategias tabulares en las que una tabla almacena un valor estado-acción por cada par (s,a) alcanzado por el agente o, en el caso de aquellos métodos basados en política, en el modelado de la política π a partir de una función parametrizada según θ.

Sin embargo, estas herramientas, a menudo, resultan ineficientes a la hora de abordar problemas de mayor complejidad donde las dimensiones del espacio de acción y espacio de estado son más altas. Surge la necesidad, por tanto, de buscar aproximadores que permitan modelar las dinámicas no lineales tanto del escenario como del agente de aprendizaje. Las redes neuronales, o el *Deep Learning*, se sitúan como una herramienta transversal dentro del marco del *Machine Learning* ideal para este tipo de problemas.

2.3.1 Introducción al Deep Learning.

Deep Learning es el nombre asignado a una familia de algoritmos de aprendizaje automático cuyo objetivo es el de modelar, mediante una serie de arquitecturas computacionales más complejas y flexibles denominadas redes neuronales, las posibles abstracciones que pudieran derivarse de un determinado conjunto de datos. Estas redes neuronales, constituidas a partir de la ordenación en capas de su elemento computacional más básico (la neurona), buscan emular el funcionamiento de un cerebro humano. Así, a partir de unos estímulos de entrada, el entramado de neuronas que conforman la red ha de ser capaz de interactuar entre sí para generar una conclusión como su señal de salida.

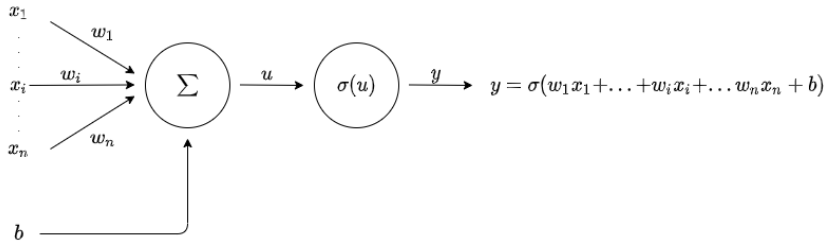


Figura 2.5 Esquema básico de una neurona.

En la Figura 2.5 se pueden identificar los principales componentes de esta unidad básica o neurona:

- **Entradas**, x_i. Las entradas de una neurona son los estímulos ante los cuales la neurona genera una respuesta. Estas entradas serían el equivalente a la información recibida por una neurona real a través de sus dendritas.
- **Pesos**, w_i. Estos pesos permiten ponderar la información recibida como entrada, dando una mayor importancia a ciertos estímulos frente a otros.
- **Sesgo o *bias***, b. El sesgo, o *bias*, es un elemento adicional que permite a la neurona estar más dispuesta o menos a disparar la función de activación y, con ello, favorecer el aprendizaje. Así, si el sesgo de una neurona toma un valor alto, esta neurona será más propensa a disparar la función de activación.
- **Función de activación**, $\sigma(\cdot)$. Por último, la función de activación corresponde a una función no lineal que permite reaccionar ante los distintos estímulos de entrada ya ponderados. Esta función es la encargada de generar una salida de la neurona y, por tanto, de suplir la función realizada por el *axón*.

$$y = \sigma\left(w_1x_1 + \ldots + w_ix_i + \ldots + w_nx_n + b\right) \tag{2.28}$$

Se identifican, en la literatura, numerosas funciones de activación [57]. Sin embargo, la más popular de estas funciones corresponde a la función de activación *Rectified Linear Unit (ReLU)*: $\sigma(u) = \max(0,u)$ [29].

Es posible identificar, en función de cómo se agrupan las diferentes capas existentes en una red neuronal, diferentes estructuras. Sin embargo, existe una que predomina en el ámbito del DRL: las redes neuronales

densas o *fully-connected*. Estas redes están constituidas por una capa de entrada (*input layer*) que recibe la información a procesar, una o varias *hidden layers* que procesan dicha información y una última capa de salida (*output layer*) que genera las salidas de la red. De esta forma, la agrupación en diferentes capas de la suficiente cantidad de estas neuronas, o perceptrones[3], permite aproximar cualquier función continua con la precisión deseada [20, 70].

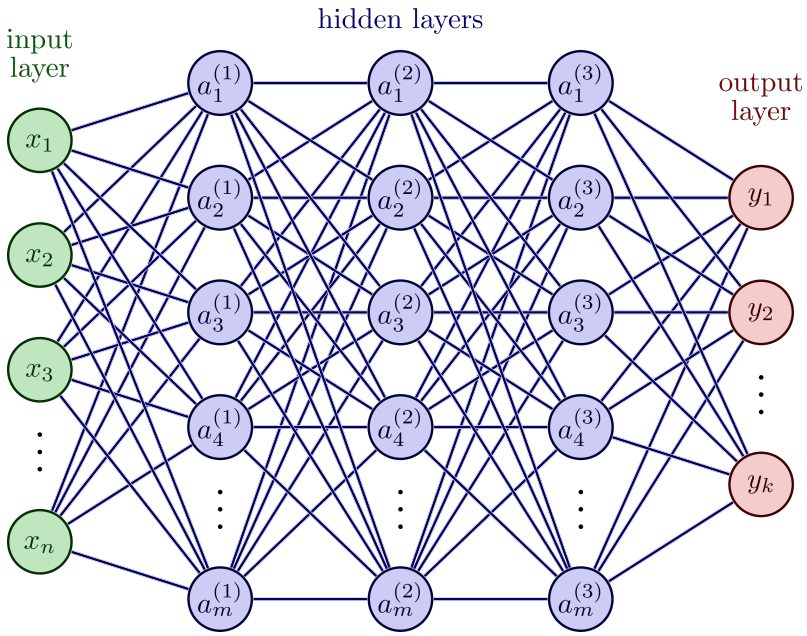


Figura 2.6 Estructura típica de una red neuronal densa.

Una vez definida la estructura de una red neuronal, es necesario someter a dicha red a un proceso de entrenamiento mediante el cual las neuronas se adapten a las dinámicas del sistema. Durante este proceso, los distintos pesos y sesgos de cada una de las neuronas de la red se ajustan de manera iterativa para mejorar el desempeño de la red al tratar de predecir una salida. Existen dos etapas fundamentales en este proceso de entrenamiento:

- ***Forward propagation***. El *forwarding* o propagación hacia delante consiste en el cálculo del vector de salidas $(y_1,...,y_k)$ a partir del vector de estímulos $(x_1,...,x_n)$ en su paso a través de las diferentes neuronas.
- ***Backward propagation***. La propagación hacia detrás, por su parte, es la encargada de realizar el aprendizaje tanto de los pesos como de los sesgos de las neuronas. Así, en esta segunda etapa, se aplica el método del gradiente (ya sea ascendente o descendente) para mejorar el desempeño de la red acorde a la función objetivo J.

$$\theta_{k+1} = \theta_k \pm \alpha \frac{\partial J}{\partial \theta} \tag{2.29}$$

donde α es el *learning rate* y, por tanto, define la velocidad de aprendizaje del algoritmo.

Este proceso de actualización de parámetros, llevado a cabo de forma iterativa desde la capa de salida hasta la de entrada, le permite a la red entender la sensibilidad de la función objetivo a cada uno de los parámetros de las neuronas y moldearse en consecuencia.

Finalmente, una vez entrenada la red neuronal, el desempeño conseguido por este tipo de técnicas es tal que se han convertido en la herramienta estándar para el procesado de grandes cantidades de datos, identificación de patrones y modelado de sistemas.

[3] En el ámbito del *Deep Learning*, usualmente se denomina perceptrón a la unidad básica funcional de una red neuronal. Esta unidad básica, propuesta en [44] en 1943, actúa de discriminador lineal, permitiendo clasificar un conjunto de datos de entrada en diferentes clases.

2.3.2 Aplicación al Reinforcement Learning.

Aunque el interés por el *Deep Learning* y las redes neuronales despierta a mediados de la década de 1980, no es hasta 1995 cuando se lleva a cabo la primera integración de esta herramienta en el *Reinforcement Learning* [74]. Se descubriría, en aquel entonces, el potencial que tiene este tipo de técnicas para modelar funciones no lineales de alta complejidad como las funciones de valor estado o estado-acción.

De esta forma, el enfoque adoptado por el *Deep Reinforcement Learning* es el de utilizar redes neuronales, típicamente redes densamente conectadas, bien como aproximadores para las funciones $V_{\pi^*}(s)$ y $Q_{\pi^*}(s,a)$, o bien para la política de actuación óptima del agente $\pi^*(a|s)$.

$$\begin{cases} Q_\theta(s,a) \to Q_{\pi^*}(s,a) \\ V_\theta(s) \to V_{\pi^*}(s) \end{cases} \qquad \pi_\theta(a|s) \to \pi^*(a|s) \tag{2.30}$$

No obstante, introducir estas redes en el algoritmo de aprendizaje conlleva realizar ciertas modificaciones sobre el mismo. Así, en el caso de los métodos *Q-Learning* se hace necesario definir una función objetivo que indique a la red cuál es la dirección de aprendizaje acertada. Se utiliza, para ello, una función de pérdidas que refleje la desviación de la recompensa acumulada obtenida con la esperada. Un ejemplo de esta función sería:

$$\mathcal{L}(\theta) = \left(r + \gamma \max_{a'} Q_{\theta^-}(s',a') - Q_\theta(s,a)\right)^2 \tag{2.31}$$

Por su parte, los métodos basados en política permiten una adaptación más sencilla del algoritmo puesto que ya trabajan con aproximadores. En este tipo de métodos, donde ya se define una función objetivo que busca maximizar la función $G_t(\tau)$, basta con optimizar la red a partir de dicha función de objetivo.

Existen multitud de algoritmos que combinan las herramientas proporcionadas por el *Deep Learning* con las ideas del aprendizaje reforzado. Entre los métodos basados en valor, son destacables, entre otros, el algoritmo DQN [46] (algoritmo responsable de la popularidad del DRL) y sus extensiones: Dueling DQN [80] y Double DQN [77]. Por su parte, entre los algoritmos basados en política destacan algunos como los algoritmos A3C [45] o PPO [62], que, en lugar de maximizar directamente la recompensa acumulada al final del episodio, maximiza la variación de dicha recompensa con cada actualización de los parámetros. Finalmente, y entre los algoritmos más populares del DRL se encontrarían aquellos que utilizan ideas de ambos grupos: DDPG [41] o SAC [28].

Ha sido gracias a este tipo de técnicas que el *Reinforcement Learning* ha acumulado tanto éxito durante la última década. El empleo de redes neuronales multicapas como funciones no lineales han permitido resolver todo tipo de problemas, desde juegos de mesa hasta el control de procesos industriales [65, 71].

2.3.3 Consideraciones prácticas en la implementación.

La implementación de un algoritmo de DRL puede resultar, en ocasiones, muy compleja. Además, el proceso de entrenamiento de un agente basado en este tipo de estrategias de aprendizaje por refuerzo suele ser algo lento y tedioso, resultando de gran ayuda tener en consideración ciertos aspectos de antemano para agilizar la tarea. A continuación, se comentan algunas herramientas útiles para acelerar dicho proceso de aprendizaje.

Buffer de experiencias.

El *buffer* de experiencias es una herramienta muy utilizada en aquellos algoritmos *off-policy* que lo permiten. Este *buffer* consiste en un histórico de datos donde el agente, en su proceso de aprendizaje, relaciona todas aquellas interacciones que lleva a cabo con el escenario. Se define, por tanto, una memoria intermedia de tamaño N_{buff} donde el algoritmo almacena las transiciones $e_t = (S_t = s, A_t = a, R_{t+1} = r, S_{t+1} = s')$. En la literatura, a estas transiciones se les denominan experiencias.

La finalidad de este *buffer* es triple. Por un lado, al existir un histórico de las últimas transiciones realizadas por el agente, se le otorga la capacidad a este de aprender mediante el uso de *mini-batches*. Este enfoque, a diferencia del aprendizaje *online*, permite hacer un uso más eficiente de los recursos hardware disponibles aprovechando, así, cada una de las pasadas hacia detrás de la red para aprender de varios conjuntos de datos

simultáneamente. Por otro lado, mediante un muestreo aleatorio del *buffer* se eliminan las correlaciones que pudieran existir entre las distintas muestras. Este aspecto es especialmente interesante en sistemas dinámicos con evolución lenta, ya que la correlación entre muestras pudiera repercutir en una inestabilidad del proceso de aprendizaje. Por último, esta herramienta puede llegar a resultar de vital importancia cuando el agente debe interactuar con un sistema físico. En estos casos, la recopilación de experiencias suele ser un proceso más costoso en recursos y en tiempo, por lo que, para solventar esto, contar con una herramienta que permite reutilizar las experiencias pasadas puede acelerar enormemente el proceso de aprendizaje. Además, dada la lenta convergencia de este tipo de métodos, realizar estas múltiples pasadas a un mismo conjunto de datos también puede beneficiar al agente en el aprendizaje de dinámicas más complejas del sistema.

Dilema de exploración-explotación.

El dilema de exploración-explotación es una problemática frecuentemente debatida en los distintos tipos de algoritmos de optimización. Este dilema, que tiene su origen en el desconocimiento del escenario involucrado, hace referencia a la tesitura que existe entre si resulta más conveniente centrar los esfuerzos del agente en explorar todo el escenario o si, por el contrario, favorece más a su objetivo centrarse en explotar aquellos puntos que ya se conocen como favorables. No existe una respuesta universal a este dilema. En la práctica, no obstante, se ha demostrado que es necesaria la combinación de ambos factores para llegar a buenos resultados. De esta forma, mientras que en la exploración se toman algunos riesgos para recopilar información del escenario, la explotación se ocupa de sacar ventaja de aquellos aspectos ya conocidos.

La forma de abordar este problema, sin embargo, varía en función del tipo de algoritmo utilizado. Así, mientras que en los algoritmos *Q-Learning* se hace uso de la estrategia de exploración conocida por ε-*greedy* (donde un factor ε define la probabilidad de llevar a cabo bien una acción aleatoria, o bien aquella indicada por la política π), los algoritmos basados en política y con un espacio de acción continuo utilizan una función de ruido para permitir una mayor exploración al agente.

El proceso de Ornstein-Uhlenbeck [85] se ha impuesto en la comunidad de RL como uno de los métodos favoritos para desempeñar esta función de exploración en este último tipo de problemas. Este proceso, descrito como un proceso estacionario de Gauss-Markov[4], permite generar un muestreo con correlación temporal idóneo para sistemas físicos con inercia. La ecuación que describe este proceso es la siguiente:

$$x_{t+1} = x_t + \Theta\,(\mu - x_t)\,dt + \sigma\sqrt{dt}\,\mathcal{N}(0,1) \tag{2.32}$$

donde Θ es un parámetro que permite cuantificar el tamaño del paso de la variable x_t, μ hace referencia a la media de la distribución de probabilidad y σ a la desviación típica de la misma. Por su parte, dt se refiere al salto temporal entre los instante t y $t+1$.

Normalización de los datos de entrada.

El aprendizaje de las redes neuronales depende, en gran medida, de la calidad de los datos de entrada. Así, en [7] se demuestra que la normalización de los datos utilizados como entrada por cada una de las capas de la red puede llegar a acelerar el proceso de aprendizaje. Esta normalización tendría lugar de la siguiente forma:

$$a_k^l = \frac{a_k^l - \mu^l}{\sigma^l} \qquad \forall k = 1,...,H \tag{2.33}$$

donde H hace referencia al número total de neuronas en cada capa oculta y μ^l y σ^l corresponden a la media y desviación típica de la distribución de los datos de salida a_k^l desde la capa l. Estos parámetros, compartidos por una misma capa l y diferentes para cada paso de entrenamiento, se calculan tal que:

$$\begin{cases} \mu^l = \dfrac{1}{H}\displaystyle\sum_{k=1}^{H} a_k^l \\ \sigma^l = \sqrt{\dfrac{1}{H}\displaystyle\sum_{k=1}^{H}\left(a_k^l - \mu^l\right)^2} \end{cases} \tag{2.34}$$

[4] Un proceso de Gauss-Markov es aquel que describe un modelo estocástico en el que la probabilidad de ocurrencia de un evento depende únicamente del evento acontecido anteriormente y sigue una distribución normal multivariable.

Saturación de gradientes.

En el DL, uno de los problemas más acusados es la aparición de *exploding gradients*. Este fenómeno, producido cuando los valores acumulados por el gradiente de la función de pérdidas (o, en caso de tratarse de algoritmos basados en política, la función de recompensa acumulada esperada) son demasiado grandes, origina una rápida inestabilidad en la red neuronal y la consecuente divergencia del método.

La solución a este problema consiste en "recortar" los valores que toman estos gradientes cuando la norma de dicho vector supera cierto umbral. Para ello, se realiza la siguiente operación:

$$\nabla J = \frac{\eta}{\|\nabla J\|} \nabla J \tag{2.35}$$

donde η corresponde al umbral máximo permitido para la norma del vector gradiente. Este hiperparámetro, sin embargo, resulta muy sensible tanto a la función de pérdidas utilizada para entrenar la red neuronal como a la arquitectura de la propia red. Resulta interesante, por ello, la solución propuesta en [63], donde se propone un cálculo de este umbral de forma adaptativa utilizando, para ello, un histórico de los valores tomados por la norma del gradiente durante todo el entrenamiento y estableciendo como umbral el percentil p.

2.4 Analogía con el control predictivo.

La estrategia de control utilizada como referencia en este trabajo es, como ya se ha comentado en el Capítulo 1, el control predictivo o MPC. Por consiguiente, resulta interesante analizar brevemente la estructura básica de este tipo de controladores, así como apuntar aquellas analogías y diferencias encontradas con respecto al enfoque propuesto por el aprendizaje reforzado.

2.4.1 Introducción al control predictivo.

El MPC [13] agrupa, en realidad, a un conjunto de técnicas de control avanzado con unos mismos principios. Estas estrategias de control se centran en la idea de optimizar una función de coste (u objetivo) a partir del conocimiento de un modelo del sistema a controlar y, por tanto, de sus dinámicas. Para ello, el controlador tiene la capacidad de analizar la evolución futura del sistema en función de la acción de control utilizada para influir sobre él.

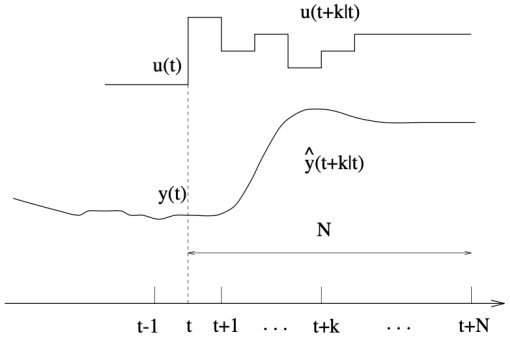


Figura 2.7 Estrategia de control MPC [13].

La estrategia utilizada por este tipo de técnicas consiste en, dado un instante temporal t, decidir la secuencia de acción $\left[u(t|t),...,u(t+N_u-1|t)\right]$ que permita optimizar la función objetivo definida. Para lograr su objetivo, dicho controlador dispone, además, de una venta temporal de N_2-N_1 instantes temporales a futuro en la que es capaz de predecir la respuesta del sistema a dichas acciones de control: $\left[\hat{y}(t+N_1|t),...,\hat{y}(t+N_2|t)\right]$. De esta forma, el agente de control debe decidir qué secuencia de acciones es la que resulta más favorable para la función objetivo J. Finalmente, y una vez decidida dicha secuencia de acción, se implementa la primera acción de control de dicha secuencia y se vuelve a repetir el proceso.

No obstante, un detalle importante a tener en cuenta en este tipo de estrategias es que la acción de control óptima no es aquella que resulta más favorable inmediatamente, sino aquella que favorece en mayor medida a la función objetivo al largo plazo. Para clarificar este punto, se presenta, de forma general, la expresión de dicha función:

$$J(N_1,N_2,N_u) = \sum_{j=N_1}^{N_2} \delta(j)\left[\hat{y}(t+j|t) - w(t+j)\right]^2 + \sum_{j=1}^{N_u} \lambda(j)\left[\Delta u(t+j-1)\right]^2 \tag{2.36}$$

En la ecuación (2.36) se observa la influencia de tres parámetros: N_1, N_2 y N_u. Estos parámetros permiten definir, por un lado, el tamaño de la ventana temporal en la que el controlador es capaz de predecir la salida del sistema $\hat{y}(t+j|t)$ (N_1 y N_2) y, por otro, el tamaño de la secuencia de acciones de control (N_u). Estos tamaños, sin embargo, pueden no coincidir, en cuyo caso debe cumplirse que $N_2 - N_1 > N_u$. Por su parte, los parámetros $\delta(j)$ y $\lambda(j)$ permiten marcar la importancia de cada uno de los términos de la función objetivo, consiguiendo, con ello, un respuesta más suave o agresiva del sistema.

Esta capacidad de predecir el comportamiento de un sistema en función de la acción de control tomada ha permitido al control MPC situarse en una posición privilegiada dentro de las estrategias de control moderno. De hecho, diversos autores han tratado de aplicar este tipo de técnicas al control de plantas termosolares. Un ejemplo de esto es [43], donde se propone una estrategia de control MPC basada en mecanismos de mercado por los cuales diferentes lazos de colectores se coordinan entre sí para maximizar la producción energética de la instalación, o [25], el cual desarrolla un sistema de control distribuido basado en MPC que permita reducir el tiempo de ejecución del algoritmo centrándose en el control individual de cada uno de estos lazos.

2.4.2 Comparación entre MPC y DRL.

Tanto el control predictivo basado en modelo, MPC, como el aprendizaje por refuerzo, DRL, surgen como soluciones para afrontar un mismo problema: el control óptimo de un sistema. No obstante, a pesar de compartir este mismo objetivo, el enfoque propuesto por cada una de estas técnicas es muy diferente. Así, por un lado, el control predictivo sigue las directrices de la teoría de control y propone hacer uso del conocimiento disponible del sistema para diseñar una ley de control óptima conforme a la función objetivo establecida. Por contra, aquellas estrategias de aprendizaje reforzado abordadas en este trabajo no conocen un modelo del sistema en cuestión, sino que utilizan la información recopilada de la interacción con su entorno para desarrollar su política de acción. Estas diferencias han ocasionado que, hasta la fecha, ambas disciplinas hayan seguido desarrollos muy diferentes entre sí.

Sin embargo, aún con estas diferencias, es posible encontrar ciertas analogías entre ambas perspectivas [6], por ejemplo, en la terminología. Así, mientras que en DRL se habla de un agente que interactúa con su entorno (o escenario), en el ámbito del control predictivo este agente se conoce como controlador y el escenario como planta o sistema. Por su parte, la interacción que se lleva a cabo con dicho entorno (o, en el caso del control MPC, planta) se realiza mediante una acción a que, en el caso del control predictivo, se conoce como acción de control u_k.

Además de estas equivalencias directas entre términos, existen otras similitudes. Se identifica, en ambos métodos, una función objetivo que optimizar. Sin embargo, mientras que en el caso del MPC esta función objetivo se trata de una función de coste J_k a minimizar, en el caso del DRL se busca maximizar la recompensa total recibida por el agente $G_t(\tau)$. Es posible, aún así, identificar una equivalencia entre ambas funciones objetivo:

$$\sum_k J_k \sim -G_t(\tau) \tag{2.37}$$

Por último, donde mayor diferencia se puede encontrar entre ambas terminologías es en la definición del concepto de estado. Así, mientras que en el control predictivo se entiende por estado aquella descripción de las propiedades internas de un sistema, en el caso del aprendizaje por refuerzo el estado atiende a una descripción tanto de las variables internas como externas de este donde, como en el caso abordado por este trabajo, se incluyen las predicciones a futuro de las diferentes perturbaciones que afectan al sistema. Asimismo, también conviene destacar las diferencias encontradas al tratar con estados parcialmente observables. De esta forma, mientras que las técnicas de control predictivo utilizan observadores para estimar los valores de aquellos estados ocultos para el controlador, en el caso de los agentes DRL se incorpora en la descripción del estado un histórico de las variables más relevantes del sistema para que sea el algoritmo el que los deduzca.

Tabla 2.1 Resumen de diferencias entre algoritmos DRL y MPC.

	DRL	MPC
Enfoque	Acción de control determinada a partir de la experiencia	Acción de control determinada a partir del conocimiento del sistema
Terminología	Agente / Escenario o entorno / Acción	Controlador / Sistema o planta / Acción de control
Función objetivo	Maximizar recompensa total: $G_t(\tau)$	Minimizar función de coste: J_k
Estado	Variables internas y externas Históricos de datos	Variables externas Observadores

Estas diferencias existentes entre ambos métodos han permitido que cada uno desarrolle una serie de fortalezas y debilidades frente al otro. Así, dada la necesidad del MPC de un modelo del sistema a controlar, este se convierte en un método muy sensible a incertidumbres en el modelo. El RL, por el contrario, al llevar a cabo un aprendizaje a partir de su experiencia con el entorno, permite al agente de control superar esta barrera de la incertidumbre y le otorga mayor flexibilidad. De hecho, son varios los estudios que comentan la adaptabilidad de los algoritmos de aprendizaje reforzado a cambios en los parámetros del entorno [69].

Por otro lado, gracias al conocimiento que el controlador MPC posee de las dinámicas del sistema, este es capaz de adelantarse al comportamiento del mismo y, con ello, establecer restricciones rígidas en cualquiera de las variables controladas. Los métodos DRL, por el contrario, no pueden hacer frente a restricciones en el sistema más allá de mediante penalizaciones en la función de recompensa (restricciones blandas), siendo este uno de los principales inconvenientes de este tipo de algoritmos. No obstante, dado que, en la práctica, el tiempo de ejecución necesario para resolver problemas de optimización no lineales con restricciones rígidas es demasiado alto, los controladores MPC tienden a utilizar estas penalizaciones blandas en la función de coste para incluir las restricciones, equiparándose, con ello, a las técnicas de DRL.

La complejidad del sistema a controlar también es un factor decisivo en la elección del controlador más conveniente para un problema de control. Así, el control predictivo, al requerir de un modelo detallado de todas las dinámicas del sistema, puede llegar a resultar poco adecuado para problemas con sistemas muy complejos. Las técnicas de DRL, por el contrario, trabajan con modelos aproximados y, por tanto, con problemas de optimización más sencillos. Además, al poder realizar el proceso de entrenamiento de manera *offline*, permite una operación mucho más rápida del algoritmo. Este detalle resulta de vital importancia, ya que, en caso de que un control MPC contara con un modelo suficientemente bueno de la planta y pudiera resolver el problema de optimización rápidamente, los métodos DRL difícilmente pudieran mejorar su desempeño. Es esta velocidad de ejecución la que ofrece una ventaja competitiva al DRL frente al MPC.

Por último, resulta interesante destacar las diferencias existentes en los horizontes de predicción de ambos métodos. Los agentes de control basados en DRL, a diferencia del horizonte de predicción de tamaño $N_2 - N_1$ de los controladores MPC, cuentan con un horizonte de predicción infinito gracias a que su función objetivo contempla todas las recompensas acumuladas por el agente en su trayectoria. Sin embargo, este horizonte infinito debe matizarse ya que el factor de olvido γ limita el interés de aquellas recompensas más lejanas en el tiempo, reduciendo, así, el horizonte de predicción (o al menos la fuerza de este). Además, debido a la falta de correlación temporal entre los datos suministrados al agente a través del estado del escenario, el uso que el agente DRL hace de estas predicciones futuras es menos eficiente que en el caso del MPC.

A la vista de estas ventajas y desventajas de cada uno de los métodos, varios autores han propuesto hacer uso de ambas técnicas para aprovechar las virtudes de ambos campos, combinando, así, la flexibilidad, velocidad y escalabilidad propias del *Reinforcement Learning* con la estabilidad característica de los controladores predictivos. Así, en [6] se desarrolla un algoritmo que, a cada instante de control, combina un problema de optimización de horizonte de predicción unidad con una estimación basada en valor de las recompensas futuras obtenibles por el agente para desarrollar una política óptima de actuación. Por último, son varios los estudios que proponen utilizar controladores MPC como medio para obtener la política de actuación óptima y, por otro lado, los agentes DRL para mejorar el desempeño de los mismos mediante un ajuste de los parámetros de control en bucle cerrado [22, 88, 89].

3 Modelo del sistema.

Remember that all models are wrong; the practical question is how wrong do they have to be not to be useful.

GEORGE E. P. BOX

En este capítulo se presenta un modelo matemático de la planta solar térmica utilizada como caso de estudio en este trabajo, así como los detalles de implementación necesarios para una posterior integración del mismo con los diferentes agentes de control en un entorno de simulación.

3.1 Campo de colectores solares ACUREX.

La planta solar ACUREX consiste en una de las primeras plantas experimentales de tecnología PTC del mundo [40]. Esta planta, con un un pico de producción de potencia de 1.2 MW, fue desarrollada en la Plataforma Solar de Almería (PSA) por el Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT) y ha servido, durante algo más de 30 años, como banco de pruebas para numerosos ensayos de diferentes algoritmos y estrategias de control avanzado [11, 12].

(a) Vista general de la instalación [87].

(b) Detalle de colectores solares [8].

Figura 3.1 Fotografías del campo de colectores solares ACUREX.

La tecnología de concentración PTC consta de una serie de lazos de colectores solares cilindro-parabólicos que dan forma a un sistema de colectores distribuidos, o *Distributed Collector System* (DCS), que operan de tal manera que permiten redirigir toda la radiación solar directa incidente sobre el espejo de los colectores hacia una tubería receptora situada en el eje focal de estos. De esta forma, un fluido caloportador -en inglés, *Heat Transfer Fluid* (HTF)- que circula a través de la tubería es calentado y conducido o bien hasta un intercambiador de calor (en caso de que el fluido utilizado se tratase de aceite o sales fundidas), o bien hasta un tanque *flash* (en caso de agua presurizada) para la producción de vapor.

3.1.1 Descripción física del sistema.

El campo solar ACUREX [8], mostrado en la Figura 3.1, cuenta con un total de 480 seguidores solares PTC de eje único, orientados en dirección este-oeste, y una apertura total de colector de 2672 m^2. Estos colectores, a su vez, se encuentran distribuidos a lo largo de 10 lazos paralelos, cada uno de ellos formado por 4 módulos de 12 colectores dispuestos en serie. Los distintos lazos, todos iguales entre sí, tienen una longitud total de 172 *m*, donde 142 *m* pertenecen a las partes activas del lazo (es decir, aquellas que reciben la radiación solar), y 30 *m* a partes pasivas como uniones o juntas (las que no reciben radiación solar).

El fluido caloportador, o HTF, utilizado en la instalación corresponde al Therminol® 55, un aceite térmico extensamente utilizado en plantas termosolares gracias a las altas temperaturas que soporta. Este HTF cuenta con un amplio rango de funcionamiento que abarca desde los -25°C hasta los 290°C y puede soportar temperaturas de hasta 315°C sin llegar a degradarse [26]. Este fluido, una vez calentado, es utilizado para la producción de vapor de agua y, con ello, alimentar bien una turbina de generación eléctrica, o bien un intercambiador de calor de la planta desalinizadora anexa al campo solar.

El sistema cuenta, además, con un tanque de almacenamiento de aceite de 140 m^3 de capacidad que dota de flexibilidad a la instalación, permitiendo, así, amortiguar la variabilidad propia de la radiación solar y facilitar el arranque en frío de la planta. De esta forma, dado un campo solar con temperaturas de entrada de 210°C y 290°C de salida, este tanque podría alojar hasta 2.3 MWh de energía por varios días.

Por último, el accionamiento del sistema se da a través de una bomba de HTF que bombea el aceite desde el fondo del tanque de almacenamiento de aceite hasta los distintos colectores de la instalación. No obstante, dicha bomba cuenta con unos límites de operación establecidos entre 2 l/s y 12 l/s que garantizan la seguridad de la instalación y reducen el riesgo de que el aceite alcance temperaturas demasiado altas.

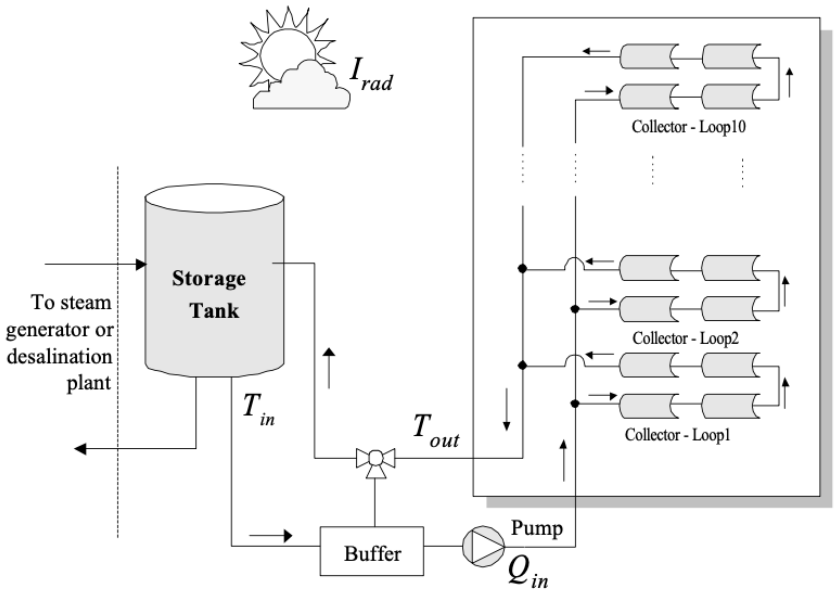


Figura 3.2 Diagrama funcional de la instalación [16].

En la Figura 3.2 se muestra un esquema de la planta completa. No obstante, dado que el objetivo de este trabajo no es más que proporcionar un primer acercamiento al control de plantas termosolares mediante técnicas de aprendizaje por refuerzo, se opta por una versión más simplificada de dicho modelo en la que únicamente se considera un único lazo de colectores cuyo problema de orientación ya ha sido resuelto.

3.1.2 Modelo de parámetros concentrados.

En la literatura, es posible encontrar diferentes descripciones del campo solar ACUREX. Así, en [14] se proporcionan dos descripciones de la planta: un modelo de parámetros concentrados y un modelo de parámetros distribuidos. En este caso, y con vistas a facilitar el posterior aprendizaje del agente de control, se opta por considerar el modelo de parámetros concentrados.

El modelo de parámetros concentrados describe la planta solar (o, en este caso, un lazo de la planta solar) como si de un único colector se tratase. Se establece, por tanto, una descripción del sistema en la que se discretizan los diferentes componentes distribuidos espacialmente a lo largo de la instalación en unos elementos más sencillos que permiten aproximar su comportamiento y se ignora, con ello, ciertos factores como la distribución espacial de las temperaturas a lo largo del lazo. Bajo esta descripción, la ecuación que define la evolución de la energía interna del fluido es la siguiente:

$$C\frac{dT_{out}}{dt} = K_{opt}\eta_o S I_{rad} - P_{cp}(T_{out} - T_{in})\,\frac{q}{1000\frac{l}{m^3}} - H_l S(T_{med} - T_{amb}) \tag{3.1}$$

donde se identifican las siguientes variables:

- Variables del lazo de control.
  - Variable controlada:
    - Temperatura del aceite a la salida del lazo de colectores, T_{out} [°C].
  - Acción de control:
    - Caudal de aceite a través del lazo, q [l/s].
  - Perturbaciones:
    - Irradiancia solar normal directa recibida, I_{rad} [W/m^2].
    - Temperatura del aceite a la entrada del lazo de colectores, T_{in} [°C].
    - Temperatura ambiente, T_{amb} [°C].
    - Eficiencia geométrica de la posición de los espejos con respecto al vector de radiación solar, η_o [p.u.].

- Variables dependientes del sistema.
  - Capacidad térmica del lazo, C [J/°C].
  - Propiedades del fluido, P_{cp} [J/m^3 °C].
  - Coeficiente global de pérdidas térmicas, H_l [W/m^2 °C].
  - Temperatura media del aceite, entre entrada y salida, T_{med} [°C].

- Parámetros físicos del sistema.
  - Eficiencia óptica del colector, K_{opt} [p.u.].
  - Superficie total del colector, S [m ^{2}].

No obstante, aunque en la literatura se ha tendido a aproximar algunos de estos parámetros y variables por valores constantes utilizando la información extraída a partir de diferentes ensayos [14], en este trabajo se analiza algo más en detalle la procedencia de los mismos para dotar de mayor realismo al modelo.

Temperatura media del aceite.

Esta temperatura hace referencia a la temperatura media del aceite en su paso a través del lazo de colectores. Esto es, su temperatura media entre la temperatura a la entrada del lazo, T_{in}, y la temperatura a la salida, T_{out}.

$$T_{med} = \frac{T_{in} + T_{out}}{2} \tag{3.2}$$

Coeficiente global de pérdidas térmicas.

El coeficiente global de pérdidas refleja las pérdidas térmicas sufridas durante el paso del aceite a través del colector. La expresión de dicho coeficiente viene recogida en [15] como:

$$H_l = 0.00249\,(T_{med} - T_{amb}) - 0.06133 \tag{3.3}$$

Propiedades del fluido.

El término P_{cp} es un parámetro referido a las propiedades del fluido. En este caso, la definición de P_{cp} viene dada por:

$$P_{cp} = c_{p_f}\rho_f \tag{3.4}$$

donde c_{p_f} [J/kg °C] y ρ_f [kg/m^3] corresponden, respectivamente, a la capacidad térmica específica y densidad del fluido.

Atendiendo a los datos suministrados por el fabricante del aceite acerca del Therminol® 55 [15], ambos parámetros pueden obtenerse según las siguientes expresiones:

$$\begin{cases} c_{p_f} = 1820 + 3.478T_f \\ \rho_f = 903 - 0.672T_f \end{cases} \tag{3.5}$$

Eficiencia óptica del colector.

La eficiencia óptica del colector permite tener en cuenta las pérdidas energéticas producidas en el proceso de reflexión y absorción, por parte de la tubería receptora, de la radiación solar. Estas pérdidas pueden deberse a diversos factores [8]:

- Reflectividad de los espejos del colector, r.
- Transmisividad de la cubierta de vidrio que recubre la tubería receptora, $\tau_{sol,c}$.
- Absortancia de la tubería, $\alpha_{sol,A}$.
- Factor de interceptación, γ_A. Este factor refleja los errores en posicionamiento del eje de la parábola formada por los espejos o errores en la propia forma de los espejos, entre otros posibles defectos.

Aunque estos parámetros presentan una dependencia con el ángulo de incidencia de la radiación solar, en este trabajo se asumirán constantes y con un valor igual a los valores medios obtenidos empíricamente en [19]. Con esto, la definición de la eficiencia óptica del colector queda:

$$K_{opt} = r\,\tau_{sol,c}\,\alpha_{sol,A}\,\gamma_A \tag{3.6}$$

Asimismo, se ha ignorado en este cálculo la influencia que pudiera tener otros factores como la suciedad o el envejecimiento de los materiales [19] debido a la complejidad que ello presentaría en el modelo.

Capacidad térmica del lazo de colectores.

La capacidad térmica del lazo de colectores corresponde, en realidad, a la capacidad térmica del fluido contenido a lo largo de todo el lazo. Se define, por tanto:

$$C = L_{loop}A_f\rho_f c_{p_f} \tag{3.7}$$

donde:

- L_{loop}: Longitud del lazo de colectores, en m.
- A_f: Área de paso del fluido a través de la tubería, en m^2.
- c_{p_f}: Capacidad térmica específica del fluido, en J/kg °C.
- ρ_f: Densidad del fluido, en kg/m^3.

Eficiencia geométrica del colector.

El cálculo de la energía transmitida al fluido caloportador necesita, previamente, conocer qué fracción del área de los espejos está dirigiendo adecuadamente la radiación solar hacia la tubería. Para ello, se ha de estudiar, por un lado, la posición del Sol respecto a los diferentes colectores solares y, por otro, la geometría de dichos colectores.

La posición del Sol respecto a los distintos colectores depende de varios factores: hora del día, día del año y coordenadas geográficas de la instalación. Sin embargo, establecer una relación entre estos parámetros y la posición solar puede ser, a menudo, complejo [8]. Surgen, como alternativa, las denominadas coordenadas solares temporales:

- **Declinación solar**, δ_1 [rad].

  La declinación solar se refiere al ángulo de posicionamiento del Sol respecto al plano terrestre ecuatorial. El cálculo de esta declinación solar viene dado por la conocida como ecuación de Spencer, en [68], mediante el siguiente desarrollo de Fourier:

$$\begin{cases} \delta_1 = 0.006918 - 0.399912\cos(w) + 0.070257\sin(w) - 0.006758\cos(2w) + \\ \qquad 0.000907\sin(2w) - 0.002697\cos(3w) + 0.001480\sin(3w) \\ w = \dfrac{2\pi}{365}(JD-1) \end{cases} \tag{3.8}$$

  donde w hace referencia al ángulo diario (es decir, el ángulo barrido por la Tierra en su movimiento de traslación alrededor del Sol) y JD corresponde al día juliano referido al año en curso (o, lo que es lo mismo, al número de días transcurridos desde el 1 de enero). Esto último, sin embargo, no es más que una simplificación que, dada la sencillez del modelo, resulta válida para una primera aproximación al problema.

- **Ángulo horario solar**, δ_2 [rad].

  El ángulo horario, por su parte, se define en [8] como el desplazamiento solar sobre el plano de la trayectoria solar, tomando como origen del ángulo el mediodía solar.

$$\delta_2 = (t_s\,[h] - 12)\,\frac{360^\circ}{24\,h}\,\frac{2\pi\,rad}{360^\circ} \tag{3.9}$$

  donde t_s corresponde a la hora solar. El cálculo de esta variable viene recogido en [8] como:

$$t_s = t_{local} + 4(L_{ref} - L_{loc}) + EoT \tag{3.10}$$

  donde:

  - t_{local}: Hora local estándar.
  - L_{ref} y L_{loc}: Longitudes del meridiano de referencia (meridiano central) y del meridiano local de la instalación.
  - EoT (*Equation of Time*): Ecuación empírica que corrige la inclinación axial y excentricidad de la órbita terrestre. La expresión de dicha ecuación viene dada en [3] por la siguiente serie de Fourier:

$$\begin{aligned} EoT = 229.18\,(&0.000075 + 0.001868\cos(w) - 0.032077\sin(w) - \\ &0.014615\cos(2w) - 0.04089\sin(2w)) \end{aligned} \tag{3.11}$$

  Se corrige, así, la desviación existente entre la hora local estándar y la hora solar debida, por un lado, a la diferencia de 4 minutos por cada grado de diferencia en la longitud y, por otro, a la distorsión provocada por la inclinación axial terrestre y la excentricidad de su órbita.

En la Figura 3.3 es posible observar los ángulos comentados anteriormente.

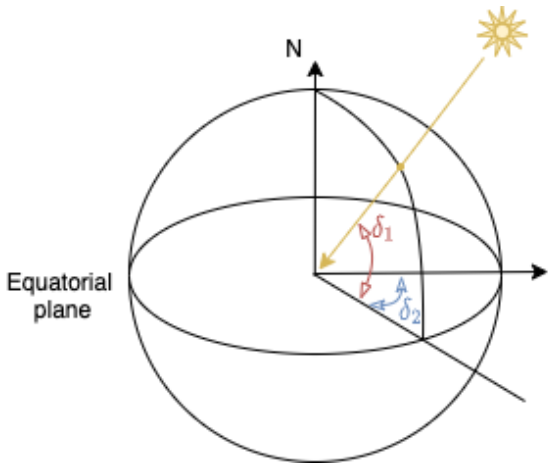


Figura 3.3 Representación de coordenadas solares temporales.

Finalmente, y a partir de la declinación solar y el ángulo horario [15], es posible determinar el ángulo de incidencia de la radiación solar sobre la superficie del colector.

$$\phi = \arccos\left(\sqrt{1-\cos^2(\delta_1)\sin^2(\delta_2)}\right) \tag{3.12}$$

La geometría de los colectores, por su parte, también juega un papel fundamental en el cálculo de la superficie efectiva de un colector. Así, en la literatura [19], pueden identificarse tres posibles efectos que disminuyan este área efectiva: las pérdidas en los extremos debido a la longitud finita de los colectores, A_e; las pérdidas por bloqueo de los rayos debido a las protecciones laterales, A_b; y las sombras que pudieran originarse entre las distintas filas de colectores, A_s.

No obstante, dado el objetivo de este trabajo, se toman ciertas hipótesis que permiten simplificar el problema. Por un lado, se considera que la distancia existente entre dos filas de colectores consecutivas es mucho mayor que la apertura de cada colector. Esto permite despreciar el área perdida debido al efecto de la sombra proyectada entre colectores adyacentes. Por otro lado, se asume que las pérdidas que tienen lugar debido a la longitud finita de los colectores es igual para todos los colectores, a pesar de que, en la práctica, el área perdida por aquellos colectores situados en los extremos de una hilera sería mayor que la perdida por aquellos intermedios debido a que no existen colectores adyacentes que reciban parte de la radiación reflejada.

Finalmente, bajo las hipótesis descritas, se puede extraer de [19] el siguiente cálculo:

$$\begin{cases} A_e = 4\,f\,W\tan(\frac{\phi_r}{3}) \\ A_b = 2\,f\,W\tan(\frac{\phi}{3}) \end{cases} \tag{3.13}$$

donde:

- f: Distancia focal del colector cilindro-parabólico.
- W: Dimensión transversal del colector.
- ϕ_r: Ángulo de incidencia a partir del cual la radiación solar incidente sobre un colector alcanza el tubo receptor del colector adyacente.

La ecuación (3.14), por tanto, reúne todos estos factores para el cálculo de la eficiencia geométrica del colector.

$$\eta_o = \left(1 - \frac{A_e + 2fW\tan(\frac{\phi}{3})}{fW}\right) \cdot \sqrt{1-\cos^2(\delta_1)\sin^2(\delta_2)} \tag{3.14}$$

3.2 Datos de entrada: perturbaciones.

A continuación, se identifican las variables de entrada al modelo que actúan como perturbaciones del sistema.

3.2.1 Irradiancia solar.

Los datos de irradiancia solar, o *Direct Normal Irradiance* (DNI), necesarios para las diferentes simulaciones del modelo se obtienen de una serie de perfiles generados de forma sintética. Además, con el objetivo de dotar de mayor variabilidad a este *dataset*, se le añaden los efectos de nubes virtuales que pudieran afectar al desempeño de la planta. En la Figura 3.4 se muestran algunos ejemplos de estos perfiles utilizados.

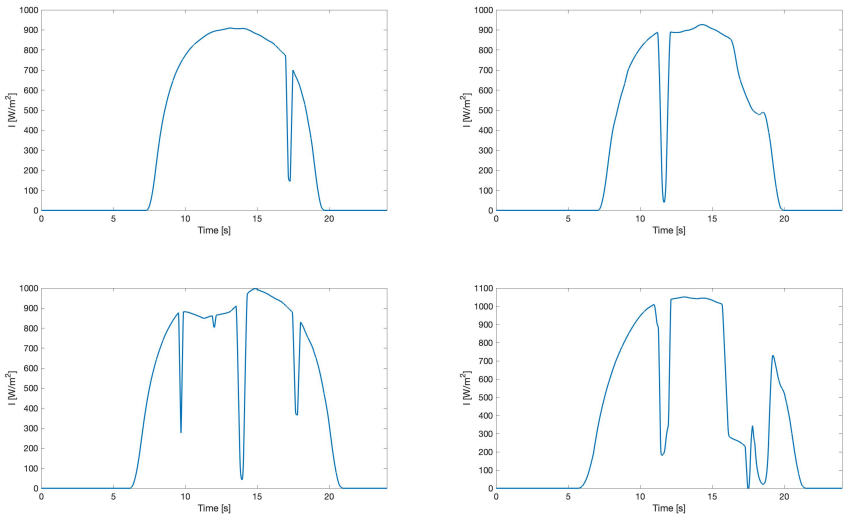


Figura 3.4 Ejemplos de perfiles DNI utilizados por el modelo.

3.2.2 Temperatura del aceite a la entrada del lazo de colectores.

La temperatura del aceite a la entrada del lazo de colectores, por su parte, se ve influenciada por la temperatura a la salida del mismo. Se recoge, en [60], dicha relación mediante una dinámica de primer orden con una constante de tiempo de 10 minutos (600 segundos).

$$\frac{T_{in}(s)}{\hat{T}_{out}(s)} = \frac{1}{600s+1} \tag{3.15}$$

donde $\hat{T}_{out}(s) = T_{out}(s) - 90$ representa la caída de temperatura que se produce en el generador de vapor de la instalación.

Finalmente, discretizando la ecuación anterior con un tiempo de simulación (o integración) de 0.25 segundos se obtiene la siguiente expresión:

$$\begin{aligned} T_{in}(k) &= \frac{600 - T_s}{600}\, T_{in}(k-1) + \frac{T_s}{600}\,(T_{out}(k) - 90) \\ &= 0.999583\, T_{in}(k-1) + 0.0004167\,(T_{out}(k) - 90) \end{aligned} \tag{3.16}$$

3.2.3 Temperatura ambiente.

La temperatura ambiente del sistema se toma constante durante toda la simulación. Sin embargo, para garantizar que el agente no ignore esta temperatura durante su aprendizaje, y dotar de mayor variabilidad al entorno, se tomará como temperatura ambiente un valor aleatorio entre 23 °C y 27 °C.

3.2.4 Eficiencia geométrica.

Finalmente, la eficiencia geométrica se calcula siguiendo las directrices recogidas en la Subsección 3.1.2. En realidad, la perturbación a tratar debería ser la posición relativa del Sol con respecto a los colectores. Sin embargo, dado que los colectores se consideran ya orientados en este trabajo, dicha posición se refleja directamente en la eficiencia geométrica del colector, lo que permite simplificar y reducir el número de variables del problema.

3.3 Tabla de parámetros.

Por último, se presenta un resumen de todos aquellos parámetros utilizados durante el modelado del sistema, junto a los valores de los mismos. De esta forma, se pretende facilitar al lector reproducir los resultados obtenidos en este trabajo.

Tabla 3.1 Parámetros del modelo de parámetros concentrados.

Símbolo	Descripción	Valor	Unidades
A_f	Área de paso del fluido a través de la tubería de un colector	$5.0671 \cdot 10^{-4}$	m^2
A_e	Superficie del colector perdida por efecto de los extremos	0.25	m^2
A_s	Superficie del colector perdida por sombras entre colectores	0	m^2
f	Distancia focal del colector cilindro-parabólico	1.1	m
G	Dimensión transversal del colector (apertura)	1.82	m
K_{opt}	Eficiencia óptica del colector	0.5661	p.u.
L_{loop}	Longitud del lazo de colectores	172	m
L_{ref}	Longitud geográfica del meridiano central	0	°
L_{loc}	Longitud geográfica del meridiano local de la instalación	-6.265939	°
S	Superficie total del colector	5.56	m^2
W	Dimensión longitudinal del colector	3.04	m

Además de estos parámetros, también resulta interesante recoger las restricciones presentes en el sistema y que se usarán, posteriormente, en el diseño de la estrategia de control. No obstante, en el Capítulo 4 se verá como estas restricciones de la Tabla 3.2 no tendrán por qué tratarse como restricciones duras, sino que podrán considerarse como restricciones blandas que penalicen la función objetivo.

Tabla 3.2 Restricciones del modelo de parámetros concentrados.

Símbolo	Descripción	Valor	Unidades
T_{out}^{max}	Temperatura de funcionamiento máxima del fluido	300	°C
T_{out}^{min}	Temperatura de funcionamiento mínima del fluido	200	°C
q^{max}	Máximo caudal admisible a través del lazo de colectores	1.2	l/s
q^{min}	Mínimo caudal admisible a través del lazo de colectores	0.2	l/s

Finalmente, y con perspectivas al futuro control de la instalación, se definen en la Tabla 3.3 las diferentes constantes de tiempo utilizadas en el entorno de simulación.

Tabla 3.3 Constantes de tiempo del modelo de parámetros concentrados.

Símbolo	Descripción	Valor	Unidades
T_{sim}	Periodo de simulación / Tiempo de integración	0.25	s
T_{sample}	Periodo de muestreo	1	s
$T_{control}$	Periodo de control	30	s

4 Formulación del problema.

The only way you can get to the very positive scenario is by great innovation. Innovation really does bend the curve.

BILL GATES

En este capítulo se desarrollan los detalles necesarios para la implementación del sistema de control en un lenguaje de programación. Se presenta, para ello, el algoritmo de aprendizaje reforzado utilizado por el agente de control, así como la librería de estandarización utilizada para la interfaz agente-escenario.

Como ya se ha introducido en el Capítulo 1, el objetivo perseguido en este trabajo es el de reproducir los resultados obtenidos por controladores predictivos, o MPC, en el contexto del control de plantas termosolares de colectores cilindro-parabólicos superando, a su vez, las limitaciones propias de estos últimos [40]. Para ello, se utilizará una arquitectura de control basada en técnicas de DRL que aproveche la experiencia recopilada por el agente para desarrollar la estrategia de control. Se busca, con esto, diseñar controladores flexibles y de rápida ejecución que permitan reducir el alto coste computacional característico de los controladores MPC no lineales, así como rebajar las exigencias en cuanto al conocimiento necesario sobre el sistema a controlar.

4.1 Algoritmo de aprendizaje.

En el contexto de un problema de RL, el algoritmo de aprendizaje es el elemento que da forma al cerebro del agente de control. Este algoritmo es el encargado de guiar el proceso de aprendizaje seguido por el agente para, con la experiencia recogida de su entorno, ser capaz de desarrollar una política de actuación óptima ante los estímulos recibidos. En el caso abordado por este trabajo, el algoritmo elegido es el *Deep Deterministic Policy Gradient* (DDPG) [41].

4.1.1 Deep Deterministic Policy Gradient (DDPG).

El algoritmo DDPG es un algoritmo *model-free* que utiliza funciones de aproximación basadas en redes neuronales para desarrollar, como su propio nombre indica, una política de actuación óptima determinista. Este algoritmo, a diferencia de otros predecesores como el DQN, tiene su ámbito de aplicación en problemas con espacios de acción continuos y de altas dimensiones. Esto hace de este algoritmo un candidato ideal para la aplicación del DRL al control de sistemas físicos.

Este algoritmo, denominado como algoritmo de tipo *actor-critic*, presenta dos comportamientos bien diferenciados: *actor* y *critic*. Por un lado, el componente *actor* del agente es el responsable de ajustar los parámetros θ^{μ} para dar forma a la política $\mu_{\theta^{\mu}}(s)$ a partir de la ecuación (2.27) aplicada a políticas deterministas. Esta ecuación, utilizando la función de aproximación $Q_{\theta^{Q}}(s,a)$ en lugar de $Q_{\mu_{\theta^{\mu}}}(s,a)$, resulta:

$$\begin{aligned} \nabla_{\theta^{\mu}} J(\mu_{\theta^{\mu}}) &\approx \mathbb{E}_{\tau\sim\mu_{\theta^{\mu}}}\left[\nabla_{\theta^{\mu}} Q_{\theta^{Q}}(s,a)\right] \\ &\approx \mathbb{E}_{\tau\sim\mu_{\theta^{\mu}}}\left[\nabla_{a} Q_{\theta^{Q}}(s,a)|_{a=\mu_{\theta^{\mu}}(s)} \nabla_{\theta^{\mu}} \mu_{\theta^{\mu}}(s)\right] \end{aligned} \tag{4.1}$$

Por otro lado, el componente *critic* del algoritmo se encarga de obtener una aproximación de la función valor estado-acción real $Q_\mu(s,a)$. Para ello, se utiliza la ecuación (2.18) para calcular, de forma recursiva, la función de aproximación $Q_{\theta^Q}(s,a)$ [66]:

$$Q_{\theta^Q}(S_t, A_t) = Q_{\theta^Q}(S_t, A_t) + \alpha\big(R_{t+1} + \gamma \max_{a' \in \mathcal{A}} Q_{\theta^Q}(S_{t+1}, a') - Q_{\theta^Q}(S_t, A_t)\big) \tag{4.2}$$

Así, mientras que el componente *critic* evalúa la bondad de cada uno de los estados alcanzados por el agente, el componente *actor* desarrolla la política de actuación que permite alcanzar los estados de mayor beneficio. Se genera, por tanto, una política de actuación mixta basada tanto en valor como en política.

No obstante, un problema surgido en este tipo de métodos donde se implementan estrategias basadas en valor a partir de redes neuronales es la divergencia que puede resultar de actualizar $Q_{\theta^Q}(s,a)$ utilizando la filosofía *Time Difference*. Esto se debe a que, como se observa en la ecuación (4.2), es necesario utilizar los valores derivados de la red $Q_{\theta^Q}(s,a)$ para la actualización de sí misma. Como novedad, en este método se propone la utilización de unos valores objetivo o *target* que imiten el comportamiento de las redes originales pero suavizadas en el tiempo. Se crearían, así, dos nuevas redes neuronales, $Q_{\theta^{Q'}}(s,a)$ y $\mu_{\theta^{\mu'}}(s)$, denominadas *target critic* y *target actor*. La actualización de estas redes se llevaría a cabo de forma suave realizando un *Polyak update*:

$$\theta' \leftarrow \zeta\theta + (1-\zeta)\theta' \tag{4.3}$$

donde ζ corresponde al tamaño de paso de la actualización y debe cumplir $\zeta \ll 1$ para garantizar una mayor estabilidad del algoritmo.

Finalmente, en el Algoritmo 1 se resume la secuencia seguida por el proceso de aprendizaje en forma de pseudocódigo.

4.1.2 Redes neuronales.

Se presentan, a continuación, las arquitecturas de las diferentes redes neuronales utilizadas para las funciones de aproximación de la política de acción μ_{θ^μ} (*actor*) y de la función valor estado-acción Q_{θ^Q} (*critic*).

Actor Network.

La *Actor Network* está compuesta por una sucesión de capas densas que permiten asociar una acción a a cada estado s del escenario. Esta red, así como la *target network* asociada, consta de los siguientes elementos:

- Tres capas densas de 400, 300 y 1 neurona respectivamente.
- Sendas capas de normalización a la salida de cada capa densa.
- Funciones de activación ReLU a la salida de cada una de estas capas de normalización, y una función sigmoidal a la salida de la red que permite acotar el valor de la acción a tomar.

Critic Network.

La *Critic Network*, por su parte, cuenta con una estructura de dos ramas en paralelo. En primer lugar, la primera rama se encarga de analizar el estado s del escenario mientras que, por otro lado, hay una segunda rama encargada de procesar la acción a. Teniendo en cuenta el resultado de ambas ramas, la red proporciona una estimación del valor estado-acción asociado a dicho par (s,a). Se distinguen, en Figura 4.1, los siguientes elementos:

- Dos ramas paralelas de capas densas. Una primera rama con dos capas de 400 y 300 neuronas respectivamente y, una segunda con una única capa de 300 neuronas. Finalmente, ambas ramas se unen en una última capa con una neurona que proporciona el valor estado-acción deseado.
- Sendas capas de normalización a la salida de cada capa densa en la primera rama.
- Funciones de activación ReLU a la salida de cada una de estas capas de normalización.

Algoritmo 1 Algoritmo DDPG.

Inicialización aleatoria de las funciones de aproximación $Q_{\theta^Q}(s,a)$ y $\mu_{\theta^\mu}(s)$.
Inicialización de las funciones de aproximación objetivo $Q_{\theta^{Q'}}(s,a)$ y $\mu_{\theta^{\mu'}}(s)$: $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$.
Inicialización del *buffer* de experiencias.
for episodio $= 1$, $N_{EPISODES}$ **do**
    Inicialización del ruido de exploración.
    Inicialización del escenario: estado inicial s_1.
    for step $= 1$, T_{sim} **do**
        Cálculo y ejecución de la acción a utilizando ruido de exploración: $a \leftarrow \mu_{\theta^\mu}(s) + \mathcal{N}$.
        Leer nuevo estado alcanzado s' y recompensa r.
        Almacenar transición $e_t = (s,a,r,s')$ en el *buffer* de experiencias.
        Muestrear un *minibatch* del *buffer* de experiencias.
        Cálculo de valor estado-acción objetivo para cada muestra i:

$$y_i = r_i + \gamma\, Q_{\theta^{Q'}}(s'_i, \mu_{\theta^{\mu'}}(s_i))$$

        Actualización de la red *critic* mediante la función de pérdidas de las N muestras:

$$\nabla_{\theta^Q}\mathcal{L} \approx \nabla_{\theta^Q}\frac{1}{N}\sum_{i=1}^{N}\left(y_i - Q_{\theta^Q}(s,a)\right)^2$$

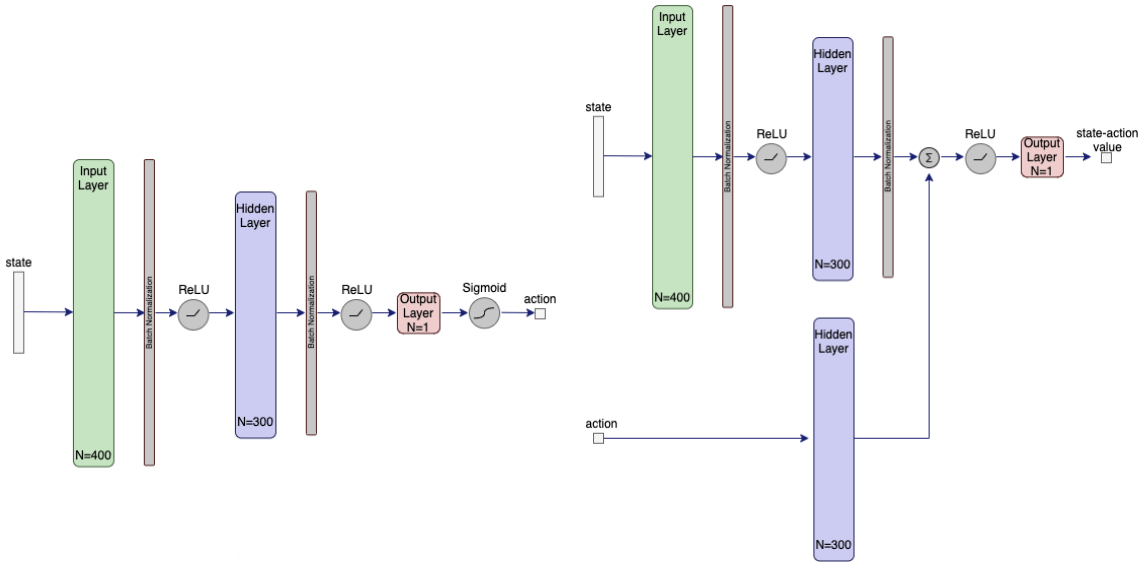
        Actualización de la red *actor* mediante la función objetivo J:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_{i=1}^{N}\nabla_a Q_{\theta^Q}(s,a)|_{a=\mu_{\theta^\mu}(s)}\nabla_{\theta^\mu}\mu_{\theta^\mu}(s)$$

        Actualización de las funciones de aproximación objetivo (redes) $Q_{\theta^{Q'}}(s,a)$ y $\mu_{\theta^{\mu'}}(s)$:

$$\theta^{\mu'} \leftarrow \zeta\,\theta^\mu + (1-\zeta)\theta^{\mu'}$$
$$\theta^{Q'} \leftarrow \zeta\,\theta^Q + (1-\zeta)\theta^{Q'}$$

    end for
end for



(a) *Actor Network.* **(b)** *Critc Network.*

Figura 4.1 Arquitecturas de las redes neuronales utilizadas.

4.2 Gym Environment.

Un factor clave a considerar a la hora de implementar un algoritmo de AI es identificar el lenguaje de programación que más se adapta a las necesidades del proyecto. Así, en función del objetivo perseguido por dicho proyecto, puede resultar más práctico utilizar un lenguaje u otro: R o Scala para análisis estadísticos exhaustivos de grandes cantidades de datos, Julia y MATLAB para análisis de más alto nivel, o C en el caso de que la velocidad y eficiencia en la ejecución jueguen un papel más importante. No obstante, debido al soporte proporcionado por la comunidad de RL en forma de librerías y a la popularidad del lenguaje en este tipo de aplicaciones, en el presente trabajo se opta por utilizar Python como lenguaje de programación.

Este lenguaje, además de contar con multitud de librerías orientadas al aprendizaje automático como TensorFlow o PyTorch, proporciona a la comunidad de *Machine Learning* una plataforma de código abierto donde compartir sus avances y resultados. Un ejemplo de esto es la librería *Gym*, la cual se ha convertido en la plataforma estándar para poner a prueba cualquier algoritmo de aprendizaje reforzado desarrollado.

4.2.1 Introducción a OpenAI Gym.

OpenAI Gym [10] surge, en 2016, como una librería de código abierto implementada en Python que busca establecer una interfaz de comunicación estándar entre cualquier algoritmo de aprendizaje reforzado y los entornos de simulación utilizados para su entrenamiento. Para ello, Gym proporciona una *Application Programming Interface* (API) que sirve de enlace entre algoritmo y entorno, permitiendo al usuario abstraerse del funcionamiento del escenario durante la programación del agente de control. Esto facilita la comparación entre diferentes algoritmos y la cooperación dentro de la comunidad de RL.

La filosofía bajo la cual funciona Gym está basada en el "agent-environment loop" representado en la Figura 4.2. Este bucle de interacción entre el agente de control y el entorno de simulación está constituido por las siguientes etapas:

- El agente realiza una acción (*action*) sobre el entorno a través de, normalmente, una señal de control.
- A partir de dicha acción, el entorno evoluciona alcanzando un nuevo estado. Este nuevo estado, u *observation*, es devuelto al agente junto a una señal de recompensa, o *reward*, que indica cuán buena ha resultado la interacción realizada.

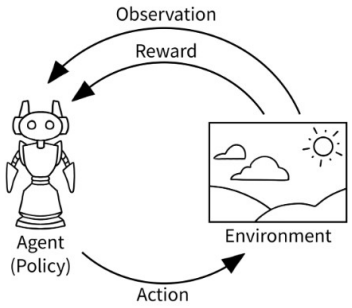


Figura 4.2 Bucle de interacción entre agente y entorno [50].

Esta sucesión de interacciones, denominadas *steps*, en las que el agente realiza una acción y el entorno le devuelve su nuevo estado y la respectiva recompensa, tiene lugar hasta que el entorno alcanza un estado considerado terminal. Este estado terminal puede ser alcanzado bien debido a algún fallo ocurrido durante la simulación, o bien debido a alguna condición de finalización de simulación deseada, y se notifica al agente devolviendo una condición de bandera booleana denominada *done*.

El aprendizaje dentro de un entorno Gym tiene lugar de forma episódica. De esta forma, la interacción del agente con el entorno se divide en forma de episodios (*episodes*) que abarcan desde un estado inicial hasta un estado terminal de finalización. El objetivo del RL es, a lo largo de uno de estos episodios, lograr maximizar la recompensa obtenida. Finalmente, para devolver un entorno a su estado inicial se realiza un *reset*.

4.2.2 Campo de colectores solares ACUREX como Gym Environment.

En esta sección se describe una adaptación del modelo descrito en la Sección 3.1 siguiendo las indicaciones de la API de la librería Gym. Para ello, Gym propone la utilización de dos atributos en cada entorno: el *action_space* y el *observation_space*. Estos atributos son los encargados de definir el formato en el que agente y entorno intercambian la información relativa a la acción realizada y al estado alcanzado. En [50] se recogen diferentes formatos para la definición de estos espacios, pero, dada la naturaleza del problema objeto de estudio, basta con espacios n-dimensionales continuos (o *Box*).

Además de estos dos espacios, Gym cuenta con una serie de métodos estándares que reflejan cada una de las interacciones descritas en la Subsección 4.2.1: *step()*, *reset()*, y *render()*. Es mediante la interfaz compuesta por estos métodos como se consigue la abstracción del entorno durante el diseño del agente de control.

Env.action_space.

Este problema, al tratar únicamente el control de un lazo de colectores, solo consta de una única acción de control: el caudal de fluido caloportador que circula a través de dicho lazo. Por tanto, el espacio de acción en este caso está compuesto por:

$$action = [q(k)]$$

Env.observation_space.

El *observation_space*, por su parte, representa el estado del entorno. En este caso, el estado utilizado contiene todas las variables involucradas en el proceso de control: variable a controlar, variable de actuación y perturbaciones. Además, dado que el objetivo del agente de control es el de maximizar la potencia térmica neta, W_{th}, producida por la planta, se incluye en el estado la potencia térmica del fluido tanto a la entrada del lazo de colectores (W_{in}) como a la salida (W_{out}). El cálculo de esta potencia térmica se realiza tal que:

$$\left.\begin{aligned} W_{in} &= c_{pf}(T_{in})\,\rho_f(T_{in})\,q\,T_{in} \\ W_{out} &= c_{pf}(T_{out})\,\rho_f(T_{out})\,q\,T_{out} \end{aligned}\right\} \longrightarrow W_{th} \approx W_{out} - W_{in} \tag{4.4}$$

Por último, se añade en la representación del estado tanto un histórico de los valores tomados por cada una de las señales consideradas de interés como una previsión a futuro de todas aquellas perturbaciones existentes en el sistema. De esta forma, se le proporciona al agente de control una visión más global del funcionamiento del entorno, facilitándole, así, que deduzca las dinámicas tan lentas existentes en el escenario en cuestión. Además, al contar con una predicción de la evolución futura de las perturbaciones, el agente de control dispone de la información necesaria para, siguiendo la filosofía del control predictivo, anteponerse a dichas perturbaciones y actuar en consecuencia.

$$\begin{aligned} state = [\quad & & & T_{out}(k), & \ldots, & T_{out}(k-d_y), \\ & & & q(k), & \ldots, & q(k-d_u), \\ & I_{rad}(k+d_{df}), & \ldots, & I_{rad}(k), & \ldots, & I_{rad}(k-d_{dp}), \\ & T_{in}(k+d_{df}), & \ldots, & T_{in}(k), & \ldots, & T_{in}(k-d_{dp}), \\ & T_{amb}(k+d_{df}), & \ldots, & T_{amb}(k), & \ldots, & T_{amb}(k-d_{dp}), \\ & \eta_o(k+d_{df}), & \ldots, & \eta_o(k), & \ldots, & \eta_o(k-d_{dp}), \\ & & & W_{out}(k), & \ldots, & W_{out}(k-d_y), \\ & & & W_{in}(k), & \ldots, & W_{in}(k-d_y) \quad] \end{aligned}$$

donde d_y, d_u y d_{dp} hacen referencia al tamaño de las ventanas temporales utilizadas para el muestreo de históricos de la variable a controlar, la señal de control y las perturbaciones, respectivamente, y d_{df} se refiere a la ventana temporal de la previsión a futuro de las perturbaciones.

Env.step().

El método *env.step()* es el encargado de simular la dinámica del sistema y calcular la evolución de este en función de la señal de control utilizada por el agente.

Código 4.1 Interfaz de uso del método env.step().

```
[observation, reward, done, info] = gym.Env.step(action)
```

En la línea de código presente en Código 4.1 se presenta la interfaz de uso del método *env.step()* según indica la API de la librería Gym. En ella, se distinguen los siguientes argumentos:

- Argumentos de entrada.
  - *action*: Acción comandada por el agente de control.
- Argumentos de salida.
  - *observation*: Estado alcanzado por el entorno tras realizar la acción.
  - *reward*: Recompensa asociada a la acción tomada por el agente.
  - *done*: Bandera booleana que indica el final del episodio al alcanzar un estado terminal.
  - *info*: Diccionario que permite añadir otra información útil para el agente.

De esta forma, a partir de la acción de control recibida como argumento de entrada, el entorno aplica las ecuaciones dinámicas del modelo (desarrolladas en la Subsección 3.1.2) para calcular la evolución del mismo. No obstante, dado que en el problema considerado existe una discrepancia entre el periodo de control utilizado por el agente, $T_{control}$, y el tiempo de integración utilizado en las ecuaciones, T_{sim}, el método *env.step()* debe realizar tantos pasos de integración como sean necesarios para alcanzar el periodo de control.

A partir de esta evolución del sistema, el entorno devuelve una representación del nuevo estado alcanzado: una *observation*. Este estado debe seguir la estructura descrita en el *observation_space* y se construye tanto a partir de los históricos pasados de los valores que han tomado las distintas variables como de las previsiones a futuro de las diferentes perturbaciones. Además, con vistas a facilitar el posterior aprendizaje del agente, se opta por normalizar el estado en cuestión, utilizando, para ello, los valores máximos y mínimos que pueden adoptar cada una de las variables [58].

Junto al nuevo estado alcanzado, el entorno de simulación debe devolver al agente una medida de cómo de buena ha sido la actuación realizada. A este parámetro se le denomina recompensa o *reward* y le sirve al agente para desempeñar su aprendizaje. En este caso, dado que el objetivo del agente es el de maximizar la potencia térmica obtenida, se utiliza la siguiente función de recompensa:

$$reward = W_{th} - \psi \ \text{máx}\left(\frac{T_{out}(k) - T_{out}^{max}}{T_{out}^{max}}, \frac{T_{out}^{min} - T_{out}(k)}{T_{out}^{max}}, 0\right)^2 - \varepsilon \ (q(k) - q(k-1))^2 \tag{4.5}$$

donde ψ y ε corresponden a sendos parámetros de ajuste que regulan la penalización recibida por salirse del rango óptimo de funcionamiento o por presentar *slew rates* demasiado altos, respectivamente.

Finalmente, se ha de comprobar si el entorno alcanza el estado terminal (*done*) para poner fin a la simulación. Para ello, se definen diferentes condiciones de finalización:

- Se alcanza el límite de simulación establecido.
- Se alcanza la temperatura de congelación del fluido caloportador: -25 °C.
- Se alcanza la temperatura de degradación del fluido caloportador: 315 °C.

Env.reset().

El método *env.reset()*, por su parte, es el encargado de reiniciar el estado del entorno y sus atributos. Este método lleva al entorno a su estado inicial antes de comenzar una nueva simulación.

Código 4.2 Interfaz de uso del método env.reset().

```
observation = gym.Env.reset(seed)
```

Los parámetros de entrada y salida de este método se reducen, respectivamente, a una semilla, o *seed*, que permite indicar el estado de inicialización deseado y al estado del entorno que se devuelve tras el reinicio. En este caso, la semilla se compone de tres parámetros que indican el estado de inicialización del entorno: perfil de irradiancia utilizado para la simulación, hora local de la instalación en la que da comienzo dicha simulación y duración de la simulación.

Env.render().

Por último, el método *env.render()* permite representar la evolución del entorno.

Código 4.3 Interfaz de uso del método env.render().

```
gym.Env.render()
```

La representación de la evolución del sistema se lleva a cabo mediante una gráfica como la de la Figura 4.3 donde se muestran los tres siguientes campos:

- Evolución del sistema. Consiste en la representación de la evolución de la variable controlada del sistema, así como de la potencia térmica transportada por el fluido.
- Acción de control. Representación del caudal de fluido que atraviesa el lazo de colectores.
- Evolución de las perturbaciones. Evolución de las distintas perturbaciones sufridas por el sistema.

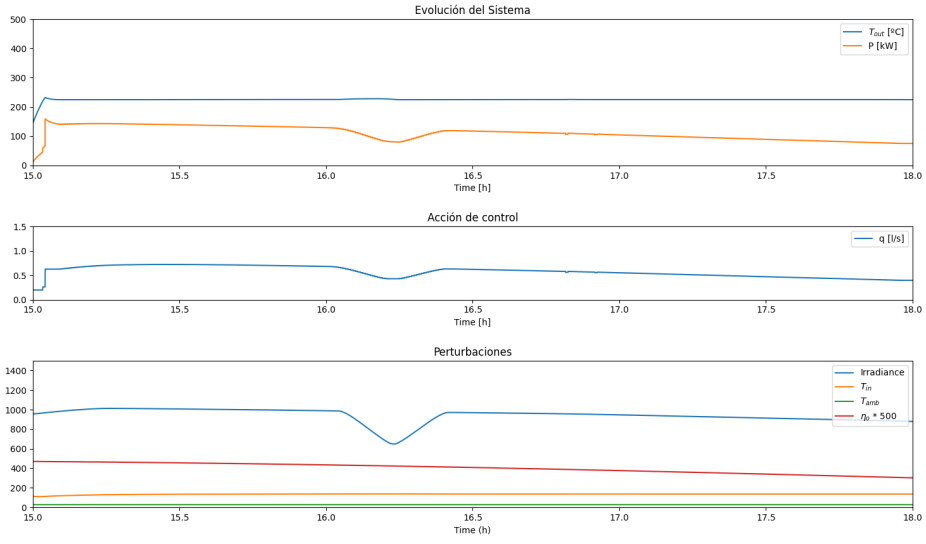


Figura 4.3 Representación de la evolución del sistema.

4.3 Algoritmo de referencia: MPC.

Se presenta, por último, la formulación del algoritmo de control utilizado como referencia: un controlador predictivo basado en modelo o MPC. Este algoritmo de control establece, a cada instante de control, un problema de optimización en el que se trata de calcular aquella secuencia de acciones de control óptima que permite minimizar una función objetivo o función de coste. Esta optimización se realiza de tal forma que, a partir del conocimiento del modelo del sistema, se calcula una predicción de los valores adoptados por la salida del mismo y se anticipa, así, a los efectos de posibles variaciones en las perturbaciones. Finalmente, y tras resolver dicho problema de optimización, el controlador aplica sobre el sistema la primera acción de control de la secuencia obtenida y espera hasta el próximo instante de control para repetir el proceso.

Como ya se ha mencionado anteriormente en el Capítulo 2, el control predictivo basado en modelo ya ha sido ampliamente utilizado para resolver el problema de control de plantas termosolares [25, 43, 60]. En esta ocasión, en concreto, se opta por retomar la formulación propuesta en [60]. Así, se plantea un problema de control en el que el objetivo del controlador es maximizar la potencia eléctrica producida a lo largo de un lazo de colectores y minimizar, en el proceso, las penalizaciones derivadas de temperaturas salidas del rango de funcionamiento óptimo así como de *slew rates* demasiado elevados. La función de coste utilizada, por tanto, resulta de la siguiente forma:

$$J(k) = -W_{th}(k) + \psi \ \text{máx}\left(\frac{T_{out}(k) - T_{out}^{max}}{T_{out}^{max}}, \frac{T_{out}^{min} - T_{out}(k)}{T_{out}^{max}}, 0\right)^2 + \varepsilon \ (q(k) - q(k-1))^2 \tag{4.6}$$

Con esto, el problema de optimización a resolver a cada instante de control se reduce a:

$$\begin{aligned} &\min_{q_k(k_c)} \sum_{k=k_c}^{k_c+N_p} J(k|k_c) \\ &s.a. \ \ q^{min} < q(k|k_c) < q^{max} \qquad \forall k = k_c, ..., k_c + N_u - 1 \\ &\qquad q(k|k_c) = q(k_c + N_u - 1|k_c) \qquad \forall k = k_c + N_u, ..., k_c + N_p \end{aligned} \tag{4.7}$$

donde N_p y N_u corresponden, respectivamente, a los horizontes de predicción y control del problema de optimización.

Por último, y con vistas a poder realizar una futura comparación entre métodos, se opta por utilizar en este trabajo un algoritmo TNC o algoritmo de Newton truncado como algoritmo de optimización. Este tipo de algoritmos establecen un proceso recursivo con el que determinar una solución aproximada al método de Newton y resulta ideal para la optimización de funciones no lineales con un gran número de variables independientes. Asimismo, para facilitar la réplica de los resultados obtenidos, en la Tabla 4.1 se recogen los parámetros utilizados para definir el problema de optimización completo.

Tabla 4.1 Parámetros del controlador MPC.

Símbolo	Descripción	Valor
N_p	Horizonte de predicción	5
N_u	Horizonte de control	3
T_{sim}	Periodo de simulación / Tiempo de integración [s]	0.25
$T_{control}$	Periodo de control [s]	30
ψ	Coeficiente de penalización por temperaturas fuera del rango óptimo	10^7
ε	Coeficiente de penalización por *slew rate*	10^5

5 Simulación. Resultados y análisis.

Theory provides the maps that turn an uncoordinated set of experiments or computer simulations into a cumulative exploration.

David E. Goldberg

Este capítulo presenta los resultados obtenidos del entrenamiento y la evaluación del agente de control desarrollado ante un simulador de la planta solar ACUREX descrita en el Capítulo 3. Estos resultados se complementan, además, con un análisis del desempeño demostrado por el agente en cuestión, así como con una breve comparativa de este con el controlador predictivo utilizado como referencia a lo largo de todo el presente trabajo.

5.1 Entrenamiento del agente de control.

El aprendizaje del agente de control tiene lugar a través de un proceso iterativo denominado entrenamiento o *training*. Este entrenamiento, proceso mediante el cual se lleva a cabo la actualización de los pesos de las redes neuronales utilizadas como aproximadores de las funciones valor estado y estado-acción, requiere de una configuración previa a través de una serie de hiperparámetros que definen el aprendizaje del algoritmo. En concreto, en este proyecto se trabaja con los siguientes hiperparámetros:

- ***Batch Size***. El *batch size*, o tamaño de lote, hace referencia al número de muestras tomadas del *buffer* de experiencias para el cálculo de la función gradiente a cada paso del proceso de entrenamiento. De esta forma, un adecuado ajuste de este hiperparámetro puede conducir a una mayor velocidad de aprendizaje, ya que, al dotar al agente de más información acerca de la función gradiente, más fácil será para este converger en la dirección correcta. No obstante, se ha de prestar especial atención a que un aumento del tamaño del lote no repercuta en una peor capacidad de generalización del agente. Por lo general, resulta necesario encontrar un valor de compromiso [38].
- **Factor de descuento**, γ. Este hiperparámetro corresponde al factor de descuento de la función de recompensas acumuladas por el agente de control (2.3). Este factor define la importancia relativa de las recompensas futuras con respecto a aquellas más inmediatas y puede entenderse como el análogo al horizonte de predicción de un controlador predictivo.
- ***Learning rates***, α_{actor} y α_{critic}. Estos hiperparámetros marcan en qué medida se incorpora el gradiente de la función de pérdidas (o función objetivo) en los pesos de las redes neuronales *actor* y *critic* durante el entrenamiento (2.29). Una vez más, es necesario encontrar un valor de compromiso, ya que, mientras que valores demasiado bajos de estos hiperparámetros pueden ralentizar enormemente el aprendizaje de las redes, unos valores demasiado altos podrían incurrir en la inestabilidad de las mismas.
- **Factor de actualización de las redes *target***, ζ. Este hiperparámetro, por su parte, define la tasa de actualización de las redes *target actor* y *target critic* (4.3). Este hiperparámetro puede entenderse, por tanto, como la velocidad a la que se mueven los *moving target* perseguidos por las funciones valor estado y estado-acción durante el entrenamiento del agente.

- **Parámetros de la función de recompensa**, ψ y ε. Finalmente, los parámetros ψ y ε corresponden a los pesos utilizados en la ecuación (4.5) para ponderar las penalizaciones en la función de recompensa. Un cuidado ajuste de estos dos parámetros es esencial para garantizar la convergencia del algoritmo.

En este trabajo, los valores utilizados para cada uno de los hiperparámetros anteriores se recogen en la Tabla 5.1. Para ello, se han tomado valores estándares cuyo buen funcionamiento ya ha sido demostrado anteriormente en otros trabajos de investigación como [69].

Tabla 5.1 Hiperparámetros de entrenamiento.

Hiperparámetro	Valor
Batch Size	128
γ	0.95
α_{actor}, α_{critic}	10^{-4}
ζ	10^{-3}
ψ	10^{7}
ε	10^{5}

En la Figura 5.1, por último, se recoge el proceso de entrenamiento del agente de control. Resulta interesante, a partir de la evolución de la suma de recompensas obtenidas en cada episodio (o *score*), analizar cuán efectivo ha sido dicho entrenamiento. En este caso, se observa un acusado primer aprendizaje durante los primeros episodios de simulación donde el agente aprende a respetar las restricciones del sistema evitando alejarse mucho del rango de temperaturas óptimo. Además de esto, en torno al episodio 1000 se vuelve a observar, aunque esta vez de forma algo más suave, una nueva tendencia creciente de la recompensa media obtenida. En este caso, puede deducirse un aprendizaje del agente de control centrado en la optimización de la función de recompensa mediante un ajuste fino de la misma en el que trata de eliminar grandes saltos en la acción de control y afinar en la maximización de la potencia térmica generada. En la figura se aprecia, además, una caída de de la curva de aprendizaje en torno al episodio 2500. Este fenómeno no resulta especialmente preocupante ya que se puede achacar al componente estocástico del algoritmo, el cual, durante un cierto número de episodios, ha detectado un óptimo local que ha llevado al agente a evolucionar en esa dirección. Esta tendencia, sin embargo, es rápidamente corregida para continuar con el aprendizaje.

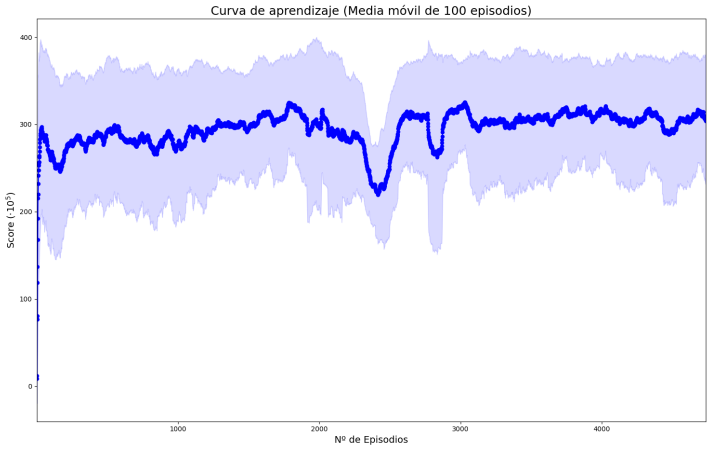


Figura 5.1 Evolución del entrenamiento del agente de control.

Se ha de destacar, sin embargo, que tanto en la gráfica anterior como durante le proceso de entrenamiento se han eliminado los *outliers* existentes. Estos *outliers* corresponden a aquellas simulaciones realizadas con perfiles de irradiancia donde una nube ha bloqueado toda la incidencia solar sobre los colectores. En estos casos, a pesar de que el control intenta llevar al sistema a un punto de funcionamiento válido, la ausencia de energía solar hace que se acumule una recompensa negativa demasiado grande y, por tanto, se desvirtúe la medida de la progresión del agente.

5.2 Evaluación del agente de control.

Una vez entrenado el agente, conviene evaluar la bondad del control ofrecido por este a través de diferentes simulaciones. Para ello, se representa el comportamiento del sistema controlado en bucle cerrado utilizando, como controlador, el agente de control basado en DRL (Figura 5.2 - Figura 5.4). Esta representación recoge la evolución de la variable controlada, que, en este caso, puede entenderse bien como la temperatura del fluido caloportador a la salida del lazo de colectores o bien como la potencia térmica neta generada por la instalación, así como la acción de control realizada por el agente sobre el entorno y los perfiles de perturbación que afectan a la simulación.

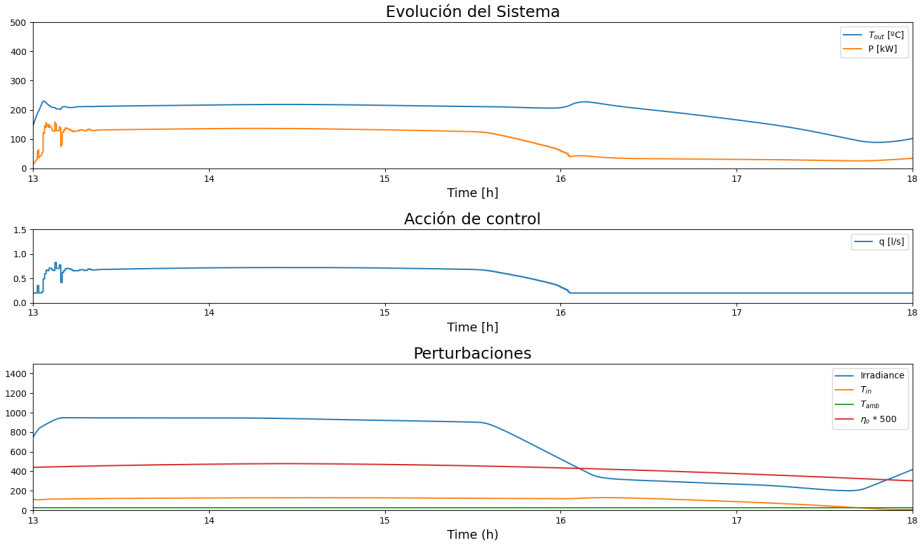


Figura 5.2 Evolución del sistema controlado por agente de control DRL (Perfil 1).

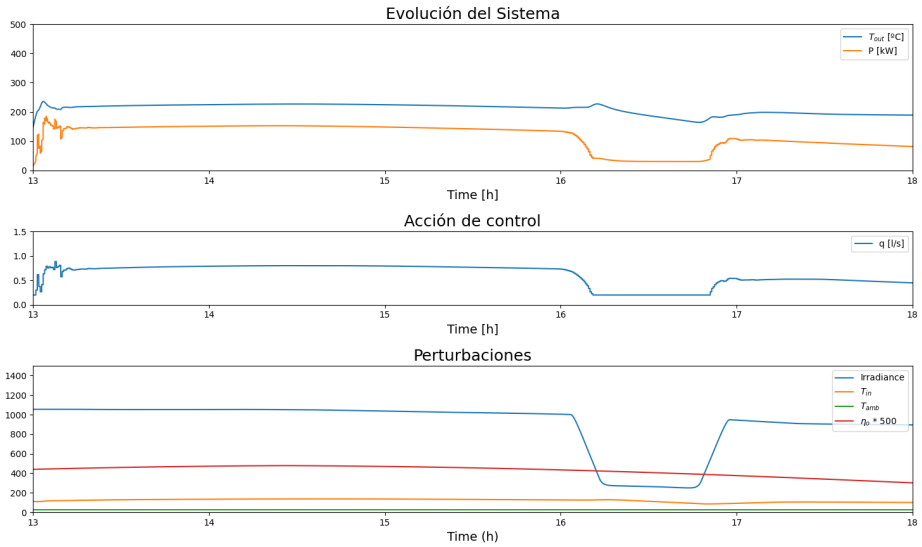


Figura 5.3 Evolución del sistema controlado por agente de control DRL (Perfil 2).

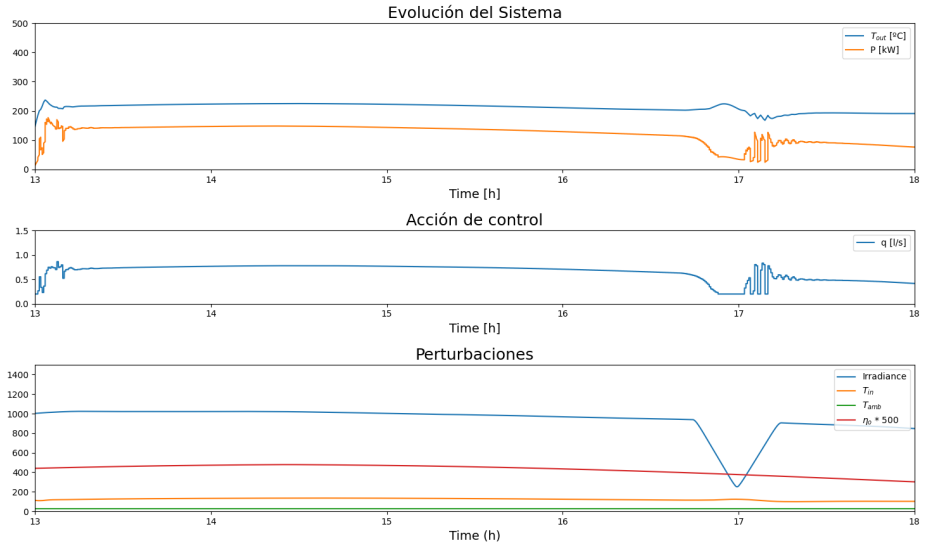


Figura 5.4 Evolución del sistema controlado por agente de control DRL (Perfil 3).

Por lo general, se demuestra un buen comportamiento del sistema controlado en bucle cerrado en el que el agente es capaz de prever los cambios en las perturbaciones para actuar sobre el caudal de fluido caloportador a tiempo de corregir, en la medida de lo posible, las posibles desviaciones que estas pudieran producir sobre la temperatura del fluido. No obstante, cabe destacar dos detalles apreciables en estas gráficas:

- En primer lugar, se observa cómo el comportamiento del agente de control durante los instantes iniciales de la simulación no resulta especialmente bueno. Estas oscilaciones observadas durante los primeros minutos simulados se deben a que, a diferencia de lo ocurrido con un controlador predictivo, el agente no es capaz de predecir a futuro la evolución de la variable de salida, sino que, en su lugar, aprende de las conclusiones derivadas de dinámicas observadas en el pasado. Así, durante aquellos instantes donde el agente de control aún no cuenta con información suficiente para rellenar la ventana temporal utilizada en la descripción del estado, el comportamiento demostrado resulta algo más errático.

  No obstante, dado que la finalidad del agente es servir de controlador durante la explotación continua de la instalación, este problema puede considerarse de menor importancia al observarse únicamente durante un corto transitorio al comienzo de su funcionamiento. Igualmente, podrían proponerse controladores más sencillos que permitieran, durante estos primeros minutos, posicionar al sistema en un punto de funcionamiento deseado mientras el agente de control inicializa su estado y, con ello, eliminar estas primeras oscilaciones.

- En segundo lugar, también se observa como el agente de control tiene ciertos problemas para controlar el sistema cuando se producen cambios bruscos en la irradiancia recibida por los colectores. Estos cambios repentinos en la cantidad de energía recibida pueden provocar cierta inestabilidad en la acción de control y, con ello, una señal algo ruidosa que pudiera afectar a los equipos físicos del sistema. No obstante, podría añadirse un filtro a la salida de dicha señal que impida saltos demasiado bruscos en la acción de control de forma que se eviten posibles estos daños. De la misma forma, cabría estudiar un ajuste más cuidado de los hiperparámetros de la función de recompensa para observar si, al igual que ocurre en ocasiones con los controladores MPC, esto solucionara las oscilaciones apreciadas.

Por último, resulta interesante comentar ciertos aspectos que en este trabajo no se han considerado pero que podrían resultar interesantes en futuras ampliaciones del mismo. Una de las virtudes de este tipo de agentes de control basados en técnicas de aprendizaje reforzado es la capacidad de continuar con el aprendizaje durante su explotación. Así, mediante un aprendizaje *online* sería posible detectar posibles desviaciones en los parámetros del sistema y, por tanto, adaptarse a ellas. Por otro lado, puede resultar de utilidad realizar un análisis de sensibilidad ante los diferentes hiperparámetros utilizados en el algoritmo. Con ello, se podría afinar el proceso de aprendizaje del agente y mejorar el desempeño del control del sistema en bucle cerrado.

5.2.1 Comparación: DRL vs. MPC.

Finalmente, se presenta una comparación de los resultados conseguidos mediante el agente de control basado en aprendizaje reforzado y el controlador predictivo MPC utilizado como referencia. En la Figura 5.5 y Figura 5.6 se observa la evolución del sistema ante dos perfiles de perturbaciones diferentes. En estas gráficas, se representa la evolución tanto de la variable controlada (temperatura de salida y/o potencia térmica) como de la acción de control utilizada para ello.

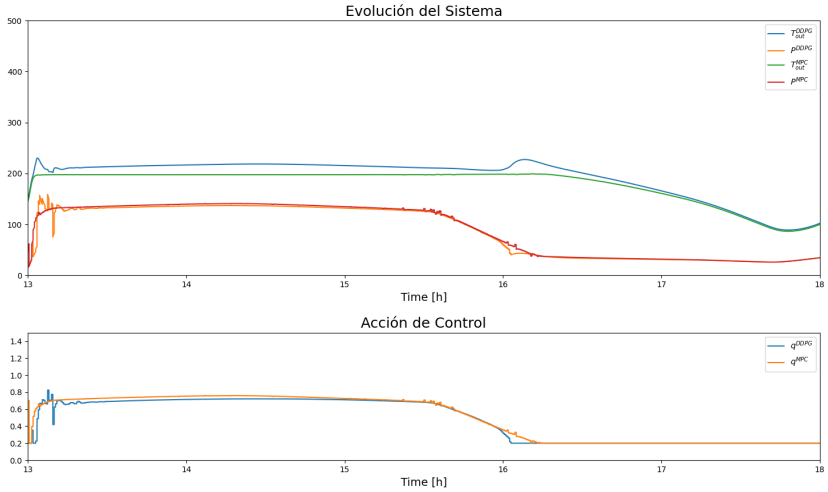


Figura 5.5 Comparación de la evolución del sistema controlado por DRL y MPC (Perfil 1).

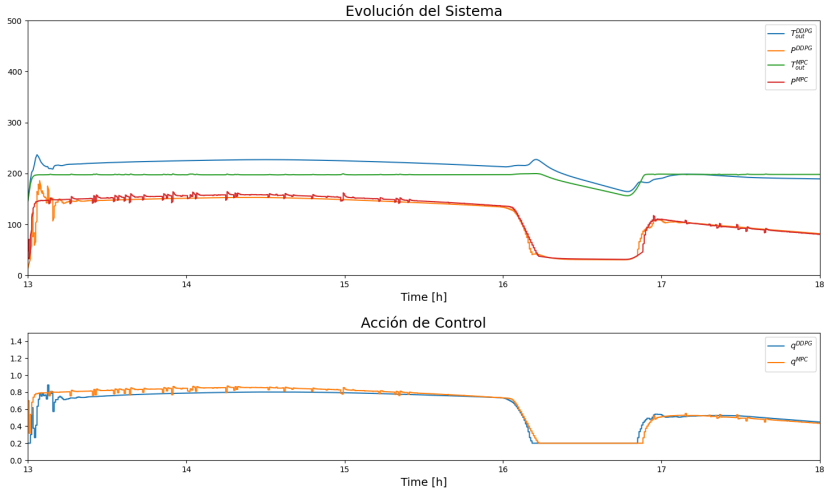


Figura 5.6 Comparación de la evolución del sistema controlado por DRL y MPC (Perfil 2).

Se observa, mediante estas gráficas, que el control ofrecido por ambos métodos, DRL y MPC, es muy similar. No obstante, existen ligeras discrepancias entre las curvas observadas. Por un lado, se aprecia cierta diferencia en el arranque de sendos controladores. Así, mientras que el control DRL necesita de cierto periodo de inicialización, el controlador MPC es capaz de ofrecer un buen control casi desde el primer minuto. Sin embargo, cabe destacar la suavidad característica de las curvas ofrecidas por el agente DRL durante el resto de la simulación, mientras que el control MPC, para una misma función objetivo, presenta una ley de control algo más “ruidosa”. No hay que olvidar, aun así, que este fenómeno es fácilmente evitable mediante un ajuste más fino de la función de coste.

Asimismo, resulta interesante utilizar, como forma de cuantificar el desempeño del agente de DRL frente al mostrado por el MPC, diferentes métricas de evaluación:

- ***Score***.

  Dado que ambos métodos utilizan la misma función objetivo ya sea como función de coste (en el caso del controlador MPC) o como función de recompensa (en el agente DRL), resulta interesante analizar el valor alcanzado por dicha función en ambos tipos de técnicas. Con esto, se establece una medida de la bondad del control ofrecido por cada uno de los controladores estudiados.

$$Score = \sum_{k=0}^{T_{sim}} \left(W_{th,k} - \psi \ \text{máx} \left(\frac{T_{out}(k) - T_{out}^{max}}{T_{out}^{max}}, \frac{T_{out}^{min} - T_{out}(k)}{T_{out}^{max}}, 0 \right)^2 - \varepsilon \ (q(k) - q(k-1))^2 \right) \quad (5.1)$$

- **Tiempo medio de ejecución**.

  Una de las ventajas ya mencionadas de las técnicas de DRL frente a controladores MPC es que los primeros permiten dar solución al problema de optimización en un tiempo mucho menor, siendo esto especialmente notable en aquellos problemas no lineales donde una solución exacta del MPC puede tardar hasta varios minutos. Se analiza, para reflejar esta virtud, el tiempo medio tomado por cada uno de los métodos para resolver el problema de control en cada instante de control.

- **Potencia térmica media**, $\overline{W_{th}}$.

  El problema de control planteado en este trabajo busca maximizar la potencia térmica neta generada por la instalación. Resulta de utilidad, por consiguiente, calcular el valor medio de la potencia alcanzada durante las diferentes simulaciones.

$$\overline{W_{th}} = \frac{1}{T_{sim}} \sum_{k=0}^{T_{sim}} (W_{out} - W_{in}) \quad (5.2)$$

- ***Slew rate*** **acumulado o** ***Accumulated Absolute Control Increment*** **(AACI)** [60].

  A pesar de que el objetivo principal del problema de control estudiado en este trabajo consiste en maximizar la potencia térmica generada, también resulta necesario respetar una serie de restricciones: evitar incrementos demasiado grandes en la acción de control y evitar que la temperatura del fluido caloportador se salga del rango de temperaturas de funcionamiento. Es interesante, por tanto, evaluar en qué medida se cumplen estas restricciones.

  Para comprobar si el controlador respeta la restricción referida a la acción de control, es conveniente analizar el *slew rate* acumulado durante toda una simulación. Esta métrica permite establecer una comparación relativa entre ambos métodos, reflejando, así, en qué medida afectan las penalizaciones por incrementos altos en la acción de control al problema. Este parámetro se define tal que:

$$AACI = \sum_{k} |q(k) - q(k-1)| \quad (5.3)$$

- **Violación de las restricciones o** ***Mean Squared Constraint Violation*** **(MSCV)** [60].

  Por último, se analizan las restricciones en las temperaturas alcanzadas por el sistema. El cumplimiento de esta restricción se puede medir a través de la diferencia acumulada entre la temperatura del fluido HTF a la salida del lazo de colectores y cada uno de los límites de su rango óptimo de funcionamiento. Para ello, se calcula el MSCV de la siguiente forma:

$$MSCV = \frac{1}{T_{sim}} \sum_{k} \text{máx}\left(T_{out}(k) - T_{out}^{max},\ T_{out}^{min} - T_{out}(k),\ 0\right)^2 \quad (5.4)$$

Con vistas a obtener unos resultados más representativos, estas métricas se calculan para un total de 20 simulaciones diferentes y, tras ello, se presenta el valor medio obtenido. No obstante, a fin de acercar dichas simulaciones al funcionamiento normal que se daría en la planta real, se omiten para el cálculo los primeros instantes de la simulación donde aún no se ha inicializado por completo el estado del agente de control.

En la Tabla 5.2 se presentan los valores obtenidos para las diferentes métricas recogidas anteriormente. Se han considerado, para ello, intervalos de simulación de cinco horas.

Tabla 5.2 Métricas de comparación entre controladores DRL y MPC.

Métrica	MPC	DRL	(DRL - MPC) / MPC
Score acumulado	563.204	563.248	$+0.01\,\%$
Tiempo de ejecución $[s]$	2.245	0.025	$-98.89\,\%$
$\overline{W_{th}}$ $[kW]$	111.244	109.466	$-1.60\,\%$
$AACI$ $[l/s]$	0.371	0.378	$+1.89\,\%$
$MSCV$ $[°C^2]$	61.352	45.113	$-26.47\,\%$

A partir de estas métricas, se puede concluir que, a pesar de que el agente de aprendizaje por refuerzo alcanza una potencia térmica media algo menor a la lograda por el controlador predictivo, el agente de control desarrollado en este trabajo supone una alternativa más que válida para el control predictivo en plantas termosolares. Así, se observa cómo el *score* acumulado (métrica utilizada como indicador de la calidad de la solución al problema de optimización) durante las diferentes simulaciones por ambos métodos resultan prácticamente idénticos.

Por otra parte, se demuestra que el aprendizaje reforzado verdaderamente aventaja a las técnicas de control predictivo en cuanto a velocidad de ejecución se refiere. Así, el controlador DRL permite ahorrar hasta casi un 99 % en tiempo de ejecución y, por tanto, en los recursos *hardware* utilizados. Esto supone un provecho en la explotación de este tipo de controladores ya que, a menudo, el tiempo tomado por los controladores predictivos no lineales hace inviable la utilización de estos para el control de la planta real. Bien es cierto, por el contrario, que el controlador MPC aquí utilizado ha sido programado en lenguaje Python y de manera artesanal, siendo posible encontrar alternativas más eficientes en otros lenguajes y librerías dedicadas. No obstante, este posible aumento de eficiencia no resulta suficiente como para invalidar los resultados aquí obtenidos.

Por último, en cuanto al cumplimiento de las restricciones resulta interesante hacer los siguientes comentarios:

- En primer lugar, se aprecian resultados muy similares en el *slew rate* acumulado durante la simulación. Estos datos reflejan la consecución de una respuesta mayoritariamente suave por parte de ambos controladores.
- En segundo llugar, sí se observan mayores diferencias en cuanto al MSCV. En este caso, el algoritmo DRL es capaz de ofrecer mejores resultados que el controlador MPC, llegando a acumular en torno a un 25 % menos de penalización. Este fenómeno en principio tan poco intuitivo, sin embargo, puede deberse a una mala sintonización de la función de coste del controlador MPC, ya que, a fin de establecer una comparación lo más precisa posible, en este trabajo se le han atribuido los pesos utilizados en el agente DRL en lugar de optimizarlos específicamente para el problema de control predictivo por MPC.

  Un último comentario interesante a realizar corresponde al rango de validez de esta última métrica. Así, según se define el MSCV, este tiene en cuenta en su cálculo aquellos instantes en los que es imposible realizar un buen control de la instalación. Esto, sin embargo, puede desvirtuar la métrica en función del perfil de irradiancia utilizado. Una posible solución a este problema podría ser, utilizando el controlador MPC como referencia de control "perfecto", no considerar para el cálculo aquellos instantes en los que este no es capaz de mantener la temperatura de salida del lazo en su rango óptimo ni aún con la acción de control saturada. En ese caso, los valores adoptados por la métrica corresponderían a $56.418°C^2$ y $34.617°C^2$ para el MPC y DRL, respectivamente.

En conclusión, se observa como el agente DRL propuesto en este trabajo no solo ofrece unos resultados comparables a los del controlador predictivo, sino que permite, además, reducir significativamente el tiempo de ejecución necesario para ello. Cabe remarcar, sin embargo, que, además del algoritmo de aprendizaje aquí desarrollado, en este trabajo se han puesto a prueba otros mecanismos de aprendizaje que no han ofrecido resultados tan satisfactorios como los aquí expuestos y que, por tanto, no han sido recogidos en este capítulo. Estos algoritmos se detallan en el Apéndice A.

6 Conclusión.

Now this is not the end. It is not even the beginning of the end.
But it is, perhaps, the end of the beginning.

WINSTON CHURCHILL

Este Trabajo Fin de Máster ha presentado y desarrollado una solución para el control continuo de plantas termosolares a través de un agente de control basado en técnicas de aprendizaje reforzado[1]. El objetivo perseguido mediante el diseño de este agente ha sido el de ofrecer una alternativa eficaz para el control predictivo en sistemas físicos, utilizando, como sistema de control de referencia, un controlador predictivo basado en modelo.

6.1 Conclusiones.

En primera instancia, y con vistas a una mejor compresión del algoritmo de control a desarrollar, se ha presentado el *Deep Reinforcement Learning* y los conceptos teóricos más utilizados en esta disciplina. Esta introducción da comienzo con una contextualización del RL y, tras un resumen de la definiciones matemáticas más relevantes, se desarrollan aquellas particularidades introducidas al incorporar al problema el enfoque propuesto por el aprendizaje profundo. Finalmente, esta introducción concluye con un análisis de la posición ocupada por el DRL frente a las técnicas de control predictivo MPC, desarrollando, para ello, los puntos fuertes y débiles de cada una de estas técnicas.

Se ha ofrecido, a continuación, una descripción sencilla del modelo dinámico de la planta termosolar utilizada como objeto de estudio en este trabajo: la antigua planta solar ACUREX situada en la Plataforma Solar de Almería. Así, a partir de un modelo simplificado de parámetros concentrados, se obtienen aquellas ecuaciones que permiten reflejar la evolución de las temperaturas a la salida de cada uno de los diferentes lazos de colectores en función de las perturbaciones incidentes sobre los mismos y del caudal de fluido caloportador que circula a través del sistema.

Una vez comprendido el comportamiento del sistema a controlar, se procede con el desarrollo del agente de control que da solución al problema planteado: un agente DDPG. Este agente de control se fundamenta en una arquitectura *actor-critic* en la que se propone un algoritmo de aprendizaje *model-free*, *off policy* y con un espacio de acción continuo. Asimismo, se adopta una formulación estándar del modelo del sistema anteriormente descrito que permite establecer una interfaz de comunicación entre agente y modelo, facilitando la interacción entre ambos.

Finalmente, los resultados derivados de los ensayos realizados con distintos perfiles de perturbaciones han demostrado que el agente de control desarrollado constituye una alternativa competitiva al control predictivo en el campo de aplicación de las plantas de generación termosolar. Así, a pesar de que el agente presentado

[1] Es posible encontrar el agente de control, así como todos los recursos adicionales utilizados durante su desarrollo, en el siguiente repositorio de GitHub: *https://github.com/fdoborrego/drl_acurex*.

tan solo representa una primera aproximación al controlador que finalmente se instalaría en la instalación real, el sistema de control conseguido es capaz de igualar e incluso, en ocasiones, mejorar la respuesta obtenida por un control predictivo basado en modelo. De esta forma, el aprendizaje reforzado ofrece una solución para el control predictivo de sistemas físicos mucho más rápida que su competidor a la vez que consigue respetar las restricciones existentes. En concreto, mientras que el controlador predictivo basado en modelo necesita de un tiempo medio de computación superior a los $2\,s$ para resolver el problema de optimización, el agente de control realiza la misma tarea en unos $0.25\,ms$. Esta reducción del tiempo de computación necesario permite al controlador calcular una acción de control en un tiempo menor al periodo de muestreo del sistema y facilita, con ello, su implementación en sistemas de tiempo real. Esto resulta especialmente interesante en plantas de gran tamaño, donde la presencia de un mayor número de lazo de colectores implicaría un tiempo de computación mucho mayor y, por tanto, la imposibilidad de utilizar controladores MPC. Finalmente, cabe destacar que esta implementación podría realizarse sin incurrir en una gran pérdida energética, ya que la diferencia observada en la potencia obtenida en ambos casos apenas alcanza un $1.5\,\%$.

6.2 Trabajos futuros.

No obstante, hay varios aspectos que no han sido abordados en el presente trabajo y se proponen, por tanto, como líneas de investigación futuras. Así, resultaría interesante realizar un estudio de la sensibilidad del proceso de aprendizaje del agente de control a los diferentes hiperparámetros existentes en el algoritmo. Un análisis pormenorizado de estos parámetros podría no solo acelerar el proceso de aprendizaje, sino permitir afinar los resultados obtenidos durante su evaluación. Asimismo, podrían desarrollarse algoritmos de optimización bayesiana que lleve a cabo una búsqueda del valor óptimo de cada uno de estos hiperparámetros utilizando una menor cantidad de recursos [86].

Pudiera ser de interés, además, estudiar diferentes descripciones del estado y función de recompensa utilizados en la descripción del escenario para evaluar posibles diferencias en la respuesta del sistema controlado. Así, podría resultar que una nueva función de recompensa menos densa fuera capaz de evitar las sobreoscilaciones observadas al producirse grandes cambios en la radiación solar incidente sobre los colectores, o que un estado que utilice diferentes tamaños de ventana temporal para la previsión de perturbaciones pudiera anteponerse mejor a las variaciones en las mismas. No se debe olvidar, tampoco, la posibilidad de desarrollar o bien nuevos agentes de control basados en otros algoritmos de aprendizaje, o bien nuevas arquitecturas de redes neuronales como, por ejemplo, las redes neuronales recurrentes[2].

Se deja como tema de investigación futura, también, el análisis de la adaptabilidad del agente a cambios en los parámetros del escenario. Como ya se ha indicado anteriormente, una de las fortalezas de este tipo de arquitecturas de control es la capacidad inherente de realizar un aprendizaje *online* y, por consiguiente, de adaptarse a variaciones en el sistema durante su operación.

Asimismo, sería interesante realizar una comparación entre el método de control propuesto en este trabajo y otras alternativas ya estudiadas en la literatura que buscan dar solución al excesivo coste computacional requerido por los controladores MPC. Así, podría analizarse, cualitativa y cuantitativamente, las ventajas y desventajas del *Deep Reinforcement Learning* frente a otros tipos de algoritmos de aprendizaje automático en los que diferentes redes neuronales aprenden, de forma *offline*, a partir de los datos recopilados durante el funcionamiento del controlador a sustituir [60].

Por último, surge, de manera natural, una posible extensión de este trabajo mediante la implementación de una descripción más compleja del escenario. Así, la descripción simplificada del modelo de parámetros concentrados en la que únicamente se utilizaba un lazo de colectores puede ser ampliada para estudiar la totalidad de la planta. De la misma forma, podría adoptarse el modelo de parámetros distribuidos para representar la dinámica del sistema con mayor fidelidad [14].

[2] Las redes neuronales recurrentes, o *Recurrent Neural Networks* (RNN), son un tipo de red neuronal capaz de procesar series temporales. Este tipo de redes permiten detectar relaciones temporales entre diferentes observaciones del estado de un escenario, pudiendo distinguir entre dependencias lejanas y cercanas en el tiempo. Se consiguen, así, predicciones futuras a partir de históricos pasados. Un ejemplo de este tipo de redes es la red *Long Short-Term Memory* (LSTM) [31].

Apéndice A
Otros algoritmos de aprendizaje.

En este apéndice del documento se recogen aquellos otros algoritmos de aprendizaje probados durante el desarrollo de este proyecto y que, dadas las condiciones bajo las cuales se han ensayado, no han ofrecido un control satisfactorio de la instalación.

A.1 Deep Q-Network.

Un enfoque práctico a la hora de utilizar el aprendizaje reforzado como algoritmo de control en sistemas con espacios de acción continuos es discretizar dichos espacios de acción y abordar, así, el problema con algoritmos más sencillos. Un algoritmo muy utilizado en este tipo de problemas es el *Deep Q-Network* (DQN).

El algoritmo de aprendizaje DQN corresponde al primer algoritmo de aprendizaje reforzado profundo capaz de desarrollar una ley de control en escenarios de altas dimensiones [46]. Este algoritmo consiste en una aplicación directa de los principios del *Deep Reinforcement Learning* desarrollados en la Sección 2.3. Así, el algoritmo DQN establece un método de resolución *model-free* y *off-policy* en el que se hace uso de la experiencia recogida de la interacción con el escenario para desarrollar una política de actuación óptima. Para ello, se define una función de aproximación para la función valor estado-acción $Q_\theta(s,a)$ que se actualiza de tal forma que minimice la función de pérdidas (2.31) utilizando el siguiente gradiente:

$$\nabla_\theta \mathcal{L}(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\left(r+\gamma \max_{a'} Q_{\theta^-}(s',a') - Q_\theta(s,a)\right)\nabla_\theta Q_\theta(s,a)\right] \tag{A.1}$$

En el Algoritmo 2 se resume, en forma de pseudocódigo, la secuencia seguida por el proceso de aprendizaje. En este algoritmo puede apreciarse la existencia de un *buffer* de experiencias que permita aleatorizar las interacciones recopiladas y evitar, así, la correlación existente entre muestras consecutivas. Además, es frecuente encontrar en la literatura que este método se acompañe con una política de acción ε-*greedy*, esto es, una política en la que existe una probabilidad $1-\varepsilon$ de que el agente de control tome la acción indicada por la estrategia $a = \max_{a_i} Q_\theta(s_i,a_i)$ y una probabilidad ε de que dicha acción se tome de forma aleatoria. Resulta muy común, con vistas a controlar el balance entre exploración y explotación durante el proceso de aprendizaje, establecer una evolución para este parámetro ε. Así, un mayor valor de ε permitirá una mayor exploración del espacio de acción, resultando idóneo para los primeros instantes de entrenamiento, mientras que un ε menor resulta más adecuado para los últimos estadios del entrenamiento, ya que permite una explotación más intensiva de los óptimos encontrados.

Finalmente, en la literatura es posible encontrar multitud de variantes de este método en las que se establecen diferentes arquitecturas de redes neuronales encargadas de predecir las funciones valor estado-acción en cuestión. Algunos ejemplos de estas se encuentran en los algoritmos *Double Deep Q-Network* (DDQN) [77] y *Dueling Deep Q-Network* (Dueling DQN) [80], los cuales, por un lado, utilizan una segunda función de aproximación para calcular el *target* de la función valor y, por otro, dividen la red neuronal en dos para estimar la función valor estado-acción a partir de la función valor estado y la función de ventaja por separado.

Algoritmo 2 Algoritmo DQN.

Inicialización aleatoria de la función de aproximación $Q_\theta(s,a)$.
Inicialización del *buffer* de experiencias.
for episodio $= 1$, $N_{EPISODES}$ **do**
    Inicialización del escenario: estado inicial s_1.
    for step $= 1$, T_{sim} **do**
        Cálculo de la acción a: $a \leftarrow \text{máx}_{a_j} Q_\theta(s,a_j)$.
        Ejecutar acción a.
        Leer nuevo estado alcanzado s' y recompensa r.
        Almacenar transición $e_t = (s,a,r,s')$ en el *buffer* de experiencias.
        Muestrear un *minibatch* del *buffer* de experiencias.
        Cálculo de valor estado-acción objetivo para cada muestra i:

$$y_i = r_i + \gamma \max_{a'_j} Q_\theta(s'_i, a'_j)$$

        Actualización de la red *critic* mediante la función de pérdidas de las N muestras:

$$\nabla_\theta \mathcal{L}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left[\left(r_i + \gamma \max_{a'_j} Q_{\theta^-}(s'_i, a'_j) - Q_\theta(s_i, a_i) \right) \nabla_\theta Q_\theta(s_i, a_i) \right]$$

    end for
end for

A.2 Soft Actor-Critic.

Un segundo enfoque consiste en desarrollar un agente de control que pueda trabajar directamente con un espacio de acción continuo. Un ejemplo de este tipo de algoritmos es el desarrollado en el Capítulo 4 en este trabajo. En la literatura, sin embargo, es posible identificar otras perspectivas para hacer frente a este planteamiento como, por ejemplo, la regularización de entropía.

El algoritmo *Soft Actor-Critic* (SAC) [28] consiste en un algoritmo de optimización *model-free* y *off-policy* que trata de establecer un equilibrio entre una política estocástica y otros enfoques como el establecido por el algoritmo DDPG donde no existe una exploración del escenario de forma intrínseca. Así, se introduce en el problema de optimización una medida de la aleatoriedad de la política de acción mediante la siguiente función objetivo:

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty} \gamma^t \ (R(S_t,A_t) + \beta H(\pi(\cdot|S_t)))] \tag{A.2}$$

donde $H(\pi) = \mathbb{E}_{a\sim\pi}[-log(\pi(a))]$ corresponde a la función de entropía y β es un coeficiente de ajuste que estable la importancia relativa entre la recompensa recibida y la aleatoriedad de la política de acción en su elección de la acción de control.

Desde este nuevo enfoque, se han de redefinir las funciones valor estado y estado-acción [53]:

$$\begin{cases} V_\pi(s) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty} \gamma^t \left(R(S_t,A_t) + \beta H(\pi(\cdot|S_t))\right) | S_0 = s] \\ Q_\pi(s,a) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty} \gamma^t R(S_t,A_t) + \beta \sum_{t=1}^{\infty} \gamma^t H(\pi(\cdot|S_t)) | S_0 = s, A_0 = a] \end{cases} \tag{A.3}$$

Asimismo, puede deducirse de las anteriores ecuaciones la relación entre sendas funciones valor estado y estado-acción:

$$V_\pi(s) = \mathbb{E}_{a\sim\pi}[Q_\pi(s,a)] + \beta H(\pi(\cdot|s)) \tag{A.4}$$

Y, finalmente, la ecuación de Bellman para $Q_\pi(s,a)$ resulta de la siguiente forma:

$$\begin{aligned} Q_\pi(s,a) &= \mathbb{E}_{\substack{s'\sim P \\ \tau\sim\pi}}[R(s,a)+\gamma(Q_\pi(s',a')+\beta H(\pi(\cdot|s)))] \\ &= \mathbb{E}_{\substack{s'\sim P \\ \tau\sim\pi}}[R(s,a)+\gamma V_\pi(s')] \end{aligned} \tag{A.5}$$

Bajo esta premisa, el algoritmo SAC propone un método de aprendizaje basado en tres funciones de aproximación: una política de acción π_{θ^π} y dos funciones valor estado-acción $Q_{\theta_1^Q}$ y $Q_{\theta_2^Q}$ (junto a sus respectivas funciones de target $Q_{\theta_1^{Q'}}$ y $Q_{\theta_2^{Q'}}$). De esta forma, se plantea un aprendizaje análogo al ya estudiado en el algoritmo DDPG, donde existe un papel de *actor* que desarrolla una política de actuación que dirige al agente a aquellos estados de mayor beneficio y otro papel de *critic* que evalúa la bondad de dichos estados alcanzados. En esta ocasión, sin embargo, existen dos funciones de estimación del valor estado-acción, lo cual permite hacer frente a la sobreestimación característica de estos algoritmos calculando el mínimo entre ambas aproximaciones.

Al igual que en los algoritmos anteriormente explicados, el proceso de aprendizaje tiene lugar a través de las respectivas funciones de pérdidas para cada una de las funciones de aproximación. Así, las funciones valor estado-acción se actualizan mediante el gradiente de la siguiente función:

$$\mathcal{L}(\theta_i^Q) = \mathbb{E}\left[\left(Q_{\theta_i^Q}(s,a) - y(r,s')\right)^2\right] \tag{A.6}$$

donde, para el cálculo del *target value* $y(r,s')$, se utiliza como aproximación del valor real de la función $Q(s,a)$ el mínimo entre ambas funciones de aproximación *target*:

$$\begin{cases} y = r + \gamma\left(\min\limits_{j=1,2} Q_{\theta_j^{Q'}}(s',\tilde{a}') - \beta\log(\pi_{\theta^\pi}(\tilde{a}'|s'))\right) \\ \tilde{a}' \sim \pi_{\theta^\pi}(\cdot|s') \end{cases} \tag{A.7}$$

La política de actuación, por su parte, se optimiza de tal forma que maximice la función valor estado $V_\pi(s)$. Para ello, se reparametriza esta ley de acción utilizando una política determinista μ_{θ^μ} y una función de ruido.

$$\begin{cases} \tilde{a}_{\theta^\pi} = tanh\left(\mu_{\theta^\pi} + \sigma_{\theta^\pi} \odot \xi\right) \\ \xi \sim \mathcal{N}(0,I) \end{cases} \tag{A.8}$$

Esta reparametrización permite, finalmente, expresar la ecuación (A.4) de tal forma que se elimine la influencia de los parámetros θ^π en el cálculo de la esperanza matemática:

$$\begin{aligned} V_{\pi_{\theta^\pi}}(s) &= \mathbb{E}_{a\sim\pi}[Q_\pi(s,a)] + \beta H(\pi(\cdot|s)) \\ &= \mathbb{E}_{a\sim\pi}[Q_\pi(s,a) - \beta\log\pi_{\theta^\pi}(a|s)] \\ &= \mathbb{E}_{\xi\sim\mathcal{N}}[Q_\pi(s,\tilde{a}_{\theta^\pi}(s,\xi)) - \beta\log\pi_{\theta^\pi}(\tilde{a}_{\theta^\pi}(s,\xi)|s)] \end{aligned} \tag{A.9}$$

Finalmente, dada la relación anterior, se define una última función de pérdidas para la política de actuación sustituyendo la estimación de la función valor $Q_\pi(s,\tilde{a}_\theta(s,\xi))$ que, según el *actor*, se debería obtener por la aproximación del valor real obtenida por el componente *critic*: $\min_{i=1,2} Q_{\theta_i^{Q'}}$. Esto resulta:

$$\mathcal{L}(\theta^\pi) = \mathbb{E}_{\xi\sim\mathcal{N}}\left[\min_{i=1,2} Q_{\theta_i^{Q'}}(s,\tilde{a}_\theta(s,\xi)) - \beta\log\pi_{\theta^\pi}(\tilde{a}_{\theta^\pi}(s,\xi)|s)\right] \tag{A.10}$$

En el Algoritmo 3 se resume, en forma de pseudocódigo, la secuencia seguida por el proceso de aprendizaje.

Como último comentario, no se ha de olvidar que, aunque en este trabajo no ha sido posible obtener un agente capaz de realizar un correcto control del sistema a partir de estos algoritmos, una correcta sintonización de los mismos podría convertirlos en alternativas válidas al método de aprendizaje desarrollado en el Capítulo 4.

Algoritmo 3 Algoritmo SAC.

1: Inicialización aleatoria de las funciones de aproximación $\pi_{\theta^\pi}(s,a)$, $Q_{\theta_1^Q}(s,a)$ y $Q_{\theta_2^Q}(s,a)$.
2: Inicialización de las funciones de aproximación objetivo $Q_{\theta_1^{Q'}}(s,a)$ y $Q_{\theta_2^{Q'}}(s,a)$: $\theta_1^{Q'} \leftarrow \theta_1^Q$, $\theta_2^{Q'} \leftarrow \theta_2^Q$.
3: Inicialización del *buffer* de experiencias.
4: **for** episodio $= 1$, $N_{EPISODES}$ **do**
5: Inicialización del escenario: estado inicial s_1.
6: **for** step $= 1$, T_{sim} **do**
7: Cálculo de la acción a: $a \sim \pi_{\theta^\pi}(\cdot|s)$.
8: Ejecutar acción a.
9: Leer nuevo estado alcanzado s' y recompensa r.
10: Almacenar transición $e_t = (s,a,r,s')$ en el *buffer* de experiencias.
11: Muestrear un *minibatch* del *buffer* de experiencias.
12: Cálculo de valor estado-acción objetivo para cada muestra i:

$$y_i = r_i + \gamma \left(\min_{j=1,2} Q_{\theta_j^{Q'}}(s_i', \tilde{a}_i') - \beta \log(\pi_{\theta^\pi}(\tilde{a}_i'|s_i')) \right)$$

13: Actualización de las redes *critic* mediante la función de pérdidas de las N muestras:

$$\nabla_{\theta_j^Q} \mathcal{L}(\theta_j^Q) \approx \nabla_{\theta_j^Q} \frac{1}{N} \sum_{i=1}^{N} \left(y_i - Q_{\theta_j^Q}(s_i, a_i) \right)^2$$

14: Actualización de la red *actor*:

$$\nabla_{\theta^\pi} \mathcal{L}(\theta^\pi) \approx \nabla_{\theta^\pi} \frac{1}{N} \sum_{i=1}^{N} \left(\min_{j=1,2} Q_{\theta_j^{Q'}}(s_i, \tilde{a}_{\theta^\pi}(s_i)) - \beta \log \pi_{\theta^\pi}(\tilde{a}_{\theta^\pi}(s_i)|s_i) \right)$$

15: Actualización de las funciones de aproximación objetivo (redes) $Q_{\theta_1^{Q'}}(s,a)$ y $Q_{\theta_2^{Q'}}(s,a)$:

$$\theta_1^{Q'} \leftarrow \zeta\, \theta_1^Q + (1-\zeta)\theta_1^{Q'}$$
$$\theta_2^{Q'} \leftarrow \zeta\, \theta_2^Q + (1-\zeta)\theta_2^{Q'}$$

16: **end for**
17: **end for**

Índice de Figuras

Índice de Tablas

Bibliografía

[1] Abbass, K., Qasim, M. Z., Song, H., Murshed, M., Mahmood, H., and Younis, I. A review of the global climate change impacts, adaptation, and sustainable mitigation measures. *Environmental Science and Pollution Research* (2022), 1–21.

[2] Adam, S. P., Alexandropoulos, S.-A. N., Pardalos, P. M., and Vrahatis, M. N. No free lunch theorem: A review. *Approximation and optimization* (2019), 57–82.

[3] Al-Naima, F., and Abdul Majeed, B. Spline-based formulas for the determination of equation of time and declination angle. *International Scholarly Research Notices 2011* (2011).

[4] Ali, R., Chuah, J. H., Talip, M. S. A., Mokhtar, N., and Shoaib, M. A. Structural crack detection using deep convolutional neural networks. *Automation in Construction 133* (2022), 103989.

[5] Allugunti, V. R. Breast cancer detection based on thermographic images using machine learning and deep learning algorithms. *International Journal of Engineering in Computer Science 4*, 1 (2022), 49–56.

[6] Arroyo, J., Manna, C., Spiessens, F., and Helsen, L. Reinforced model predictive control (rl-mpc) for building energy management. *Applied Energy 309* (2022), 118346.

[7] Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[8] Berenguel, M., Camacho, E. F., and Rubio, F. R. *Control of solar energy systems*. Springer, 2012.

[9] Bithas, K., and Kalimeris, P. A brief history of energy use in human societies. In *Revisiting the energy-development link*. Springer, 2016, pp. 5–10.

[10] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).

[11] Camacho, E., Rubio, F., Berenguel, M., and Valenzuela, L. A survey on control schemes for distributed solar collector fields. part i: Modeling and basic control approaches. *Solar Energy 81*, 10 (2007), 1240–1251.

[12] Camacho, E., Rubio, F., Berenguel, M., and Valenzuela, L. A survey on control schemes for distributed solar collector fields. part ii: Advanced control approaches. *Solar Energy 81*, 10 (2007), 1252–1272.

[13] Camacho, E. F., and Alba, C. B. *Model predictive control*. Springer science & business media, 2013.

[14] Camacho, E. F., Berenguel, M., and Rubio, F. R. *Advanced control of solar plants*. Springer, 1997.

[15] Camacho, E. F., Rubio, F., and Gutierrez, J. Modelling and simulation of a solar power plant with a distributed collectors system. In *Power Systems: Modelling and Control Applications*. Elsevier, 1989, pp. 321–325.

[16] Cardoso, A., Henriques, J., and Dourado, A. Fuzzy supervisor and feedforward control of a solar power plant using accessible disturbances. In *1999 European Control Conference (ECC)* (1999), IEEE, pp. 1711–1716.

[17] Chowdhury, M., Rahman, K. S., Selvanathan, V., Nuthammachot, N., Suklueng, M., Mostafaeipour, A., Habib, A., Akhtaruzzaman, M., Amin, N., and Techato, K. Current trends and prospects of tidal energy technology. *Environment, development and sustainability 23*, 6 (2021), 8179–8194.

[18] Chu, Y., and Meisen, P. Review and comparison of different solar energy technologies. *Global Energy Network Institute (GENI), San Diego, CA 1* (2011), 1–52.

[19] Contreras, R. C. *Análisis, modelado y control de un campo de colectores solares distribuidos con sistema de seguimiento en un eje*. PhD thesis, Universidad de Sevilla, 1986.

[20] Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems 2*, 4 (1989), 303–314.

[21] Elia, A., Kamidelivand, M., Rogan, F., and Gallachóir, B. Ó. Impacts of innovation on renewable energy technology cost reductions. *Renewable and Sustainable Energy Reviews 138* (2021), 110488.

[22] Esfahani, H. N., Kordabad, A. B., and Gros, S. Reinforcement learning based on mpc/mhe for unmodeled and partially observable dynamics. In *2021 American Control Conference (ACC)* (2021), IEEE, pp. 2121–2126.

[23] Eurostat. Energy statistics - an overview, 2022. Accessed 14 July 2022, *https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Energy_statistics_-_an_overview#Primary_energy_production*.

[24] Fraunhofer, I. Study: Levelized cost of electricity—renewable energy technologies. *Fraunhofer ISE: Freiburg, Germany* (2021).

[25] Frejo, J. R. D., and Camacho, E. F. Centralized and distributed model predictive control for the maximization of the thermal power of solar parabolic-trough plants. *Solar Energy 204* (2020), 190–199.

[26] Gulzar, O., Qayoum, A., and Gupta, R. Photo-thermal characteristics of hybrid nanofluids based on therminol-55 oil for concentrating solar collectors. *Applied Nanoscience 9*, 5 (2019), 1133–1143.

[27] Guo, L.-N., She, C., Kong, D.-B., Yan, S.-L., Xu, Y.-P., Khayatnezhad, M., and Gholinia, F. Prediction of the effects of climate change on hydroelectric generation, electricity demand, and emissions of greenhouse gases under climatic scenarios and optimized ann model. *Energy Reports 7* (2021), 5431–5445.

[28] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning* (2018), PMLR, pp. 1861–1870.

[29] Hahnloser, R., Sarpeshkar, R., Mahowald, M., Douglas, R. J., Seung, S., et al. Digital selection and analog amplification co-exist in an electronic circuit inspired by neocortex. *Nature 405*, 6789 (2000), 947–951.

[30] He, W., Zhang, L., and Yuan, C. Future air temperature projection in high-density tropical cities based on global climate change and urbanization–a study in singapore. *Urban Climate 42* (2022), 101115.

[31] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[32] Huang, T., and Smith, C. 'artificial' intelligence. In *The History of Artificial Intelligence* (2006), University of Washington.

[33] Hwangbo, J., Sa, I., Siegwart, R., and Hutter, M. Control of a quadrotor with reinforcement learning. *IEEE Robotics and Automation Letters 2*, 4 (2017), 2096–2103.

[34] Jiang, Z. Installation of offshore wind turbines: A technical review. *Renewable and Sustainable Energy Reviews 139* (2021), 110576.

[35] Ju, X., Xu, C., Hu, Y., Han, X., Wei, G., and Du, X. A review on the development of photovoltaic/-concentrated solar power (pv-csp) hybrid systems. *Solar Energy Materials and Solar Cells 161* (2017), 305–327.

[36] Karthikeyan, A., and Priyakumar, U. Artificial intelligence: machine learning for chemical sciences. *Journal of Chemical Sciences 134*, 1 (2022), 1–20.

[37] Kashyap, A. Artificial intelligence medical diagnosis. *Scholars Journal of Applied Medical Sciences (SJAMS) 6* (2018), 4982–4985.

[38] Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836* (2016).

[39] Khan, S. G., Herrmann, G., Lewis, F. L., Pipe, T., and Melhuish, C. Reinforcement learning and optimal adaptive control: An overview and implementation examples. *Annual reviews in control 36*, 1 (2012), 42–59.

[40] Len, A. J. G., del Pozo, A. J. S., and Camacho, E. F. Aplicaciones de control predictivo en plantas solares ccp. *Revista Iberoamericana de Automática e Informática Industrial 19*, 3 (2022), 309–317.

[41] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[42] Ma, Y., Zhu, W., Benton, M. G., and Romagnoli, J. Continuous control of a polymerization system with deep reinforcement learning. *Journal of Process Control 75* (2019), 40–47.

[43] Masero, E., Maestre, J. M., and Camacho, E. F. Market-based clustering of model predictive controllers for maximizing collected energy by parabolic-trough solar collector fields. *Applied Energy 306* (2022), 117936.

[44] McCulloch, W. S., and Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics 5*, 4 (1943), 115–133.

[45] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning* (2016), PMLR, pp. 1928–1937.

[46] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[47] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature 518*, 7540 (2015), 529–533.

[48] Olabi, A., and Abdelkareem, M. A. Renewable energy and climate change. *Renewable and Sustainable Energy Reviews 158* (2022), 112111.

[49] OpenAI. Extra material: Proof for using q-function in policy gradient formula - spinning up, 2022. Accessed 24 September 2022, *https://spinningup.openai.com/en/latest/spinningup/extra_pg_proof2.html*.

[50] OpenAI. Gym documentation, 2022. Accessed 14 September 2022, *https://www.gymlibrary.dev/*.

[51] OpenAI. Part 1: Key concepts in rl, 2022. Accessed 24 September 2022, *https://spinningup.openai.com/en/latest/spinningup/rl_intro.html*.

[52] OpenAI. Part 3: Intro to policy optimization - spinning up, 2022. Accessed 24 September 2022, *https://spinningup.openai.com/en/latest/spinningup/rl_intro3.html*.

[53] OpenAI. Soft actor-critic, 2022. Accessed 2 November 2022, *https://spinningup.openai.com/en/latest/algorithms/sac.html*.

[54] Pereira, T. D., Tabris, N., Matsliah, A., Turner, D. M., Li, J., Ravindranath, S., Papadoyannis, E. S., Normand, E., Deutsch, D. S., Wang, Z. Y., et al. Sleap: A deep learning system for multi-animal pose tracking. *Nature methods 19*, 4 (2022), 486–495.

[55] Philipps, S. P., Bett, A. W., Horowitz, K., and Kurtz, S. Current status of concentrator photovoltaic (cpv) technology. Tech. rep., National Renewable Energy Lab.(NREL), Golden, CO (United States), 2015.

[56] Puterman, M. L. Markov decision processes. *Handbooks in operations research and management science 2* (1990), 331–434.

[57] Ramachandran, P., Zoph, B., and Le, Q. V. Searching for activation functions. *arXiv preprint arXiv:1710.05941* (2017).

[58] Rao, N., Aljalbout, E., Sauer, A., and Haddadin, S. How to make deep rl work in practice. *arXiv preprint arXiv:2010.13083* (2020).

[59] REE. El sistema eléctrico español: Informe resumen de energías renovables de 2021, 2021. Accessed 17 July 2022, *https://www.sistemaelectrico-ree.es/informe-de-energias-renovables*.

[60] Ruiz-Moreno, S., Frejo, J. R. D., and Camacho, E. F. Model predictive control based on deep learning for solar parabolic-trough plants. *Renewable Energy 180* (2021), 193–202.

[61] Sarker, I. H. Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science 2*, 3 (2021), 1–21.

[62] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[63] Seetharaman, P., Wichern, G., Pardo, B., and Le Roux, J. Autoclip: Adaptive gradient clipping for source separation networks. In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)* (2020), IEEE, pp. 1–6.

[64] Şen, Z. Solar energy in progress and future research trends. *Progress in energy and combustion science 30*, 4 (2004), 367–416.

[65] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science 362*, 6419 (2018), 1140–1144.

[66] Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. Deterministic policy gradient algorithms. In *International conference on machine learning* (2014), PMLR, pp. 387–395.

[67] Sørensen, B. A history of renewable energy technology. *Energy policy 19*, 1 (1991), 8–12.

[68] Spencer, J. Fourier series reprensentation of the position of the sun. *Search 2*, 5 (1971), 172.

[69] Spielberg, S., Tulsyan, A., Lawrence, N. P., Loewen, P. D., and Gopaluni, R. B. Deep reinforcement learning for process control: A primer for beginners. *arXiv preprint arXiv:2004.05490* (2020).

[70] Sutton, R. S., and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

[71] Syafiie, S., Tadeo, F., and Martinez, E. Model-free learning control of neutralization processes using reinforcement learning. *Engineering Applications of Artificial Intelligence 20*, 6 (2007), 767–782.

[72] Talukder, B., Ganguli, N., Matthew, R., Hipel, K. W., Orbinski, J., et al. Climate change-accelerated ocean biodiversity loss & associated planetary health impacts. *The Journal of Climate Change and Health* (2022), 100114.

[73] Tan, K. M., Babu, T. S., Ramachandaramurthy, V. K., Kasinathan, P., Solanki, S. G., and Raveendran, S. K. Empowering smart grid: A comprehensive review of energy storage technology and application with renewable energy integration. *Journal of Energy Storage 39* (2021), 102591.

[74] Tesauro, G., et al. Temporal difference learning and td-gammon. *Communications of the ACM 38*, 3 (1995), 58–68.

[75] Tulbure, A.-A., Tulbure, A.-A., and Dulf, E.-H. A review on modern defect detection models using dcnns–deep convolutional neural networks. *Journal of Advanced Research 35* (2022), 33–48.

[76] Turing, A. M., and Haugeland, J. Computing machinery and intelligence. *The Turing Test: Verbal Behavior as the Hallmark of Intelligence* (1950), 29–56.

[77] Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence* (2016), vol. 30.

[78] Velasco, J. G. *Energías renovables*. Reverte, 2009.

[79] Walker, J. S., Kopp, R. E., Little, C. M., and Horton, B. P. Timing of emergence of modern rates of sea-level rise by 1863. *Nature communications 13*, 1 (2022), 1–8.

[80] Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning* (2016), PMLR, pp. 1995–2003.

[81] Watson, J. B. Psychology as the behaviorist views it. *Psychological review 20*, 2 (1913), 158.

[82] Weng, L. A (long) peek into reinforcement learning. *lilianweng.github.io* (2018). Accessed 24 September 2022, *https://lilianweng.github.io/posts/2018-02-19-rl-overview/*.

[83] Wikipedia contributors. Law of total expectation — Wikipedia, the free encyclopedia, 2022. Accessed 24 September 2022, *https://en.wikipedia.org/w/index.php?title=Law_of_total_expectation&oldid=1093873409*.

[84] Wikipedia contributors. Markov decision process — Wikipedia, the free encyclopedia, 2022. Accessed 22 September 2022, *https://en.wikipedia.org/w/index.php?title=Markov_decision_process&oldid=1106395391*.

[85] Wikipedia contributors. Ornstein–uhlenbeck process — Wikipedia, the free encyclopedia, 2022. Accessed 26 September 2022, *https://en.wikipedia.org/w/index.php?title=Ornstein%E2%80%93Uhlenbeck_process&oldid=1090237106*.

[86] Wu, J., Chen, X.-Y., Zhang, H., Xiong, L.-D., Lei, H., and Deng, S.-H. Hyperparameter optimization for machine learning models based on bayesian optimization. *Journal of Electronic Science and Technology 17*, 1 (2019), 26–40.

[87] Yebra, L., Berenguel, M., Bonilla, J., Roca, L., Dormido, S., and Zarza, E. Object-oriented modelling and simulation of acurex solar thermal power plant. *Mathematical and Computer Modelling of Dynamical Systems 16*, 3 (2010), 211–224.

[88] Zanon, M., and Gros, S. Safe reinforcement learning using robust mpc. *IEEE Transactions on Automatic Control 66*, 8 (2020), 3638–3652.

[89] Zanon, M., Gros, S., and Bemporad, A. Practical reinforcement learning of stabilizing economic mpc. In *2019 18th European Control Conference (ECC)* (2019), IEEE, pp. 2258–2263.

[90] Zhang, H., Baeyens, J., Degrève, J., and Cacères, G. Concentrated solar power plants: Review and design methodology. *Renewable and sustainable energy reviews 22* (2013), 466–481.

[91] Zhao, D., Dai, S., Li, M., Wu, Y., Zheng, L., Wang, Y., and Yun, D.-Q. Improved efficiency and stability of perovskite solar cells using a difluorobenzothiadiazole-based interfacial material. *ACS Applied Energy Materials 4*, 10 (2021), 10646–10655.

Glosario

AACI *Accumulated Absolute Control Increment*.

AI *Artificial Intelligence*.

API *Application Programming Interface*.

CIEMAT Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas.

CSP *Concentrated Solar Power*.

DCS *Distributed Collector System*.

DDPG *Deep Deterministic Policy Gradient*.

DL *Deep Learning*.

DNI *Direct Normal Irradiance*.

DQN *Deep Q-Network*.

DRL *Deep Reinforcement Learning*.

HTF *Heat Transfer Fluid*.

LFR *Linear Fresnel Reflector*.

LSTM *Long Short-Term Memory*.

MDP *Markov Decision Process*.

ML *Machine Learning*.

MPC *Model Predictive Control*.

MSCV *Mean Squared Constraint Violation*.

PDS *Parabolic Dish System*.

PSA Plataforma Solar de Almería.

PTC *Parabolic Trough Collector*.

PV *Photovoltaic*.

RL *Reinforcement Learning*.

RNN *Recurrent Neural Networks*.

SAC *Soft Actor-Critic*.

SPT *Solar Power Tower*.

TES *Thermal Energy Storage*.