UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

APRENDIZAJE REFORZADO FUERA DE LINEA EN CONTROL NO LINEAL DE ESTANQUES

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

JORGE ANDRES FABRY GIL

PROFESOR GUÍA:
MARCOS ORCHARD CONCHA

MIEMBROS DE LA COMISIÓN:
DORIS SÁEZ HUEICHAPAN
JUAN YUZ EISSMANN

Este trabajo ha sido parcialmente financiado por:
FONDECYT 1210031

SANTIAGO DE CHILE
2022

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA Y MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: JORGE ANDRES FABRY GIL
FECHA: 2022
PROF. GUÍA: MARCOS ORCHARD

APRENDIZAJE REFORZADO FUERA DE LINEA EN CONTROL NO LINEAL DE ESTANQUES

En la presente tesis se estudia el uso de Random Ensemble Mixture de Deep Q-Network (REM), método de Reinforcement Lerning (RL) Off-policy en un entorno Offline. El aporte de esta tesis está en demostrar la factibilidad de utilizar esta metodología sobre bases de datos construidas por controladores aleatorios sobre una planta no lineal, mejorando la recompensa promedio. Previamente, se han utilizado algoritmos Off-policy para recorrer de forma efectiva la cadena de decisión de Markov.

Para mejorar la generalización fuera de línea se utiliza REM, un robusto algoritmo de Q-learning (aprendizaje de la función de recompensa Q) el cual impone consistencia de las ecuaciones de Bellman en combinaciones convexas aleatorias de múltiples estimadores de la función Q.

Como ambiente se utiliza un estanque cónico, representado en un ambiente virtual GYM (ambiente clásico para probar algoritmos de RL) y simulaciones en Python. Creando con estos tres bases de datos, dos generadas a partir de agentes DQN interactuando con el ambiente y una generada mediante el control de agentes controladores Proporcional, Integral y Derivativo (PID) sobre las simulaciones en Python. Con estas bases de datos se entrenan agentes, observando la capacidad de obtener mejores políticas finales y robustez frente a desperfectos y perturbaciones.

Agradecimientos

Agradecimientos al laboratorio de electromovilidad y en especial al grupo de reuniones semanales con problemáticas mineras. El trabajo de esta investigación de desarrollo de gran manera gracias a sus aportes y iteraciones semanales. En especial al Profesor Marcos por su constante apoyo, ayuda y paciencia.

Agradezco a mis compañeros de universidad y a todos los que me acompañaron durante mi proceso universitario. Agradezco a Jou Hui por ayudarme a creer en mi mismo. A Martín por su compañía incondicional. A Pablo y Jose Domingo por motivarme a seguir adelante. A Barbi por quererme mucho desde el día 1 en Beauchef. A Lenny por su compañía y alegría.

Agradezco a mis amigos y familiares que ayudaron a formar a la persona que soy hoy. A mis padres por apoyarme en mis procesos. A mis hermanos por todos los momentos que compartimos. A Pancho, Matías, Matías y Pascualito por compartir conmigo los viajes que me forjaron como persona. A mi familia por todos los momentos compartidos.

Por último, agradezco a María Paz y su familia, por acogerme y darme espacio para terminar esta tesis en su casa.

Tabla de Contenido

1. Introducción

Durante la última década el aprendizaje de máquinas ha tenido un boom importante, tanto en la investigación como su aplicación en la industria. Como prueba de ello es posible ver el número de publicaciones anuales en *Conference on Neural Information Processing Systems* abreviado como **NeurIPS**, una de las conferencias de aprendizaje de máquinas e inteligencia artificial, donde cada año se reporta un crecimiento cercano a un 40 % de artículos enviados y publicados[1][2][3][4][5]. La Figura 1.1 muestra lo anterior en términos de los artículos enviados y aceptados. En cuanto a la industria, el aprendizaje de máquinas ha tenido un crecimiento exponencial, alcanzando 2.9 mil millones de dólares en 2019 y 4 mil millones de dólares en 2020 solo en Estados Unidos (alcanzando 11.3 mil millones de dólares a nivel mundial), y se proyecta que continúe con esta tendencia [6].

Figura 1.1: Publicaciones anuales en **NeurIPS**

El poder de computación de propósito general en Unidades de Procesamiento Gráfico (GPUs) ha sido utilizada extensamente en el desarrollo e investigación de redes neuronales. Una de las razones por el crecimiento del aprendizaje profundo ha sido atribuida al avance en las capacidades de las GPUs [7]. Se ha observado un aumento en la capacidad de cómputo exponencial [8] y un aumento en la complejidad de las técnicas actuales, lo cual responde a la evolución de modelos sobre sus predecesores y a la necesidad de responder a problemáticas más específicas u obtener mejores métricas.

El aprendizaje de máquinas se divide principalmente en cuatro partes. Aprendizaje supervisado, no-supervisado, semi-supervisado y por refuerzo[9]. Los cuales mediante datos etiquetados (supervisado), no etiquetados (no-supervisado), ambos (semi-supervisado) e interacciones con el ambiente (por refuerzo) aprenden generalizaciones sobre los datos o sistemas. El aprendizaje reforzado es en particular interesante por entregar un formalismo matemático para el control basado en aprendizaje [10]. Utilizando algoritmos de esta área es posible adquirir automáticamente políticas de comportamiento cercanas al óptimo teórico. Una de las aristas que ha resultado interesante es la del aprendizaje reforzado fuera de línea, la cual responde, mediante la lógica de aprendizaje reforzado, a problemas cuyo ambiente no puede ser interactuado directamente, ya sea porque al hacer esto se incurre en un costo económico alto, como en la industria minera, o el hacerlo implique traspasar limites morales y éticos, como sucede en la industria médica y medioambiental.

Un número de trabajos recientes han ilustrado el poder del aprendizaje reforzado enfocado en datos para el aprendizaje de políticas para el diálogo [11], manipulación del comportamiento robótico [12][13] y habilidades de navegación robótica [14] entre otros. Los buenos resultados presentados por estos agentes en el control de sus respectivas áreas trae la duda de si este tipo de algoritmos se pueden utilizar en otros ámbitos donde se utiliza la misma configuración, donde existe un agente controlador, el cual tiene acceso o conocimiento del estado completo o parcial y se tiene claridad sobre la función de mérito a optimizar. Además de lo anteriormente mencionado, se ha mostrado un interés importante por estas técnicas durante el último año, a tal punto que se han creado librerías enfocadas completamente en el aprendizaje reforzado fuera de línea, como *d3rlpy* introducida en noviembre del año 2021.

En esta tesis se estudia la factibilidad de acercar esta metodología a control de procesos mineros. Como la estructura del problema mencionado anteriormente se puede aplicar a procesos de control, es factible pensar que estos se pueden aplicar a procesos mineros de recirculación. Por lo reciente de estos métodos y la rapidez con la cual se está trabajando, aumenta substancialmente la cantidad de áreas donde estas técnicas se han aplicado. En el área de la minería existían artículos que utilizaban métodos de aprendizaje reforzado fuera de línea basada en modelos [15]. Pero junto con la finalización de esta investigación se han utilizado metodologías de aprendizaje reforzado offline basada completamente en datos [16].

Se menciona la minería porque es un buen ejemplo de sistemas no lineales, controlados mediante el input visual de un operador experto, donde se incurre en un costo muy alto al dejar que un agente interactúe con el estado. De igual forma se podría tener la medicina, donde un doctor experto recomienda la ingesta de un medicamento o el procedimiento médico adecuado, para un sistema no lineal, complejo, basándose en sus observaciones y conocimientos, donde se incurre en un costo muy alto al dejar que el agente interactúe con la persona, debido a las complicaciones éticas de recomendar medicamentos erróneos para comprender de mejor manera la cadena de decisión de Markov.

1.1. Motivación

En el contexto de los crecientes avances en técnicas de procesamiento y análisis de datos, existe un notorio interés por tratar de determinar mejores prácticas de utilización de activos basándonos en el análisis de datos históricos, reconociendo que existen políticas de tomas de decisiones dispares, difíciles de identificar y analizar por separado.

La política elegida para controlar un sistema puede necesitar únicamente componentes observables del sistema, en este caso se habla de un proceso de decisión Markoviano (MDP) completamente-observado. El caso contrario, un proceso de decisión Markoviano parcialmente-observado, es el cual no tiene acceso a todo el estado del MDP. En la industria existen políticas, forma en como actúan los operadores, las cuales dependen de estimaciones para una correcta estimación de los puntos de funcionamiento óptimo. Si estas estimaciones no son suficientemente robustas, los operadores, en ocasiones, deciden desviarse de la política establecida con el fin de llegar a la producción diaria, como puede suceder en el caso de las operaciones mineras. Se quiere encontrar un modelo que entregue un *set point* robusto, que

entregue información importante a los operadores. Si se cuenta con datos sobre la operación del sistema, es posible identificar políticas de operación que proporcionen información adecuada de cómo se debe operar la planta para seguir el tratamiento necesario. Con esto se pueden utilizar métodos de aprendizaje reforzado offline para encontrar una política que mejore el desempeño actual de la planta.

La presente tesis logra realizar un control de seguimiento a las simulaciones propuestas en la sección 1.1 con data generada con controladores aleatorios, logrando robustez y una mejoría en la política (obteniendo mayor recompensa). Es interesante ya que anterior a estos resultados solo se había realizado construcción de datasets mediante agentes de aprendizaje reforzado online, los cuales tienden a tener un periodo de exploración. Esta exploración permite tener un mapeo adecuado de la cadena de Markov, ayudando fuertemente al entrenamiento del agente fuera de linea.

1.2. Definición del escenario a estudiar

Para corroborar la factibilidad de lo propuesto en un contexto en el cual es posible modelar el ambiente, se propone estudiar el control de altura de un estanque cónico. La elección de este se debe a su no linealidad, la facilidad en su modelación y la presencia de elementos similares al sistema que motiva este estudio. En términos generales, el sistema consta de un estanque cónico, el cual presenta un flujo de salida regulado por medio de una válvula. El flujo de salida se dirige a un estanque de recirculación, del cual se extrae el fluido a través de una bomba eléctrica, mediante un variador de frecuencia se controla el flujo de entrada al estanque cónico, de esta forma se completa el ciclo. En la Figura 1.2 se puede apreciar el sistema hidráulico [1]. Las variables controladoras son la frecuencia del actuador (bomba eléctrica) y el grado de apertura de la válvula de salida, la cual se mantiene con apertura

[1] Figura extraída del enunciado del primer laboratorio de control

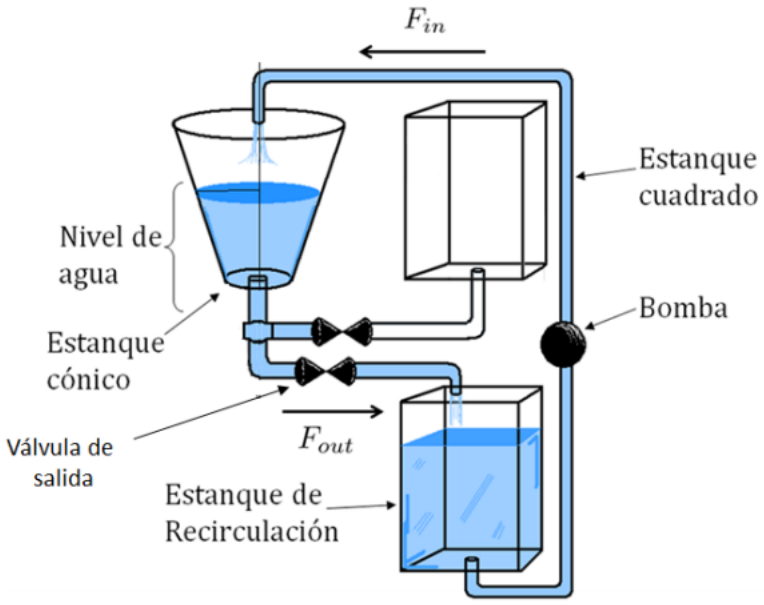


Figura 1.2: Estanque Cónico

constante. La variable a controlar es la altura del estanque.

1.3. Hipótesis de investigación

Es posible la utilización del algoritmo de aprendizaje fuera de línea Random Ensamble Mixture de Deep Q-Networks (REM DQN) para controlar el proceso dinámico, con tiempos de reacción significativos y no lineal de un estanque cónico. Sin necesidad de utilizar algoritmos de aprendizaje reforzado para generar el set de datos. Logrando un agente que mejore el control encontrado en los datos del dataset de entrenamiento. Un agente Deep Q-Network es un agente de aprendizaje reforzado el cual, mediante redes neuronales profundas, genera una función aproximadora del valor de recompensa Q. REM DQN un robusto algoritmo de Q-learning (aprendizaje de la función de recompensa Q) el cual impone consistencia de las ecuaciones de Bellman en combinaciones convexas aleatorias de múltiples estimadores de la función Q. El algoritmo mismo se verá con mayor detención en la sección de Marco Teórico.

1.4. Objetivo general

El objetivo general de esta investigación es realizar un control de seguimiento de la altura del estanque, mediante aprendizaje reforzado fuera de línea. Obteniendo una política que mejore la política de comportamiento (políticas presentes en el dataset).

1.5. Objetivo específicos

Como objetivos específicos se espera:

- Crear un ambiente virtual inspirado en el estanque cónico.
- Generar bases de datos mediante el entrenamiento de un agente Deep Q-Network **DQN** online en el estanque cónico.

- Entrenar agentes fuera de línea capaces de controlar el sistema mediante el uso de la información transicional del agente **DQN**.
- Generar bases de datos con controladores **PIDs** en el estanque cónico.
- Entrenar agentes fuera de línea capaces de controlar el sistema mediante el uso de la base de datos del controlador **PID**.

Las bases de datos creadas permiten obtener información del proceso, similar a la que se tendría mediante el control por parte de los operadores, sin necesidad de ser óptimo, pero que otorgue resultados favorables, mejorando el actuar pensando en optimizar la función de mérito. Esto se puede observar en menos recursos gastados, alejándose de una zona de operación problemática u obteniendo mayor *throughput*.

1.6. Metodología y herramientas

En una primera instancia se prueba el algoritmo sobre datos generados por el entrenamiento de un agente DQN sobre el ambiente virtual GYM del estanque cónico. De obtenerse resultados favorables sobre el ambiente virtual GYM se muestra que el código del algoritmo REM-DQN y el ambiente se codificaron de buena manera. De forma paralela se genera una base de datos con controladores PID aleatorios actuando sobre una simulación de Python. Luego se entrena un agente REM-DQN sobre la base de datos generada por los controladores PID y se observa la calidad del agente entrenado con data de controladores aleatorios.

En vistas del objetivo planteado, se propone utilizar *OpenAI GYM* para crear el ambiente virtual con el cual interactúa el agente **DQN** y el controlador **PID**. El código de los controladores se realiza en el lenguaje de programación python y las redes neuronales de **DQN** online y offline se realizan con *Pytorch*.

1.6.1. *OpenAI GYM*

OpenAI GYM es una herramienta para desarrollar y comparar algoritmos de aprendizaje reforzado. Soporta enseñarle a agentes desde caminar hasta jugar juegos tales como *Pong* o *Pinball*[17]. En términos sencillos, esta librería permite crear ambientes virtuales para ser utilizados con algoritmos de aprendizaje reforzado. Resulta interesante utilizar esta herramienta, ya que genera un ambiente compatible con varios algoritmos y es utilizada de forma transversal en todo proyecto de aprendizaje reforzado, si bien no es necesario utilizarla es poco común ver un proyecto de aprendizaje reforzado que no utilice esta herramienta. En esta investigación se utiliza para generar un ambiente personalizado para el estanque cónico.

1.6.2. *Pytorch*

Existen 3 grandes librerías con funcionalidades de aprendizaje profundo:

- textitCaffe y *Caffe2*: no tiene mucho más que las funcionalidades mínimas y la documentación no se encuentra estructurada intuitivamente. Es de difícil incorporación al repertorio de conocimientos y no posee cualidades que lo potencien como una librería interesante a explorar[18].
- *TensorFlow* (TF) y *Keras*: *TensorFlow* es una librería desarrollada por *Google Brain*, tiene buena tiene buena documentación y es posible realizar código personalizado con

facilidad. *Keras* se construye sobre *TF* como una versión de alto nivel, lo cual hace que sea más difícil implementar funciones personalizadas[18].

- *PyTorch* (*PT*): Librería desarrollada por el equipo de *Facebook*. Desarrollada más recientemente que *TF*, pero con una comunidad de rápido crecimiento. *PT* es dinámica y corre código de manera procedural, mientras que *TF* es necesario diseñar el modelo completo para iterarlo dentro de una *Sesión* [19].

Por lo anteriormente mencionado, *Caffe* y *Caffe2* no es recomendada. *Pytorch* junto a *TensorFlow* son las librerías de cálculo numérico predilectas para desarrollar aprendizaje de máquinas, principalmente por su facilidad de uso y su capacidad de ejecutar cálculos en GPU. Dependiendo del grupo investigativo se prefiere una librería sobre la otra, en ambas se puede realizar el mismo trabajo, la facilidad de utilización viene dada por el manejo que se tiene de esta y la arquitectura a replicar, dado que ambas tienen una selección extensa pero distinta de métodos soportados[18]. Mediante *Google Trends* se obtiene la figura 1.3 se puede observar la tendencia de interés entre las librerías, es posible observar que a partir del 2021 *PyTorch* comienza a ser más utilizado, mientras que *Caffe2* nunca fue muy interesante a nivel mundial. Tanto *Pytorch* como *TensorFlow* son librerías capaces de generar las redes

Figura 1.3: Interés comparativo entre las librerías de aprendizaje profundo en Google Trends.

capaces de llevar a cabo esta tesis. Se elige *Pytorch* por preferencia personal.

2. Marco teórico y estado del arte

En la presente sección se presenta el modelo fenomenológico del estanque utilizado para la modelación del estanque cónico, el control PID utilizado para generar datos de entrenamiento, una extensa explicación del aprendizaje reforzado en sus versiones online, fuera de póliza y fuera de línea. La explicación completa responde a que las distintas versiones se van construyendo una sobre otra con distintas variaciones. Por último se realiza una revisión del estado del arte en aprendizaje reforzado.

2.1. Modelo Fenomenológico del estanque

La bomba eléctrica puede modelarse como una ecuación de la recta como se muestra en la ecuación (2.1), el flujo de salida mediante la ecuación (2.2) y la variación de volumen se relaciona con ambos flujos anteriores mediante la ecuación (2.3) la cual responde a una ley de conservación.

$$F_{in}(t)\left[\frac{cm^3}{s}\right] = c_1 \cdot Frec[\,\%] + c_2 \tag{2.1}$$

$$F_{out}(t)\left[\frac{cm^3}{s}\right] = \beta \cdot \sqrt{h} \tag{2.2}$$

$$\frac{\delta V}{\delta t} = F_{in}(t) - F_{out}(t) \tag{2.3}$$

Mediante el teorema de Thales se establece una relación entre el radio y la altura, también es posible definir la constante α, la cual simplifica las ecuaciones:

$$r = \frac{h}{H}R,\ \alpha = \frac{H^2}{\pi R^2}$$

Luego, a partir del volumen de un cono, se puede relacionar la altura de agua con la frecuencia a través de la dinámica del sistema.

$$V(h) = \frac{\pi R^2 h^3}{3H^2} \tag{2.4}$$

$$\Rightarrow \frac{dV}{dt}(h, f) = \frac{\pi R^2 h^2}{H^2}\dot{h} = c_1 \cdot f + c_2 - \beta \cdot \sqrt{h} \tag{2.5}$$

$$\dot{h} = (c_1 \cdot f + c_2 - \beta \cdot \sqrt{h}) \frac{\alpha}{h^2} \tag{2.6}$$

2.1.1. Constantes del sistema

Las constantes c_1, c_2, β y α describen la dinámica del sistema, como se trata de un estanque imaginario se toman los valores presentados en la Tabla 2.1, pudiendo ser estos otros. Es importante mencionar que estas constantes dan un sistema físicamente plausible y coherente.

Tabla 2.1: Datos obtenidos de la prueba de llenado

c_1	c_2	β	α
3.9251	-117.753	11.1424	1.703

2.2. Control PID

El control proporcional, integral y derivativo (PID) es el mecanismo de control favorito de los procesos industriales por su simplicidad, efectividad a bajo costo y facilidad de implementación. Incluso con los métodos existentes y la data presente para alimentarlos más del 95 % de los bucles de control en la capa física son realizados por controladores PID[20][21][22][23].

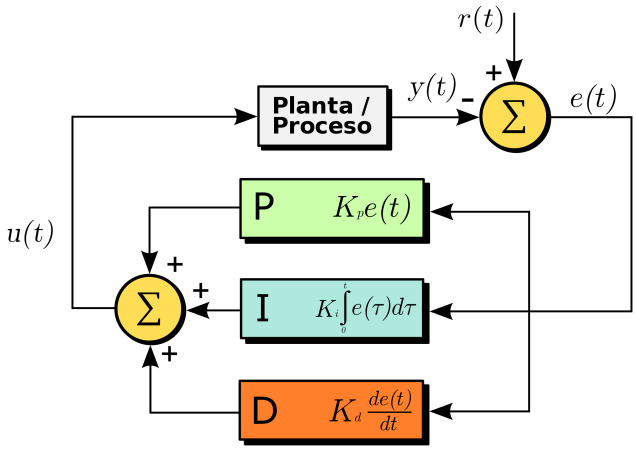


Figura 2.1: Lazo de control PID.

A grandes rasgos, el control PID toma el error, el error derivativo y el integral para retroalimentarlos a la planta multiplicada por una constante, con el fin de controlar el sistema, como muestra la imagen 2.1 [2]. La ecuación 2.7 describe el bucle de retroalimentación, donde

[2] Imagen cortesía de TravTigerEE, donada Wikimedia commons

K_p, K_i, K_d corresponden a las constantes proporcional, integral y derivativa.

$$\mathrm{u(t)} = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de}{dt} + u(0) \tag{2.7}$$

2.3. Aprendizaje reforzado

El aprendizaje reforzado toma inspiración en la corriente psicológica conductista, intenta que un agente (ente encargado de tomar las decisiones) interactúe con el ambiente y a partir de estas interacciones el agente recibe una retroalimentación, que le permite corregir o mejorar su actuar. La Figura 2.2 ejemplifica esta interacción. Entiéndase ambiente como entorno en el cual el agente puede tomar acciones, el cual se modifica mediante acciones y su evolución temporal, el agente puede observar el sistema mediante un vector de estado. Esta descomposición en términos matemáticos corresponde a un proceso de decisión Markoviano [24].

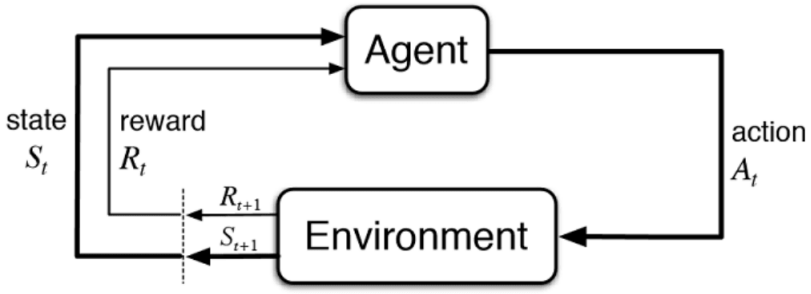


Figura 2.2: Interacción entre el agente y el ambiente [25]

2.3.1. Proceso de decisión Markoviano

El proceso de decisión Markoviano (PDM) se define como una tupla $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, d_0, r, \gamma)$[26] donde:

- $\mathcal{S}$: Es el set de estados $s \in \mathcal{S}$
- $\mathcal{A}$: Es el set de acciones $a \in \mathcal{A}$
- T: Define la distribución de probabilidad condicional que describe la dinámica del sistema de la forma $T(s_{t+1}|s_t, a_t)$
- d_0: Define el estado inicial de la distribución T
- r: Define la función de mérito, la cual entrega una recompensa para una pareja estado-acción, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
- γ: Factor de descuento $\gamma \in (0, 1]$ permite asignarle menor importancia a recompensas futuras, con la lógica de hacer un *trade-off* entre maximizar recompensa actual frente a las futuras.

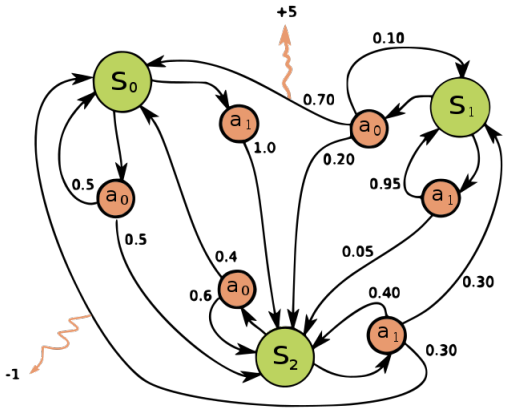


Figura 2.3: Proceso de decisión markoviano.

El agente interactúa con el ambiente mirando el estado s_t y tomando una acción a_t que lleva a un nuevo estado s_{t+1} con probabilidad $T(s_{t+1}|s_t, a_t)$, como se muestra en la Figura [3]2.3. El objetivo del aprendizaje reforzado es aprender una política que defina una distribución de acciones condicionadas a estados $\pi(a_t, s_t)$, con esta definición se puede derivar la distribución de la trayectoria. La trayectoria es una secuencia de acciones y estados de largo H, dada por $\tau = (s_0, a_0, ..., s_H, a_H)$ [27]. La distribución de la trayectoria p_π para un PDM $\mathcal{M}$ y una política π está dada por la ecuación 2.8

$$p_\pi(\tau) = d_o(s_0) \prod_{t=0}^{H} \pi(a_t|s_t) T(s_{t=1}|s_t, a_t) \tag{2.8}$$

Consecuentemente, la función objetivo $J(\pi)$ del aprendizaje reforzado puede ser escrita en su forma más amplia como la esperanza bajo la distribución de la trayectoria [28], presentada en la ecuación 2.9.

$$J(\pi) = \mathbb{E}_{\tau \sim p_\pi(\tau)} \left[\sum_{t=0}^{H} \gamma^t r(s_t, a_t) \right]. \tag{2.9}$$

2.3.2. Loop de aprendizaje

Los algoritmos de aprendizaje reforzado siguen un *loop* simple para aprender una política; el agente interactúa con el ambiente, definido por el PDM $\mathcal{M}$ mediante el uso de una política $\pi(a|s)$, la cual puede ser generada mediante exploración o utilizando la política actual, observando el estado s_t y seleccionando una acción a_t para luego observar el siguiente estado y la recompensa asociada $r_t + r(s_t, a_t)$. Esto se puede repetir por varias iteraciones para luego usar las transiciones observadas (s_t, a_t, s_{t+1}, r_t) para actualizar su política. Estas actualizaciones pueden usar transiciones anteriores, guardadas en un *Replay Buffer* ($\mathcal{D} = \{(s_t^i, a_t^i, s_{t+1}^i, r_t^i)\}$) [10].

[3] Imagen cortesía de Waldo Álvarez, donada a Wikimedia commons

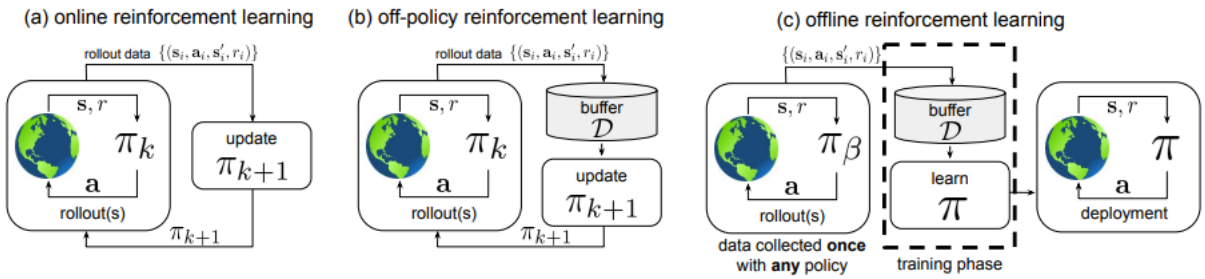


Figura 2.4: Ilustración del aprendizaje reforzado clásico (a), aprendizaje fuera de política clásico (b), y aprendizaje reforzado fuera de línea (c).[10]

2.3.3. Exploración ϵ-*greedy*

Algunos algoritmos requieren exploración para encontrar regiones del espacio de estado con alto retorno, para obtener una política óptima. Al realizar esta exploración en el espacio estado-acción se gana información de este, que a la larga mejora el desempeño del agente, pero se pierde la recompensa inmediata de realizar la mejor acción conocida [29].

ϵ-*greedy* es un método para balancear la exploración y explotación de la mejor política conocida. En las primeras iteraciones del algoritmo se conoce muy poco el espacio estado, además se espera que la política inicial sea ineficiente, por lo que resulta atractivo explorar. Luego de muchas iteraciones, no es probable generar una ganancia significativa por explorar un poco más, ya que se conoce el espacio estado relativamente bien, por lo que se quiere explorar poco y explotar la política, la cual debería tener retornos altos. ϵ demarca la probabilidad de explorar [29][30] y decae durante el entrenamiento hasta llegar a una cota inferior.

2.3.4. Algoritmos populares

En la mayoría de los problemas no es posible calcular de forma empírica el valor de la función de recompensa, en algunos algoritmos se construye una tabla que mapea la pareja acción-estado con la esperanza de la recompensa asociada, llamadas *Q-table*. Para problemas de alta dimensionalidad o alta granularidad no es práctico o hasta posible construir una tabla que sea capaz de realizar esta tarea. Existen diversas formas de atacar este problema, más adelante se abordará el algoritmo *Deep Q-network* (**DQN**), el cual realiza una aproximación a la función Q mediante una parametrización de esta en los pesos de una red neuronal profunda. En esta sección se presentan los algoritmos más comunes en el área de aprendizaje reforzado, explicando con mayor profundidad lo fundamental para esta investigación.

2.3.4.1. Gradiente de la política

La forma directa de optimizar la función objetivo de aprendizaje reforzado es estimando directamente el gradiente de esta. Típicamente, se asume que la política está parametrizada por un vector θ y, por lo tanto, está dada por $\pi_\theta(a_t|s_t)$, esta parametrización puede denotar los términos de una aproximación de taylor o pesos en una red neuronal [31][32][33]. Al calcular el gradiente de la ecuación 2.9 adoptando una formulación de horizonte infinito y

escribiéndolo con respecto a $d^{\pi}(s)$ resulta la ecuación 2.10.

$$\nabla_{\theta} J(\pi_{\theta}) = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\pi}(s_t), a \sim \pi_{\theta}(a|s)} \left[\nabla_{\theta} log \pi_{\theta}(a|s) \hat{A}(s,a) \right]. \tag{2.10}$$

Donde $\hat{A}(s,a)$ es un estimado del retorno centrado en la recompensa promedio. Con este gradiente se pueden actualizar los parámetros θ de la política π_{θ}, para llegar a una política cercana u óptima.

2.3.4.2. Aproximación mediante programación dinámica

La programación dinámica es un algoritmo que resuelve problemas combinando soluciones a subproblemas. Subdivide el problema general en subproblemas solapados, es decir, resolver el problema en tiempo t requiere la solución de $t-1$, el problema en $t-1$ requiere la solución en $t-2$ y así recursivamente hasta un origen [34]. El tipo de problemas clásicos que se atacan con programación dinámica tienen la forma $f(x_t) = g(x_t) + f(x_{t-1})$ más una condición inicial, donde f y g son funciones cualquiera.

Otra forma de optimizar la función objetivo es reconociendo que de ser posible estimar la recompensa en términos de la pareja estado-acción, es fácil recuperar una política cercana a la óptima. La función de recompensa entrega un estimado de la recompensa acumulada esperada al seguir una política $\pi(a_t|s_t)$ al partir de un estado-acción (s_t, a_t), esta función se define como se muestra en la ecuación 2.11, la cual puede ser escrita en su forma recursiva como se muestra en 2.12 y subsecuentemente esta puede ser escrita en términos del *operador de Bellman*[35] $\mathcal{B}^{\pi}$ para la una política π como $\overrightarrow{Q}^{\pi} = \mathcal{B}^{\pi} \overrightarrow{Q}^{\pi}$ [27], donde $\overrightarrow{Q}^{\pi}$ es una función Q^{π} representada por un vector de largo $|\mathcal{S}| \times |\mathcal{A}|$.

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{\tau \sim p_{\pi}(\tau|s_t, a_t)} \left[\sum_{t'=t}^{H} \gamma^{t'-t} r(s_t, a_t) \right]. \tag{2.11}$$

$$Q^{\pi}(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim T(s_{t+1}|s_t, a_t), a_{t+1} \sim \pi(a_{t+1}|s_{t+1})} \left[Q^{\pi}(s_{t+1}, a_{t+1}) \right]. \tag{2.12}$$

Para aprender esta función Q se reemplaza la política con términos implícitos de la función Q $(\pi(a_t|s_t) = \delta(a_t = arg\, max Q(s_t, a_t)))$ y esta se reemplaza en la ecuación 2.12, resultando en la ecuación 2.13, se puede expresar nuevamente como $\overrightarrow{Q} = \mathcal{B}^{*} \overrightarrow{Q}$ en notación vectorial, donde $\mathcal{B}^{*}$ es el operador de optimidad de Bellman. Para convertir esta ecuación en un algoritmo se puede minimizar la diferencia entre el lado izquierdo y derecho de la ecuación $Q_{\phi} - (r_t + \gamma \max_{a_{t+1}} Q_{\phi}(s_{t+1}, a_{t+1})$ con respecto a los parámetros de la función Q perimétrica $Q_{\phi}(s_{t+1}, a_{t+1})$ de parámetros ϕ [36].

$$Q^{*}(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim T(s_{t+1}|s_t, a_t)} \left[\max_{a_{t+1}} Q^{*}(s_{t+1}, a_{t+1}) \right]. \tag{2.13}$$

De esta formulación inicial se desprenden los algoritmos de programación dinámica, variando el tamaño del *Replay buffer*, la completitud de la minimización y el uso de otros algoritmos entre algunas variaciones.

2.3.4.3. Algoritmos de actor-critico

Los algoritmos de actor-crítico combinan las ideas básicas de los dos algoritmos comentados anteriormente, implementa la parametrización de la política y la función de valor, utilizando la función de valor para obtener un mejor estimado del retorno centrado en la recompensa promedio $\hat{A}(s,a)$ en el cálculo del gradiente de la política [10].

2.3.4.4. Aprendizaje reforzado basado en modelos

Aprendizaje reforzado basado en modelos es un término general para referirse a métodos que utilizan estimados explícitos de la transición o función de dinámica $T(s_{t+1}|s_t,a_t)$. Esta puede ser parametrizada mediante el vector de parámetros ψ quedando $T_\psi(s_{t+1}|s_t,a_t)$. No hay una receta única para este subconjunto de métodos [10]. Como ejemplo, algunos de los algoritmos de aprendizaje reforzado basado en modelos aprenden únicamente la dinámica del modelo $T_\psi(s_{t+1}|s_t,a_t)$ para luego utilizarlo para el testeo, usualmente mediante control predictivo basado en modelos (MPC) [37] con métodos de optimización de trayectoria variados [38][39].

2.4. Aprendizaje Reforzado fuera de la política

El aprendizaje reforzado fuera de política lleva su nombre debido a que la modificación de la política se lleva a cabo con un *Replay Buffer*, el cual contiene políticas transicionales entre la inicial y la actual. Como se observa en la Figura 2.4 (b). Esto permite puede escatimar costos y tiempo al no tener que usar la planta real para realizar en cada descenso de gradiente [40], no obstante, se continúa la interacción con el ambiente, incorporando las nuevas interacciones en el *Replay Buffer*.

2.4.1. Deep Q-Network

Q-learning es un algoritmo fuera de política, ya que el objetivo de aprendizaje puede ser computado sin saber la política de comportamiento con la cual se generaron los datos pi_β. En este método se construye una función acción-valor $Q^\pi(s,a)$ la cual permite conocer el valor de una acción dado el estado actual. Una forma de aproximar esta función es mediante *Deep Q-Network* DQN[41].

DQN es un algoritmo el cual parametriza la función Q_θ mediante una red neuronal convolucional y utiliza *Q-learning* con una red objetivo, utilizando una política con exploración *ϵ-greedy* para la recolección de datos [42]. DQN minimiza el error de diferencia temporal Δ_θ usando la perdida $\mathcal{L}(\theta)$ en *mini-batches* de tuplas representativas de experiencias pasadas (s,a,r,s') (de aquí en adelante se usará la simbología $x' = x_{t+1}$, $x = x_t$ intercambiablemente), muestreado de un *Replay Buffer* $\mathcal{D}$ recolectado durante el entrenamiento.

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,r,s'\sim\mathcal{D}}\left[l_\lambda(\Delta_\theta(s,a,r,s'))\right] \tag{2.14}$$

$$\Delta_\theta(s,a,r,s') = Q_\theta(s,a) - r - \gamma \max_{a'} Q_{\theta'}(s',a') \tag{2.15}$$

donde l_λ es la *Huber loss* dada por la ecuación 2.16.

$$l_\lambda(u) = \begin{cases} \frac{1}{2}u^2 & \text{si } |u| \leq \lambda \\ \lambda(|u| - \frac{1}{2}\lambda) & \text{en caso contrario.} \end{cases} \tag{2.16}$$

2.5. Aprendizaje Reforzado fuera de línea

El aprendizaje reforzado fuera de línea puede ser definido como la formulación basada en datos del problema de aprendizaje reforzado, tiene como objetivo optimizar la función 2.9 utilizando un *Replay Buffer* estático $\mathcal{D} = \{(s_t^i, a_t^i, s_{t+1}^i, r_t^i)\}$, encontrando la mejor política posible utilizando estos datos. La modalidad fuera de línea resulta interesante cuando las interacciones con el ambiente resultan costosas, peligrosas o poco éticas. Como puede resultar en aplicaciones mineras, donde un cambio a una política sub-óptima puede presentar una perdida monetaria importante, aplicaciones donde un robot puede maniobrar peligrosamente cerca de humanos o en aplicaciones médicas donde él entrega medicamentos para obtener información de cómo reacciona un paciente no es ético. Esta modalidad se asemeja más a la formulación típica de aprendizaje supervisado, con la diferencia que el aprendizaje supervisado intenta imitar el comportamiento que presentan los datos, mientras que el aprendizaje reforzado offline intenta responder preguntas contrafactuales, que son fundamentalmente preguntas del tipo "¿qué pasaría si?", se intenta buscar una mejor forma de iterar a la estipulada por la política de comportamiento presente en los datos[10]. La diferencia entre fuera de línea y fuera de política es que la primera no tiene interacción con el ambiente durante el entrenamiento. Es importante tener contacto con el ambiente debido a varios factores que influencian el desempeño del algoritmo, estos son principalmente el fenómeno conocido como *distributional shift* o movimiento de la distribución y la incapacidad de explorar. Como se observa en la Figura 2.4 (c).

2.5.1. Incapacidad de explorar

Al no tener contacto con el ambiente el algoritmo no puede descubrir regiones del espacio de estado de alta recompensa si transiciones hacia estas zonas no están representadas en el dataset $\mathbb{D}$. La única forma de tratar este problema es contar con un dataset $\mathbb{D}$ que abarca el espacio de transiciones con alto retorno, permitiendo un adecuado aprendizaje.

2.5.2. *Distributional shift*

Esto ocurre cuando la función de aproximación estimación está entrenada bajo una distribución, pero es evaluada en una distribución diferente, es decir, los estados visitados por la política de comportamiento aprendida $\pi(a|s)$ y su distribución $d^{\pi}(s)$ cambian frente a los encontrados en la data (política de comportamiento π_β y la distribución de los estados en la política de comportamiento $d^{\pi_\beta}(s)$). Esto sucede ya que en contexto fuera de línea la política puede entrar en estados fuera de la distribución de comportamiento $d^{\pi_\beta}(s)$ y en estos estados no se puede hacer garantías sobre el error de generalización, una vez que la política entra en un estado fuera de distribución seguirá realizando errores y puede quedar siempre fuera de la distribución[10].

Estos problemas tienen solución, pero es necesario identificar su existencia para lidiar con ellos de forma oportuna.

2.5.3. *Random Ensamble Mixture*

Random Ensamble Mixture o REM consiste en realizar una mezcla aleatoria de varios modelos para obtener un resultado más completo, lo que en la bibliografía ha mostrado mejores resultados. El objetivo detrás de esta metodología es que distintos modelos se enfocan en distintas partes del estado para encontrar la recompensa esperada, esto por variaciones en

el conocimiento del espacio acción-estado dado por las exploraciones y la ruta de minimización utilizada. De esta forma se tienen "expertos" que conocen el proceso desde un paradigma y al juntarlos se obtiene mayor información del proceso en general [43]. Incrementando el número de modelos para ensamble típicamente incrementa el rendimiento de modelos de aprendizaje supervisados y se ha demostrado que trae mejoría en aprendizaje reforzado fuera de línea. En REM-DQN se usan múltiples funciones Q parametrizadas para estimar los valores Q. Una combinación convexa de múltiples estimaciones de la función de labores Q es en sí un estimado de función de valor Q. Considerando lo anteriormente mencionado, se puede entrenar una familia de aproximadores a la función Q definidos por la mezcla de las probabilidades en un (k-1)-símplex.

En específico, para cada *mini-batch*, se escoge una distribución categórica α aleatoria, la cual describe una combinación convexa de los K estimados para aproximar la función Q óptima. Este aproximador es entrenado contra su función Q objetivo para minimizar el error de diferencia temporal 2.18. De esta forma, la función de perdida DQN queda como se muestra en la ecuación 2.17, donde P_Δ representa la probabilidad de la distribución sobre un (K-1)-símplex $\Delta^{k-1} = \{\alpha \in \mathbb{R}^k : \alpha_1 + \alpha_2 + ... + \alpha_k = 1, 0 \leq \alpha_j, j = 1, 2, ..., k\}$.

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}} \left[\mathbb{E}_{\alpha \sim P_\Delta} \left[l_\lambda(\Delta_\theta^\alpha(s, a, r, s')) \right] \right] \tag{2.17}$$

$$\Delta_\theta^\alpha(s, a, r, s') = \sum_k \alpha_k Q_\theta^k(s, a) - r - \gamma \max_{a'} \sum_k \alpha_k Q_{\theta'}^k(s', a') \tag{2.18}$$

2.6. Estado del arte

Si bien el concepto de aprendizaje reforzado fuera de línea ha estado presente desde hace décadas [44], gracias a las redes neuronales profundas y la capacidad de cómputo actual [10] se comenzaron a desarrollar nuevos métodos, los cuales aún están en su infancia. En términos generales, estos métodos siguen uno de dos conceptos distintos para atacar el problema de *distributional shift*, restringir la distribución de la función Q o generar un dataset lo suficientemente numeroso y diverso para permitir la convergencia del método. En esta sección se explicarán algunos de estos algoritmos, ventajas y desventajas.

2.6.1. Métodos de restricción

2.6.1.1. Batch Constrained deep Q-learning (BCQ)

La intuición clave detrás del algoritmo **BQC** es que se reconoce la existencia de un "error de extrapolación" (*distributional shift*), frente al cual los algoritmos fuera de línea anteriores fallaban en otorgar políticas confiables. Para atacarlo se propone utilizar *Q-learning*, pero en el paso de maximización se consideran solo las acciones a' tal que (s', a') está, probablemente, en el *replay buffer*, en vez de todas las acciones posibles.

BQC entrena un modelo generativo, en el caso clásico un *variational auto-encoder*, para reproducir acciones presentes en el dataset. Este modelo generativo está condicionado por el espacio de estado presente en los datos, produciendo solo acciones efectuadas en el dataset. Estas acciones son perturbadas por un modelo de perturbación. Con esta acción perturbada se entrena una *deep Q-network* [45].

2.6.1.2. Bootstrapping Error Accumulation Reduction (BEAR)

Al igual que **BCQ** se reconoce la existencia de un error causado por la propagación de valores-Q erroneos durante Q-learning "error de *bootstrapping*". Ambos se refieren al fenómeno de *distributional shift* siendo *bootstrapping* error terminología específica para problemas de aprendizaje reforzado que usa *bootstrapping*, mientras que el error de extrapolación es un término más amplio, usado en aprendizaje reforzado también.

La intuición detrás del algoritmo es obligar a la política de entrenamiento π a tener el mismo soporte que la política de comportamiento π_β. Esto se logra a través de *Maximum Mean Discrepancy* (MMD), la distancia entre las acciones de un de una política de comportamiento desconocida π_β y el actor π [46]. Este método de restricción utiliza un ensamble de redes Q, pero se podría aplicar a otras arquitecturas. Usando lo anterior, la ecuación para mejorar la política viene dada por la Ecuación 2.19[46].

$$\pi_\phi := \max_{\pi \in \Delta_{|S|}} \mathbb{E}_{S\sim\mathcal{D}} \mathbb{E}_{a\sim\pi(.|s)} \left[\min_{j=1,\ldots,k} \hat{Q}_j(s,a) \right] \quad \text{s.t} \quad \mathbb{E}_{S\sim\mathcal{D}} \left[\mathbf{MMD}(\mathbb{D}(\cdot|s), \pi(\cdot|s)) \right] \leq \epsilon \tag{2.19}$$

La restricción ocupada muestra propiedades de convergencia con suposiciones más débiles,

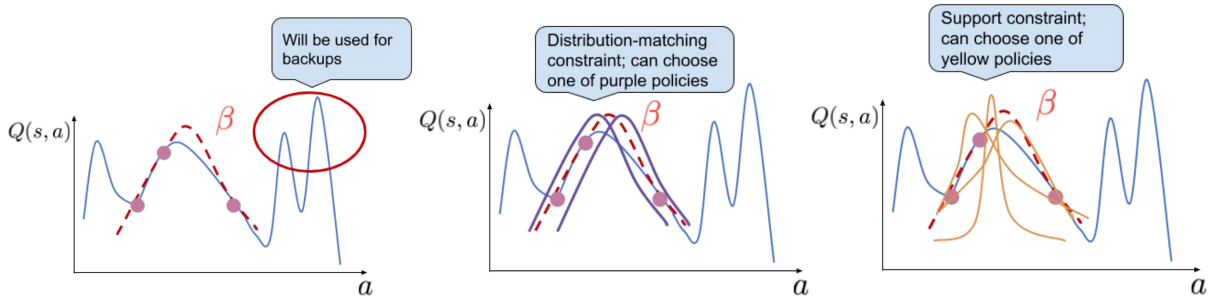


Figura 2.5: Ilustración de la restricción de soporte (BEAR) a la derecha, restricción por igualación de distribución al centro y ejemplo de valores Q fuera de la distribución[46].

siendo capaz de restringir la suboptimalidad del método. Frente a BCQ, BEAR se comporta mejor cuando la los datos son recolectados con una política subóptima (o incluso aleatoria). Mientras que BCQ es generalmente mejor cuando los datos son de una política experta[46].

2.6.2. Métodos enfocados en datos

2.6.2.1. Scaling Data-driven Robotics with Reward Sketching and Batch Reinforcement Learning

Más que un método en sí busca mostrar una metodología de aprendizaje, donde se construye un dataset extenso y variado mediante la interacción de un humano y el entrenamiento mediante métodos fuera de política[47]. Combinando ideas del aprendizaje reforzado, interacción computacional y bases de datos. Los pasos de esta metodología son:

- Obtener demostraciones: pueden obtenerse por diversos métodos como teleoperación humana, políticas escritas o entrenadas. A medida que se entrena se introducen las políticas efectuadas por el agente.

- Elegir recompensa: a un subset de datos escogidos se les indica la recompensa producida, seleccionada por humanos.
- Aprendizaje de la recompensa: se aprende una red neuronal r_ψ para redecir la recompensa asociada a cada intervalo de tiempo.
- Aprendizaje reforzado fuera de línea: Se entrena al agente, no es necesario utilizar un algoritmo ideado para trabajar fuera de línea, simplemente uno fuera de póliza.
- Evaluación: periódicamente evaluar al robot y incorporar nuevas demostraciones al dataset.
- Repetir el bucle.

2.6.2.2. REM-DQN

Este algoritmo se explicó con mayor detalle en secciones superiores, dado que es el algoritmo elegido para resolver el problema. Es interesante abordarlo dado que presenta buenos resultados sin la necesidad de acotar el espacio de estado a las parejas acción estado vistas en el dataset, pero necesitando una mayor cantidad y diversidad de datos para el entrenamiento [48].

2.6.3. Futuro y benchmarks

Es posible identificar *benchmarks* para algoritmos de aprendizaje reforzado en línea, tal como los juegos de Atari [49]. Sin embargo, para los algoritmos fuera de línea se continúan generando compendios de data sets que permitan conocer las ventajas de cada método [50].

El aprendizaje reforzado online está evolucionando rápidamente, dentro de los métodos que mayores promesas entregan se encuentran REM-DQN, SAC, TD3, AWAC, BCQ, BEAR, CQL, CRR, PLAS, PLAS+P, IRIS y TD3+BC. Cada uno de estos métodos sobresale en una tarea en particular [51]. Dependiendo de la forma en como se construye el dataset hace variar que método alcanza mejores métricas [48]. En otras tareas de aprendizaje de máquinas se construyen nichos en torno a sets de datos, utilizando este set de datos como validación del desempeño del algoritmo o arquitectura en esa actividad en particular, como resulta con las bases de datos CIFAR-10,CIFAR-100[52] y ImageNet [53]. Al estudiar una base de datos estática se puede conocer las ventajas y falencias de esta base de datos, que a su vez demuestran las fortalezas de un método enfrentándose a ese problema.

Para aprendizaje reforzado se ha trabajado construyendo simuladores de ambiente, pensando en su modalidad en línea o fuera de póliza. Últimamente estos ambientes se han usado para generar bases de datos, pero el como se construyen estas bases de datos varía el desempeño final del aprendizaje fuera de línea [48][43]. La importancia de un dataset estático presenta la posibilidad de estudiar las características para mejorar los algoritmos hacia distintos nichos. Puede cambiar el desempeño de un algoritmo con el solo hecho de cambiar la cantidad de datos [43], calidad de los datos[45], ambiente [51], entre otros.

A medida que se avance el estado del arte comenzarán a aparecer sets de datos aceptados de forma transversal.

2.7. Uso del marco teórico

Para finalizar la sección se presenta un diagrama que enlaza los términos explicados y su uno en el presente estudio. Tal como muestra la figura 2.6 utilizando el modelo fenomenológico

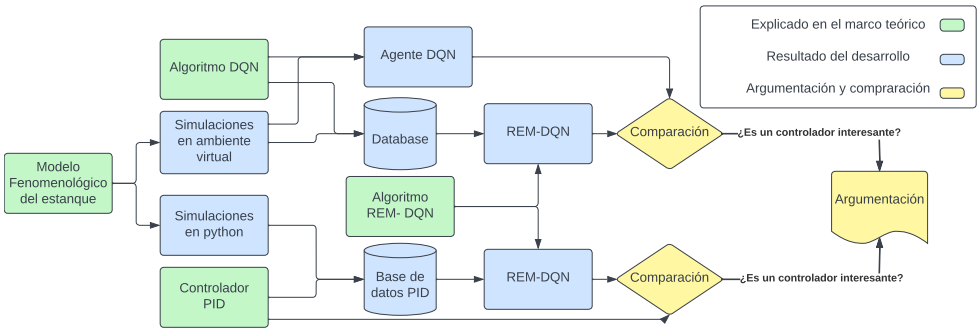


Figura 2.6: Activación por rango

del estanque cónico se entrenan controladores PID, un agente DQN y agentes REM-DQN con el fin de poner a prueba la capacidad de un agente REM-DQN de controlar el sistema. Para esto es importante explicar los conceptos básicos detrás del modelo fenomenológico a utilizar, el controlador PID y algoritmos de aprendizaje reforzado.

3. Metodología

La implementación realizada consiste en el algoritmo REM-DQN, el código corresponde a una iteración sobre una versión actualizada del código del artículo original [43]. En este se estudia la posibilidad de usar técnicas de ensamble en aprendizaje reforzado offline sin necesidad de ajustar debido al movimiento de la distribución, el estudio se hace sobre la capacidad de aprender y el desempeño de técnicas de ensamble en juegos de la consola *Atari 2600*. En esta sección se explicará el código y las modificaciones necesarias de este, el cual se divide en tres partes fundamentales, la generación de datos, el entrenamiento offline y la inferencia.

3.1. Diseño del experimento

3.1.1. Creación del Ambiente virtual

Para poder generar una base de datos a partir de donde el algoritmo de REM-DQN aprenda, es necesario crear un algoritmo que simule numéricamente un ambiente, es decir que pueda generar datos de la forma (s, a, s', r), para esta primera etapa es importante que se pueda interactuar con este a posteriori con el fin de probar que tan bien resulto el algoritmo.

Se decidió realizar un ambiente virtual personalizado de *GYM*, lo que permite que pueda ser usado en distintos algoritmos de aprendizaje reforzado, sin necesidad de variar el código interno. Esto se debe a que no hace presunciones sobre la estructura del agente y es compatible con cualquier librería de computación numérica[17].

Un ambiente virtual no es más que una simulación con ciertos componentes que le permiten interactuar de buena manera con librerías pre-existentes de aprendizaje reforzado. Históricamente estas simulaciones se han llamado ambientes virtuales y simulan las interacciones con el agente.

En términos generales, un ambiente de *GYM* necesita dos funciones, una que restaure los valores a los iniciales una vez que se completa una iteración, y una segunda que realice un paso, es decir, se le entrega una acción y revuelva $(s', r, done)$, donde s' corresponde al siguiente estado, r la recompensa asociada a esa transición de estado y *done* un booleano que es 0 si se sigue la iteración y 1 si se terminó la iteración. En el pseudocódigo 1 se observa el funcionamiento de la función de paso.

Código 1 Función de paso

1: set_point← new_set_point ▷ Cambiar setpoint con una probabilidad de 0.5 %
2: u←frecuencia(a) ▷ Traducir la acción a un cambio en la frecuencia
3: $h(s,u) \leftarrow$solve$(\dot{h}(s,u))$ ▷ Se resuelve la ecuación diferencial 2.6
4: r←reward$(e, d_e, s, ...)$ ▷ Se calcula la recopmensa
5: return $(s', r, done)$ ▷ Inicializar ambiente

Es común que los ambientes tengan una función que muestre el ambiente, esta función se activa sí $RENDER == True$. Para este ambiente se tiene una visualización como la que se muestra en la Figura 3.1. En la Figura 3.1 se puede ver el seguimiento de la altura al *setpoint*,

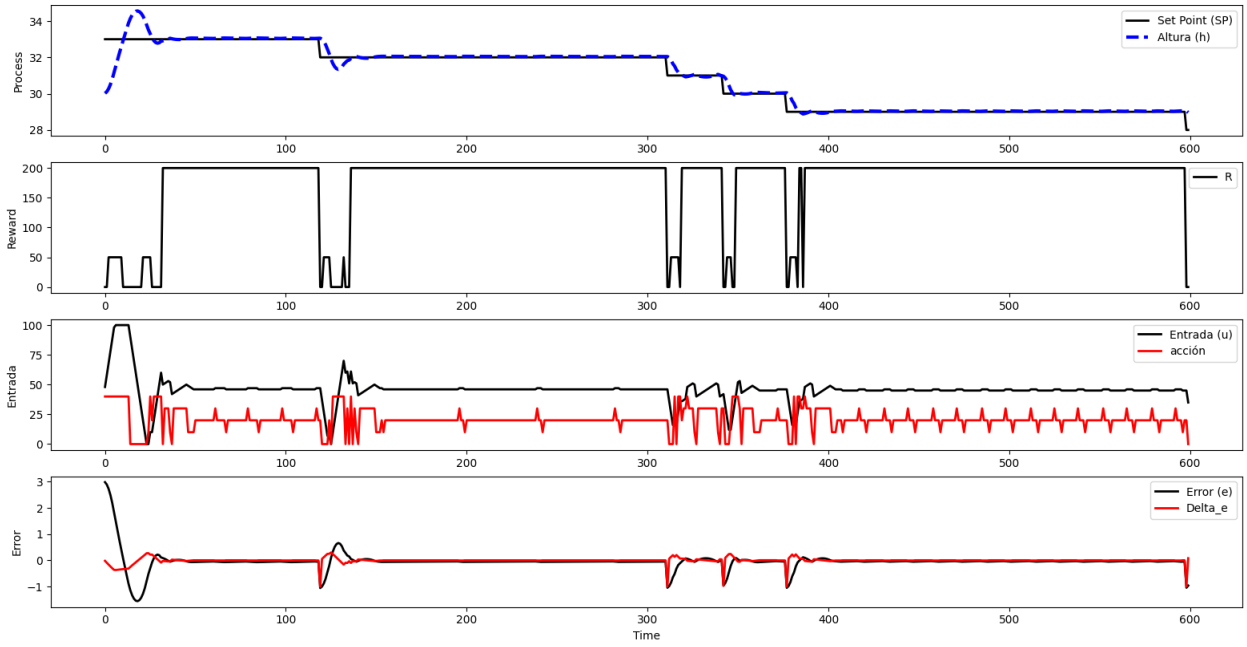


Figura 3.1: Ejemplo del ambiente virtual visualizado.

la recomienza asociada a ese estado, la entrada y su acción asociada y por último el error y la variación de este.

3.1.2. Generación de datos

En [43] se crea un set de datos mediante el entrenamiento de varias instancias de agentes DQN online, en particular para cada juego que se estudia se entrenan cinco agentes distintos, con inicializaciones aleatorias y se guardan en tuplas de (observación, acción, recompensa, siguiente observación). Por la naturaleza del problema, este set de datos resulta 3.5[43] veces más grandes que ImageNet [54] e incluye muestras de las políticas intermedias vistas mientras se optimiza el agente DQN.

Para esta investigación se crean dos bases de datos, una mediante el entrenamiento de un agente DQN y otra mediante la simulación del estanque controlado por PIDs con parámetros variables cercanos a una linealización en torno a tres alturas con funciones de activación

tangente hiperbólicas para intercambiar. Esta última nace con la idea de simular el comportamiento sub-óptimo que se podría presentar en un periodo de tiempo frente a un cambio de turno en una planta.

3.1.3. Estanque DQN dataset

Para generar el data set con la red DQN se usa tres capas lineales, dos de *batch normalization* y dos de activación *ELU*. Estas se disponen intercaladas (lineal,*batch norm*, *ELU*) terminando con una lineal. Las entradas de las capas lineales son espacio de observación, 256, 64 y la salida de la última coincide con el espacio de acción. Esta arquitectura es la que da mejores resultados dado el problema presentado luego de probar variaciones de esta. Si bien es posible obtener mejores resultados con un análisis exhaustivo de los posibles agentes controladores DQN basta generar un controlador lo suficientemente bueno para poder generar una base de datos que permita un entrenamiento. Ambas bases de datos (DQN y PID) son imperfectas por diseño.

Código 2 Generación

```
env.make(), agent.make()                          ▷ Inicializar ambiente y agente
for Iteraciones de calentamiento do
    a ∼ A                                         ▷ Se toma una acción aleatoria
    s', r ← env.step(a)          ▷ Se obtiene el estado y recompensa asociada a la acción
    D ← D ∪ (s, a, s', r)                         ▷ Se agrega la experiencia al Replay Buffer

for Numero de iteraciones do
    a ← agent.act(s)          ▷ El agente elige una acción para el estado del sistema
    s', r ← env.step(a)          ▷ Se obtiene el estado y recompensa asociada a la acción
    D ← D ∪ (s, a, s', r)                         ▷ Se agrega la experiencia al Replay Buffer
    agent.learn()                                     ▷ Se realiza un paso de aprendizaje
    D̄ ← D̄ ∪ (s, a, s', r)                         ▷ Se agrega la experiencia al Dataset
    if done then
        env.reset()
env.close(), agent.close()                        ▷ Se cierra el agente y el ambiente
```

El pseudocódigo 2 muestra el funcionamiento del código generador de datos, en términos generales se itera un agente DQN que a medida va aprendiendo y cambia su política se guardan estas interacciones en un data set $\overline{\mathcal{D}}$ para ser utilizadas en el entrenamiento fuera de línea.

3.1.4. Estanque PIDs dataset

El data set generado mediante el control realizado por PIDs aleatorios en torno a linealizaciones realizadas a distintas alturas intenta imitar el comportamiento subóptimo, pero cercano al óptimo, entregado por un operador.

El código para generar el data set sigue la lógica presentada en el pseudocódigo 3. Para cada iteración se inicializan valores aleatorios y se itera en el tiempo, controlando la altura con el PID y guardando la experiencia en el data set. donde f es una función de activación por rangos, definida como se presenta en la ecuación (3.0), se ve sobre complicada, pero

Código 3 Generación

for Iteraciones **do**
  $K_1, \tau_I 1, \tau_D 1, ..., K_4, \tau_I 4, \tau_D 4$ ▷ Se definen las constantes para la iteración
  for Pasos de tiempo **do**
    set_point ← new_set_point ▷ Cambiar setpoint con una probabilidad de 0.5 %
    $K \leftarrow f(K_1, K_2, K_3, K_4, h)$ ▷ Se escogen las constantes PID cercanos a la...
    $\tau_I \leftarrow f(\tau_I 1, \tau_I 2, \tau_I 3, \tau_I 4)$ ▷ linealización para esa altura
    $\tau_D \leftarrow f(\tau_D 1, \tau_D 2, \tau_D 3, \tau_D 4)$
    $P, I, D \leftarrow calcPID(K, \tau_I, \tau_D, e, ie, \dot{e})$ ▷ Se calculan los parámetros PID
    $a \leftarrow action(P, I, D, u_0, u_{t-1})$ ▷ Se calcula la acción a tomar
    u←frecuencia(a) ▷ Traducir la acción a un cambio en la frecuencia
    u←antiwindup(u) ▷ Se realiza anti-windup
    $h(s, u) \leftarrow$ solve$(h(\dot{s}, u))$ ▷ Se resuelve la ecuación diferencial 2.6
    r← reward$(e, d_e, s, ...)$ ▷ Se calcula la recompensa
    $\overline{\mathcal{D}} \leftarrow \overline{\mathcal{D}} \cup (s, a, s', r)$ ▷ Se agrega la experiencia al *Dataset*

en términos generales pondera los términos x_1, x_2, x_3, x_4 por la activación de esa altura, la Figura 3.2

$$\begin{aligned} f(x_1, x_2, x_3, x_4, h) =& x_1 \frac{-tanh(h-5)+1}{2} + x_2 \frac{tanh(h-5) - tanh(h-10)}{2} \\ &+ x_3 \frac{tanh(h-10) - tanh(h-21)}{2} + x_4 \frac{tanh(h-21)+1}{2} \end{aligned} \tag{3.0}$$

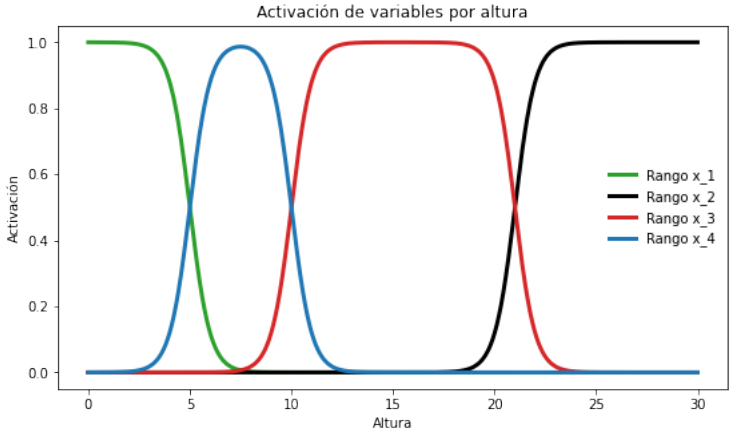


Figura 3.2: Activación por rango

3.1.5. Entrenamiento offline

Con los data sets recopilados se entrena al agente REM-DQN de forma similar al agente DQN online. En el pseudocódigo 4 se muestra el bucle de entrenamiento, es importante notar que el ambiente se utiliza únicamente para modelar el retorno asociado al desempeño del agente en la época actual.

Código 4 Entrenamiento REM-DQN

```
env.make(), agent.make(), D̄   ▷ Inicializar ambiente, agente y un dataloader del dataset.
for Numero de épocas do
    for Numero de batch do
        batch ∼ D̄                                   ▷ Se extrae un batch del dataset
        agent.learn(batch)
    R=0                                             ▷ Se inicializa la recompensa total
    for Episodios de evaluación do
        a ← agent.act(s)          ▷ El agente elige una acción para el estado del sistema
        s', r ← env.step(a)       ▷ Se obtiene el estado y recompensa asociada a la acción
        R ← R + r
        if done then
            env.reset()
    return R
env.close(), agent.close()                          ▷ Se cierra el agente y el ambiente
```

3.1.6. Inferencia

El entrenamiento del agente REM-DQN resulta en un conjunto de redes neuronales estimadoras de la función Q, ya que se guarda un snapshoot después de terminar cada época del entrenamiento. La etapa de inferencia sirve para evaluar las políticas alcanzadas, equivalente a permitir que el agente controle en la planta real. El bucle de inferencia es equivalente al bucle de evaluación del pseudocódigo 4, solo que esta vez se renderiza.

3.2. Metodología de análisis

Al momento de escritura de esta tesis los esfuerzos realizados con esta metodología se ven volcados a obtener mejores métricas de recomienza sin tomar en consideración la calidad del control entregado y robustez de este frente a distintos desperfectos que pudiesen ocurrir. En esta tesis se pretende observar las capacidades de un agente REM-DQN de controlar un sistema.

Para esto realiza un control del sistema mencionado anteriormente y se observa la mejoría del agente REM-DQN frente al controlador a partir del cual fue entrenado. Para esto se hace una comparación de desempeño y comportamiento frente a distintos modos de falla.

3.2.1. Desempeño

Para medir el desempeño de los controladores se observan distintas métricas:

- Respuesta del controlador frente a una entrada escalonada: Es común que este tipo de controladores tenga problemas para poder controlar a ciertas alturas. Esto ocurre cuando no se pasó por estos estados, dando una mala aproximación de la función de merito.
- Recompensa promedio: La función de recompensa fue diseñada para mantener el controlador lo más cerca del *set-point*. Mientras más alto mejor es el desempeño.
- Esfuerzo de control: Cambio en la entrada para cada controlador. Mientras menor mejor.

Comúnmente se debe limitar el esfuerzo de control por las limitaciones en equipos como actuadores, sensores y redes [55] .

3.2.2. Comportamiento frente a eventualidades

Los sistemas están expuestos a degradación en el tiempo o fallas de componentes, por lo que es importante que el sistema de control sea robusto ante este tipo de perturbaciones. La ecuación (2.6) define la dinámica del estanque y permite tres tipos de perturbaciones.

- Cambio de la frecuencia de entrada: corresponde a cualquier tipo de problema que interfiera con la capacidad de la bomba para trabajar en condiciones nominales. Puede suceder debido a obstrucciones en el tubo de entrada, abolladuras o fallo de la bomba en sí. Se modela mediante cambios en las constantes c_1 y c_2.

- Cambio de la frecuencia de salida: corresponde a cualquier tipo de problema que interfiera con la frecuencia de salida. Su puede deber a perforaciones en el estanque o la tubería inferior, como también a obstrucciones o ensanchamiento de esta. Se modela mediante cambios en la constante β.

- Cambio con la geometría del estanque: corresponde a daños estructurales provocados al estanque, cambiando su geometría. Se modela mediante cambios en la constante α.

4. Resultados y análisis

En las secciones anteriores se mencionó el caso que se pretende estudiar, la motivación para este y la implementación del aprendizaje. En esta sección se presenta la creación de los datos y el entrenamiento de los agentes REM-DQN correspondientes. Para el correcto funcionamiento de estos se tunea la tasa de aprendizaje, el tamaño del batch, la arquitectura de la red, el espacio de observación y el espacio de acción.

4.1. Creación de Datos mediante entrenamiento de un agente online

4.1.1. Funciones de mérito

Se probaron 2 funciones de recompensa para esta etapa **CRF** y **FPID**. **CRF** aumenta exponencialmente hasta saturarse en 2000, como se muestra en el código 5.

Código 5 CRF

```
1: function CFR(e):
2:     if |e|<0.1 then
3:         r=2000
4:     else
5:         r = 100/|e|
6:     return r
```

FPID sigue la lógica de un controlador PID difuso, el cual le otorga recompensa si el control de seguimiento va en la dirección correcta, entendido como el signo del error, y con la velocidad correcta, entendido con la derivada del error. En la Tabla 4.1 muestra la relación entre mérito y positividad/negatividad del error y su derivada.

e/Δ e	N	0	P
N	–	-	+
0	–	++	–
P	+	-	–

Tabla 4.1: Merito según error y derivada del error.

4.1.2. DQN FPID

Se iteró sobre la tasa de aprendizaje, encontrando que 0.00001 se encontraban los mejores resultados y se decide correr el algoritmo hasta la 2600. Variar el número de iteraciones puede resultar en mejores resultados, pero el fin de la creación de esta base de datos es mostrar que en la ineficacia explorada es posible encontrar patrones adecuados para generar el aprendizaje fuera de línea. En la Figura 4.1 se pueden observar distintas tasas de aprendizaje, seleccionándose la data recopilada de las interacciones del agente en color rojo.

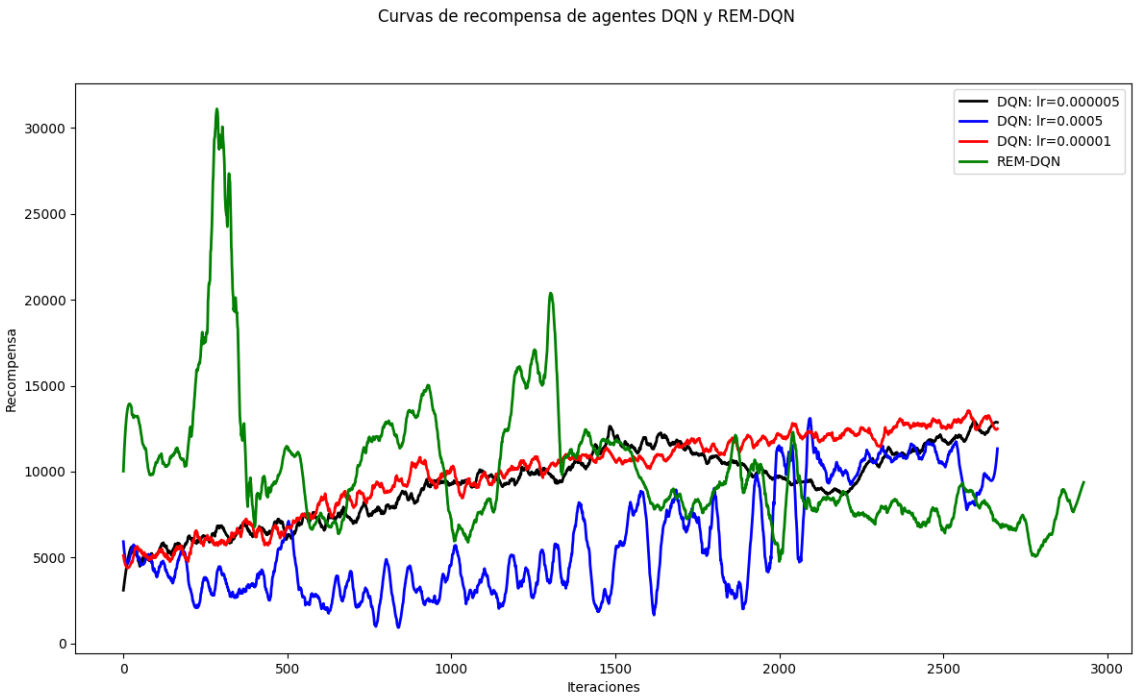


Figura 4.1: Curva de recompensa en el entrenamiento de distintos agentes

Con el fin de comparar rápidamente el comportamiento de un agente, se decide generar un gráfico donde se pase por las alturas pertinentes, dando poco más de cincuenta instantes de tiempo para la estabilización por parte del agente, el cual se presenta en la Figura 4.2. Es posible observar que el resultado del mejor agente DQN tiene problemas de *overshooting* para todas las alturas superiores a veinte centímetros, pudiendo controlar bien en veinte centímetros y luego volviéndose incapaz de controlar.

Si bien este control no es perfecto, no es necesario que lo sea, ya que tiene la finalidad de completar el data set para un agente REM-DQN offline. Resulta interesante es ver que es capaz de aprender el agente REM-DQN.

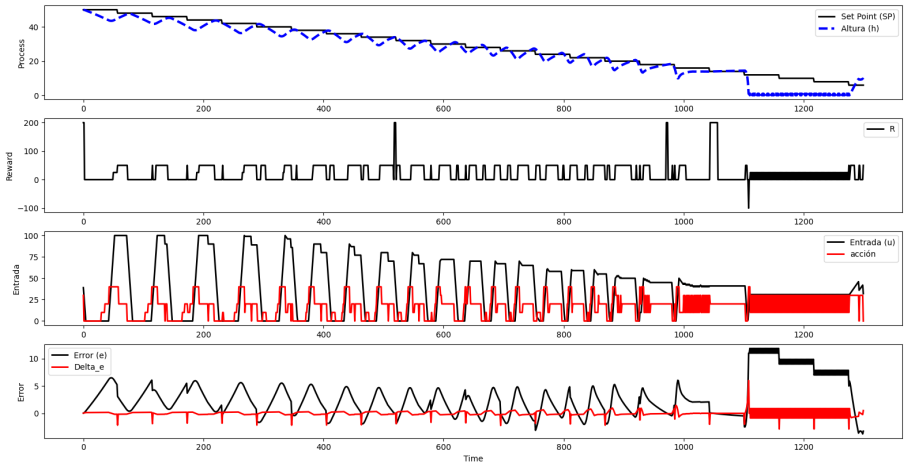


Figura 4.2: Respuesta del agente DQN entrenado con la función de mérito **FPID** a la referencia escalonada.

4.1.3. DQN CFR

Al igual que con la función de mérito **FPID**, se iteró sobre la taza de aprendizaje, encontrando que 0.0001 es la que entrega mejores resultados. En la Figura 4.3 se puede observar la recompensa durante el proceso de entrenamiento para los distintos agentes DQN suavizado. Se puede observar que el agente aprende como controlar el sistema y el aprendizaje tiene pendiente positiva a diferencia del entrenamiento para la función **FPID** donde la recompensa pareciera estancarse.

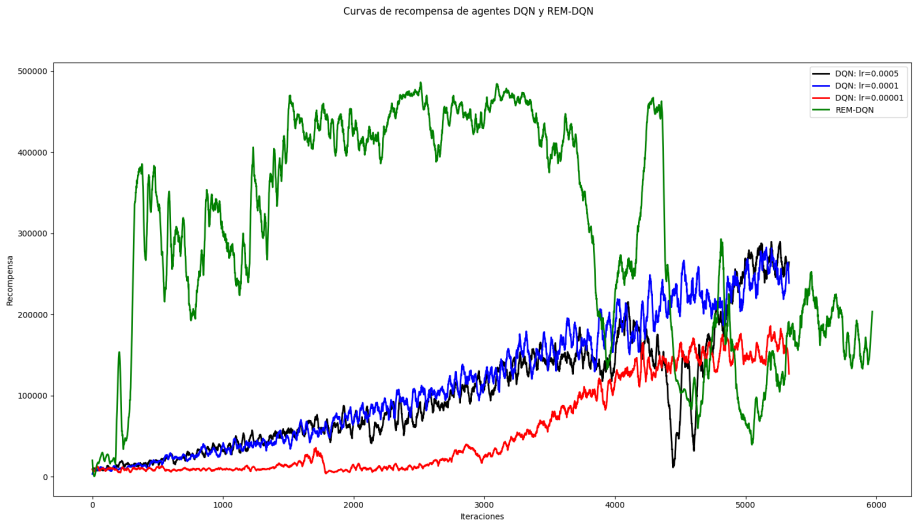


Figura 4.3: Curva de recompensa en el entrenamiento de distintos agentes.

Nuevamente, se pretende obtener una idea de cómo se comporta el controlador frente

a la referencia escalonada, presentándose este en la Figura 4.4. Es posible observar que el controlador DQN tiene un gran control del sistema, con problemas menores. Para captar problemas con el control basta observar el retorno de recompensa, si este toma tiempo en llegar al máximo se tiene un tiempo de estabilización lento, si esta llega el máximo y vuelve a bajar da evidencia de *overshoot*, si no llega al máximo muestra un *offset* y si oscila evidencia oscilaciones. Para alturas mayores a treinta centímetros tiene un tiempo de estabilización muy lento, pudiendo ocupar una variedad de frecuencias para la entrada se mantiene cerca de las que controlan el sistema. Para alturas intermedias es inconsistente, en algunas es capaz de controlar con bajo tiempo de estabilización, con otras oscila y con otras mantiene un offset o tiempo de estabilización muy largo.

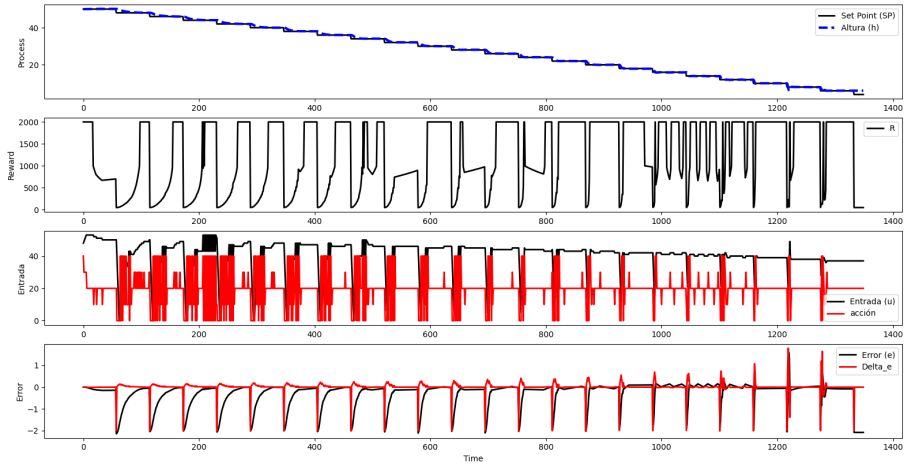


Figura 4.4: Respuesta del agente DQN entrenado con la función de mérito **CRF** a la referencia escalonada.

4.2. Creación de Datos mediante PID

Se crean distintas bases de datos a medida que se va variando el espacio de acción y observación, la longitud de esta depende exclusivamente de la cantidad de RAM presente en el computador. Generalmente se lograba generar bases de datos con 40000 iteraciones de 600 instantes de tiempo. En la Figura 4.5 se muestra un ejemplo del tipo de control realizado por los controladores PID. Si bien estos realizan un buen trabajo, para el intervalo de alturas menores a diez centímetros, por lo general, los controladores no son capaces de controlar el sistema. Esto se debe principalmente a la alta no-linealidad presente en esta zona, un ejemplo de esto se presenta en la Figura 4.6.

Si bien es cierto que el control no es el adecuado para alturas muy bajas, estos datos con sus imperfecciones sirven para entrenar a un agente que intente encontrar patrones adecuados para un controlar el sistema dentro de los márgenes que le posibilita la data recopilada.

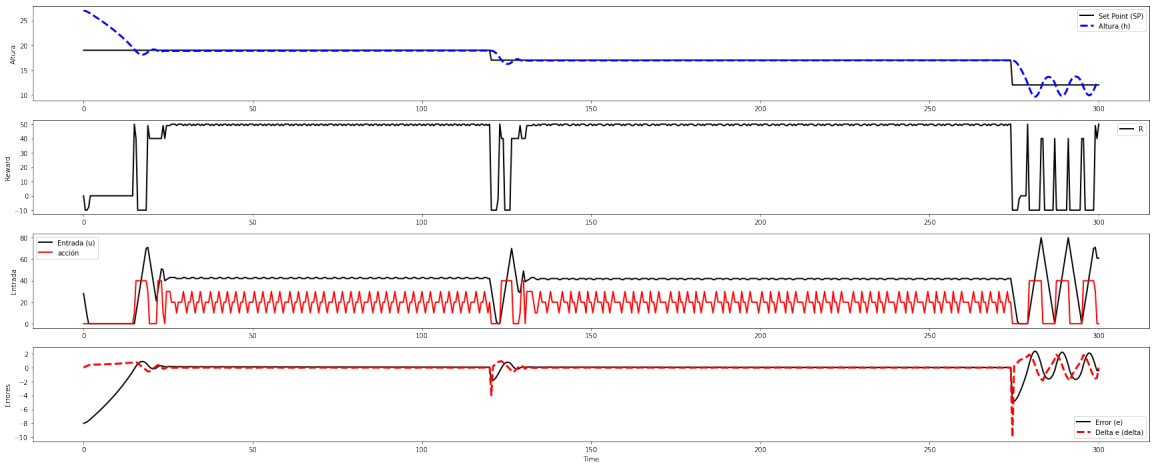


Figura 4.5: Ejemplo de datos generados mediante control PID

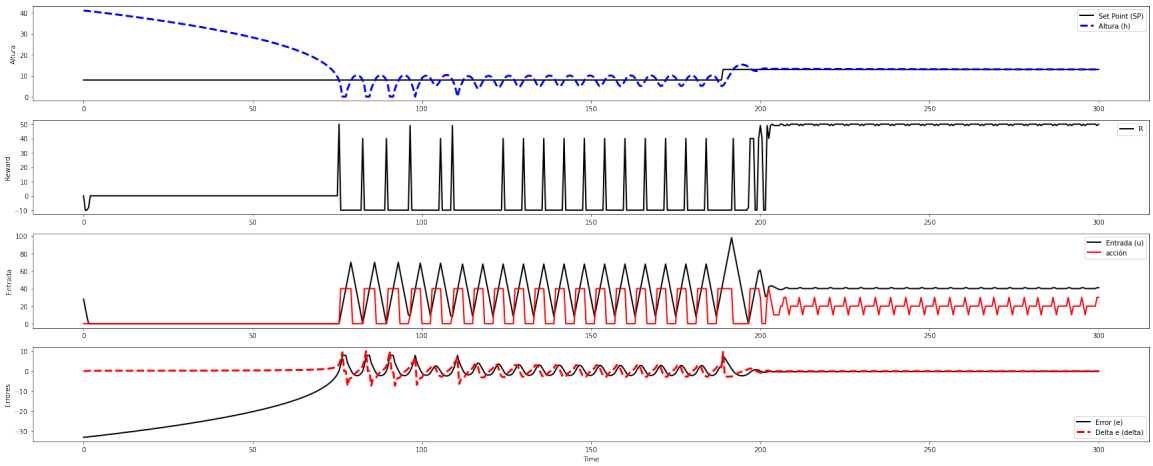


Figura 4.6: Fallo del control PID debido a la altura.

4.3. Entrenamiento REM-DQN offline

En esta sección se presenta la forma en la cual se entrenaron los tres casos distintos y los resultados pertinentes.

4.3.1. REM-DQN usando datos DQN-CFR

En la Figura 4.3 se observa que la recompensa promedio del agente REM-DQN alcanza valores superiores a los alcanzados por el agente DQN creador de los datos que hicieron posible el entrenamiento, alcanzando valores cercanos al doble. Para realizar la comparación se utiliza la red generada en la época 2859, obteniéndose la respuesta a la referencia escalonada, como se presenta en la Figura 4.7

Es posible observar que se tiene un muy buen control para alturas doce centímetros, donde comienza a tener comportamientos oscilantes. Es posible ver que para alturas bajas el controlador oscila en torno a la referencia, obteniendo el valor máximo para luego volver a uno intermedio. Lo anterior se ve plasmado en las acciones que toma el controlador en este

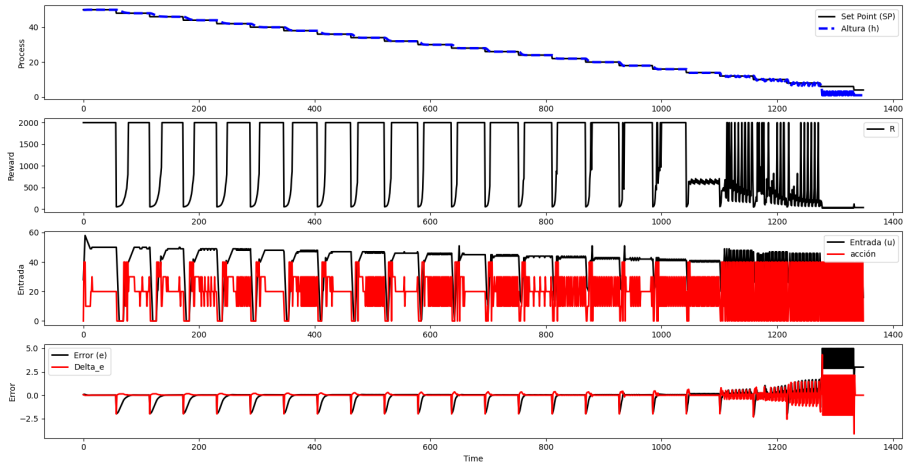


Figura 4.7: Respuesta del agente DQN a la referencia escalonada.

segmento de alturas, intercalando entre subir 10 % la frecuencia de la bomba y disminuir 10 % la frecuencia de la bomba. Para la última altura presentada se observa un *offset* en el control.

4.3.2. REM-DQN usando datos DQN-FPID

En la Figura 4.1 se puede observar la curva de recompensa promedio para cada iteración, esta es suavizada mediante un filtro Savitzky–Golay con una ventana de 51 ajustado a un polinomio de orden 3. Se observa que el desempeño del agente REM supera al de los agentes DQN en torno a la época trescientos, teniendo altos y bajos. Se escoge el agente de la época 293 para ser comparado cualitativamente con el desempeño del mejor agente DQN.

En la Figura 4.8 se observa la respuesta a la referencia escalonada, tal como se hizo con el agente DQN. Es posible observar que el desempeño del agente REM-DQN es superior al DQN. La estrategia de control que decide tomar el agente REM-DQN es interesante, fluctúa entre las acciones 4 y 0, las cuales corresponden a subir y bajar 10 % de la frecuencia de la bomba, lo cual produce un control con alto costo. Se observa que para valores superiores a treinta centímetros existe un *overshoot* menor al producido por el agente DQN, bajo treinta centímetros oscila en torno a la referencia con una alta frecuencia, lo que genera menos ganancia con esta función de recompensa. Bajo los diez centímetros sigue el mismo comportamiento, pero con un *offset*. Estas conclusiones se llegan a partir del gráfico del error y la recompensa.

Estos resultados pueden mejorar si se obtienen mejores iteraciones del agente DQN. Lo anterior no es difícil, ya que se utiliza la versión Nature DQN que obtiene la mitad de los puntos que la versión MuZero correspondiente al estado del arte. No obstante, los resultados obtenidos demuestran la capacidad del agente REM-DQN mejorar las políticas presentes en los datos de entrenamiento, creando una política superior.

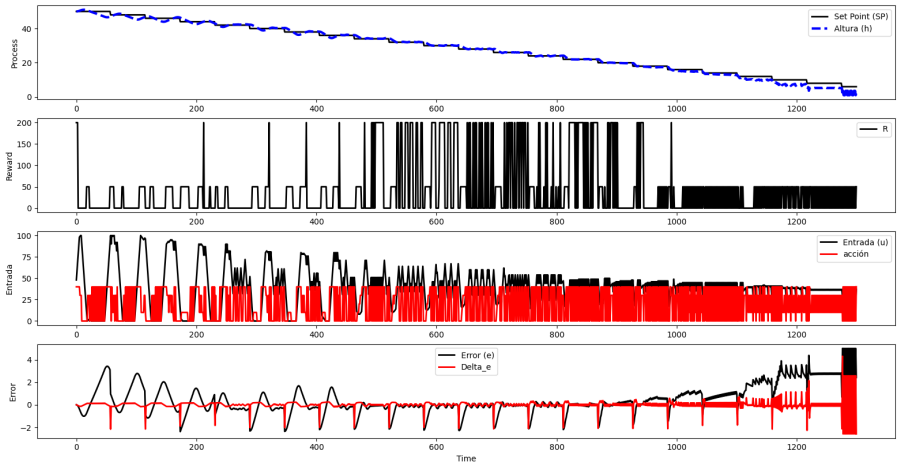


Figura 4.8: Respuesta del agente DQN a la referencia escalonada.

4.3.3. REM-DQN usando datos PID

El agente REM-DQN entrenado usando datos PID funciona de buena manera, sobrepasando la recompensa obtenida por el control mediante PID. Lo anterior se estudia con más detalle en la sección de **comparación de control** donde se utiliza este agente para realizar la comparación.

4.4. Comparación Control

Resulta interesante comparar el desempeño comparativo del agente REM-DQN frente al controlador PID en el ambiente virtual, para lo cual se decide observar cómo reaccionan a posibles fallas en la planta y el desempeño de ellos en general. Se decide utilizar el agente REM-DQN entrenado mediante PID aleatorio, esto motivado por la idea de recrear el comportamiento de un operario de la planta y el entrenamiento con data de agentes DQN de cierta forma es sintética.

4.4.1. Desempeño

La forma directa de comparación es mediante el desempeño en el seguimiento de referencia, a simple vista el agente REM-DQN es capaz de lograr un mayor desempeño comparado a un PID aleatorio. Para comprobarlo se realizan mil iteraciones donde la referencia es la misma para ambos controladores y anota la recompensa y el esfuerzo de control, entendido como variación en la frecuencia de la bomba. Los resultados se presentan en la Tabla 4.2, en la cual es posible observar que se tiene un aumento del desempeño en un 38.6 %, pero este aumento viene con un mayor esfuerzo de control, obteniendo un promedio cercano al triple del esfuerzo realizado por el controlador PID. Lo anterior se explica porque no se le restringe en la función de recompensa el esfuerzo de control. Es ciertamente discutible la utilidad de los resultados dependiente de la aplicación, si fuera importante el esfuerzo de control debiese hacerse explícito en la función de mérito, de lo contrario el algoritmo aprende a comportarse maximizando la recompensa obtenida.

Criterio	PID	REM-DQN	Variación
Recompensa promedio	33,299.15	46,155.53	38.6 %
Esfuerzo de control	281,763.37	807,762.20	186.6 %

Tabla 4.2: Comparación del desempeño entre control PID y agente REM-DQN

La Figura 4.9 presenta la comparación del control realizado por el agente frente al control realizado por el controlador PID. Como el PID se inicializa de forma aleatoria no se puede decir nada de este, pero pareciera ser que la parte integral es baja para valores altos, lo que se aprecia en lo lento que se minimiza el *steady state error*. En términos generales, el agente no tuvo problemas para valores superiores a 20 centímetros. En la Figura 4.10 muestra el

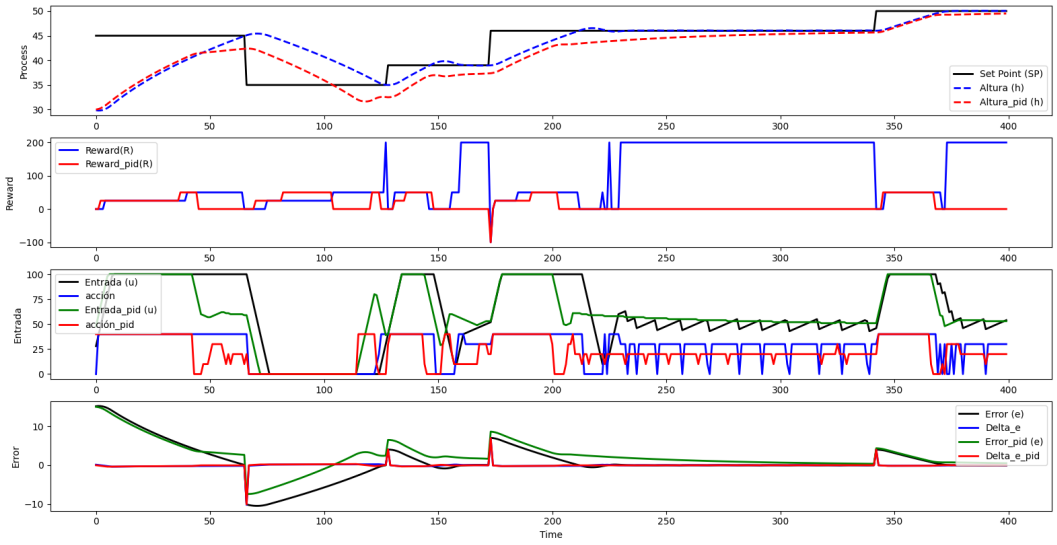


Figura 4.9: Comparación del desempeño de ambos controladores para valores altos

comportamiento de los controladores para valores bajos. Se puede observar que el controlador PID para este rango produce *overshoot* alto, esto se puede atribuir al desbalance del término derivativo.

Se puede observar que el desempeño del agente fue superior en el intervalo cercano a veinte centímetros, inferior en el intervalo cercano a quince centímetros y discutiblemente igual en valores menores a diez centímetros. Revisando los datos era común que los controladores PID no fueran capaces de controlar de manera adecuada en este rango, lo cual explica en parte porque el agente no es capaz de controlar en este segmento.

Es discutible si los resultados presentados son de interés, es necesario ponerse en el marco

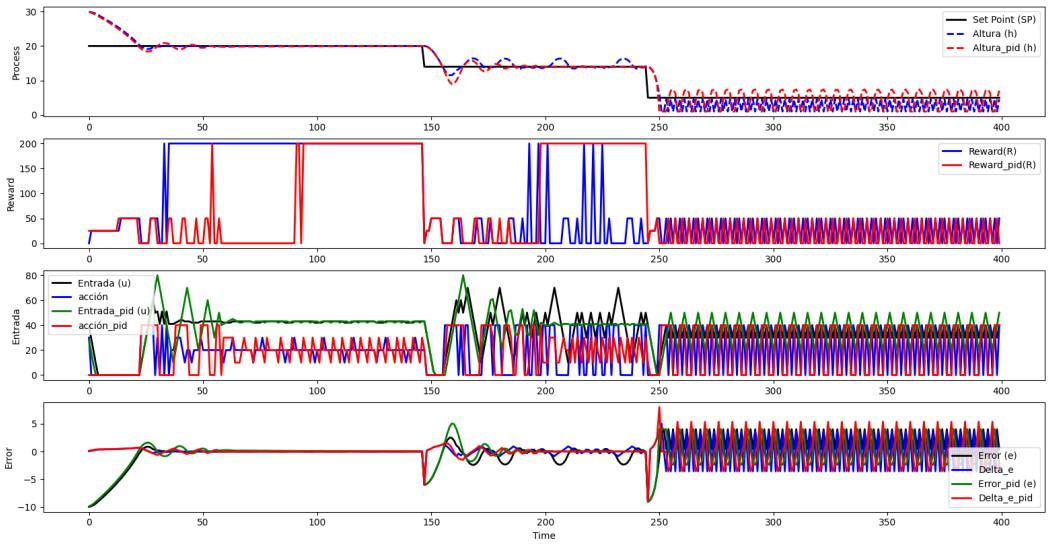


Figura 4.10: Comparación del desempeño de ambos controladores para valores bajos

fuera de línea para comprender la utilidad del método. Si contamos con un sistema no lineal, costoso y difícil de modelar, basta tener suficientes datos sobre el funcionamiento diario de este para obtener información importante, en forma de recomendación, para mejorar el desempeño diario dado lo que se ha vivido anteriormente. Este es el contexto que motiva el estudio.

4.4.2. Comportamiento frente a eventualidades

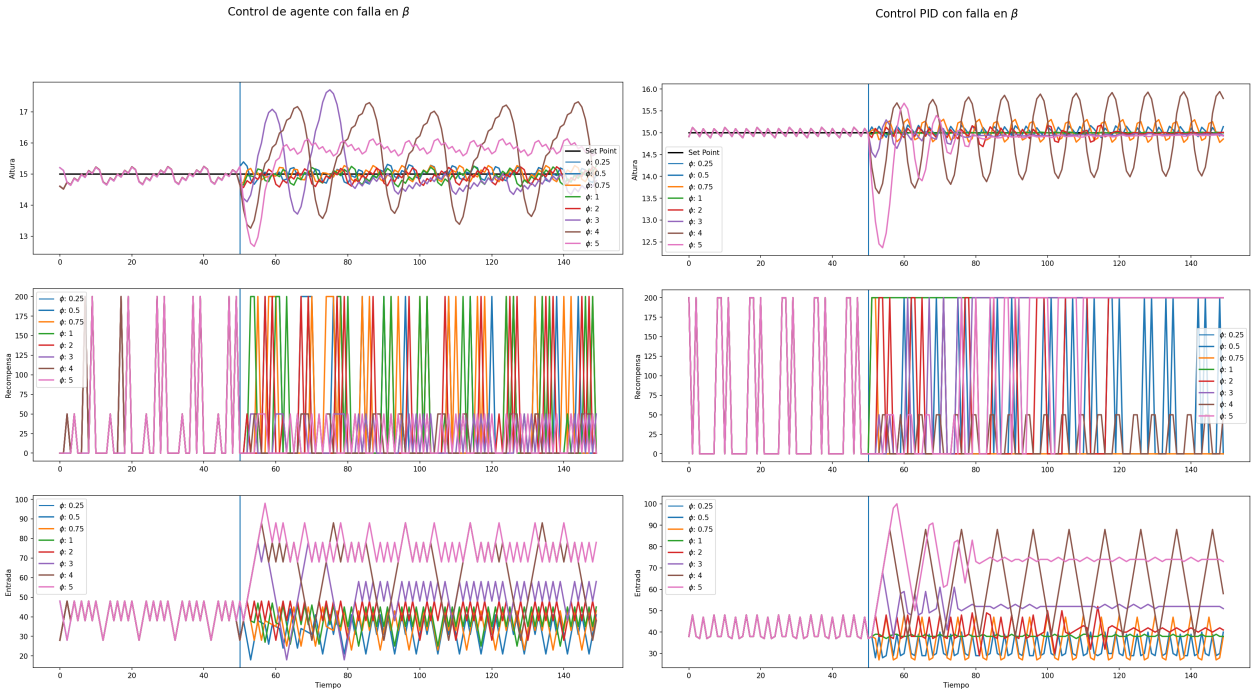
Se harán variar los valores nominales presentados en la Tabla 2.1 en razones incrementales desde un cuarto del valor nominal hasta 6 veces el valor nominal mediante un multiplicador ϕ, viendo como se comportan los controladores a distintas alturas.

4.4.2.1. Cambio en β

β controla la cantidad de agua que sale del sistema en determinado intervalo de tiempo, por lo que determina la frecuencia necesaria a utilizar dada cierta altura, y en algunos casos la incapacidad de la bomba en controlar el sistema. Un mayor β implica mayor frecuencia para entrar en el sistema suficiente agua para suplir la que se está retirando con el aumento, en algunos casos no es posible suplir este delta, siendo incontrolable el sistema, hasta que este llega a una altura donde se mantiene estable con la bomba funcionando a su frecuencia máxima.

La Figura 4.11 muestra como para valores bajos se generan oscilaciones en torno al punto de referencia, produciéndose debido a que el estanque se vacía más lento de lo que se ha visto en la data anteriormente, este *missmatch* de información hace que el agente no sea capaz de controlar el sistema, realizando acciones que lo alejan del punto de referencia hasta

que llega a un estado en el cual sabe como comportaste. Las oscilaciones que se aprecian en la Figura 4.11 se atenúan a medida que aumenta la altura de referencia. Para el control PID se atenúan de forma significativa a los veinte centímetros de altura, pero empeoran considerablemente, comparado con el agente, para alturas menores a quince centímetros. En contraste el control mediante agente REM-DQN empieza a atenuar estas oscilaciones a 40 centímetros de altura, demostrando que tiene problemas con valores de $\phi > 3$. Es interesante ver que bajo este margen el agente es capaz de controlar en un intervalo generoso en torno al punto de referencia.



(a) Seguimiento mediante agente REM-DQN.

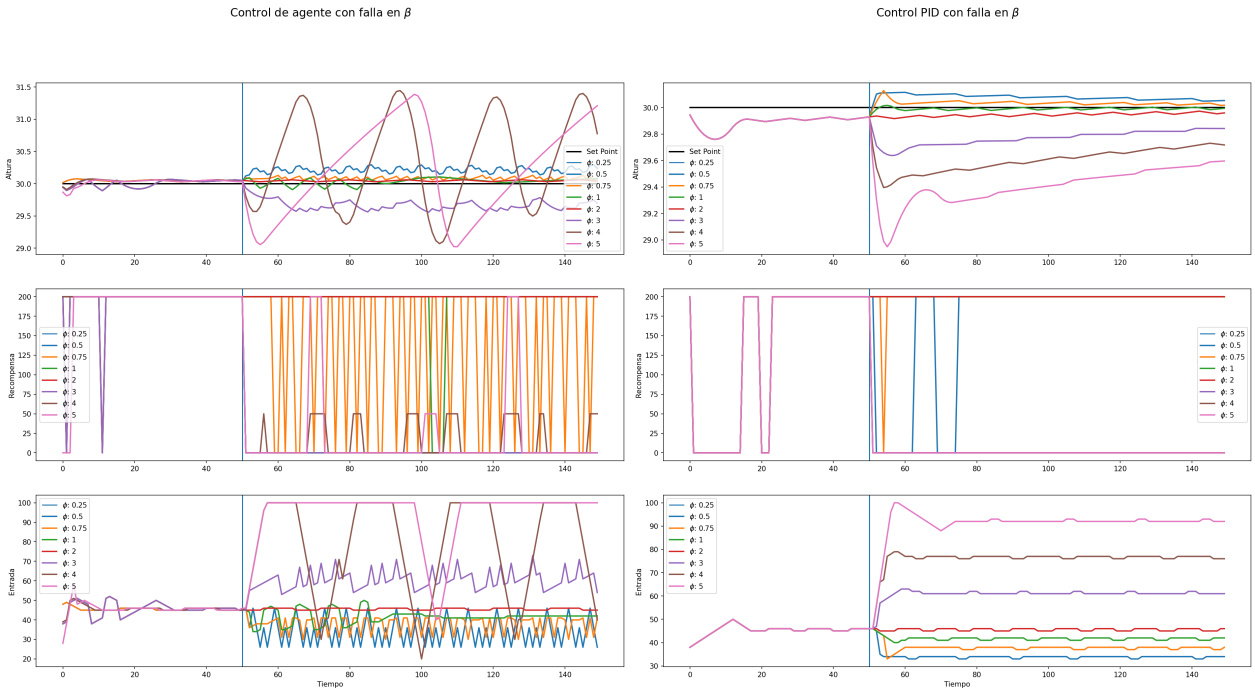
(b) Seguimiento mediante PID

Figura 4.11: Seguimiento de referencia a 15cm de altura.

La Figura 4.12 muestra el comportamiento de los controladores para alturas mayores a veinte centímetros. Como se puede observar continúan las oscilaciones del agente mencionadas anteriormente y se ve con mayor claridad el desfase provocado por el cambio de β. Se observa que el sistema no se puede controlar para $\phi = 5$.

4.4.2.2. Cambio en c_1

c_1 modula la frecuencia de la bomba, una variación en esta cambia el porcentaje de frecuencia necesaria para estabilizarse en un punto. Por lo anterior, es que mientras menor sea c_1 se requerirá mayor variación de frecuencia para lograr controlar el sistema y, mientras menor sea, menor variación se necesitará. Los resultados obtenidos son congruentes con esto, para $\phi > 0.5$ no es posible controlar el sistema y para $\phi = 0.75$ la altura no puede subir más de 49 centímetros con la bomba a máxima potencia.



(a) Seguimiento mediante agente REM-DQN.

(b) Seguimiento mediante PID

Figura 4.12: Seguimiento de referencia a 30cm de altura.

La Figura 4.13 muestra como se comportan los controladores a alturas bajas, menores a veinticinco centímetros, donde el control PID comienza a estabilizarse, mientras en control mediante el agente se mantiene oscilante. Debido a un fenómeno similar al ocurrido con la variación de beta, pero esta vez en el sentido contrario. Es posible ver como $\phi = 0.7$ tiene un mayor periodo oscilatorio y este va subiendo a medida que sube la altura, ya que para mayores alturas se necesita mayor variación en la entrada (o más prolongada si no se puede variar más) para poder generar la variación en altura.

La Figura 4.14 muestra como se comportan los controladores con la tarea de seguir la referencia, como se ve llega un punto donde el control PID es capaz de estabilizarse mientras el seguimiento realizado por el agente sigue siendo ondulatorio. Si bien se demuestra que no ocurrirán errores catastróficos mediante el funcionamiento del agente, estos resultados pueden tener una mejoría, en particular esto podría venir en forma de un set de datos que permita a la red conocer formas anómalas de funcionamiento del sistema, tal vez no llevadas al extremo, pero dentro de los parámetros aceptables.

4.4.2.3. Cambio en c_2

c_2 tiene que ver con el peso de la columna de agua sobre la bomba, el cual hay que sobrepasar para empezar a entregar agua al sistema, por lo que mientras aumenta c_2 se realiza una transición de una facilidad para suplir la demanda de agua con menores frecuencias a una necesidad de mayor frecuencia para suplir la necesidad de agua. En casos extremos, cuando $\phi > 3$ el sistema no es controlable, ya que se necesita mayor frecuencia de la disponible para realizar el control.

(a) Seguimiento mediante agente REM-DQN.

(b) Seguimiento mediante PID

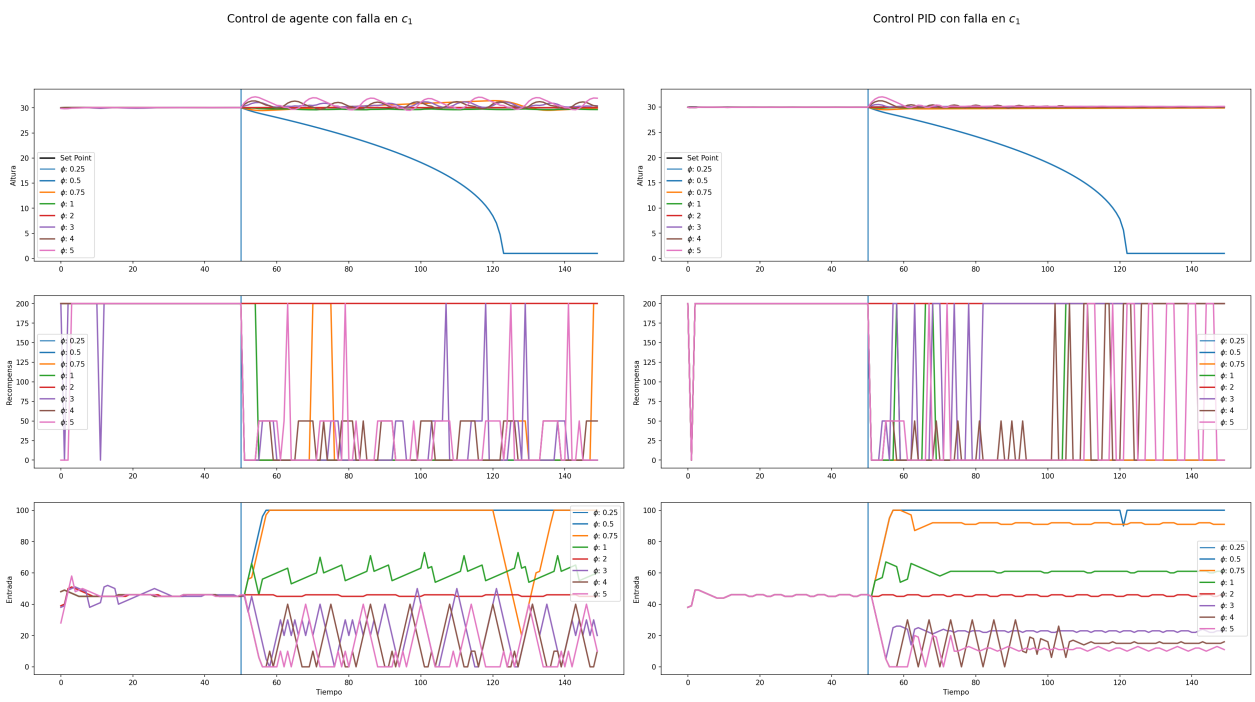
Figura 4.13: Seguimiento de referencia a 15cm de altura.

En la Figura 4.15 se puede observar el comportamiento de los controladores durante prácticamente todo el rango de alturas. Si bien los controladores tienen peores resultados en rangos bajo diez centímetros, esto es propio de los controladores, de la naturaleza altamente no lineal en esta sección y no tiene que ver con la falla en cuestión. Se observa la imposibilidad de controlar para $\phi > 3$ y que ambos controladores tienen problemas con $\phi = 3$, siendo el control PID el que logra un mejor control, sin ser capaz de entrar a la zona donde se obtiene recompensa máxima. El control de seguimiento realizado por el agente se mantiene oscilante en torno al punto de referencia sin ser capaz de mantenerse cerca de este. Lo anterior da indicios que el control bajo un problema de tal magnitud no tendría problemas sustanciales, pero operaria de forma sub-óptima.

4.4.2.4. Cambio en α

La variación de α no resultó en cambios significativos en la forma de controlar para valores altos, esto se explica fácilmente separando α de su multiplicador y sacando este fuera de la ecuación diferencial, quedando $h_n\dot{e}w = \dot{h} * \phi$ explicando que la nueva dinámica del sistema varía con ϕ, por lo que si ϕ disminuye el sistema cambia más lento, permitiendo mayor control, y de forma contraria si ϕ aumenta el sistema cambia más rápido, por ende, más difícil de controlar.

La Figura 4.16 muestra el comportamiento de ambos controladores realizando el seguimiento de referencia para una altura de treinta centímetros, lo cual se usa para ejemplificar alturas mayores a veinte centímetros. Se puede observar que ambos controlan bien el sistema, obteniendo el máximo puntaje en casi toda la instancia, además se puede observar que mientras mayor es el valor de ϕ son más recurrente las oscilaciones y viceversa, demostrado



(a) Seguimiento mediante agente REM-DQN. (b) Seguimiento mediante PID

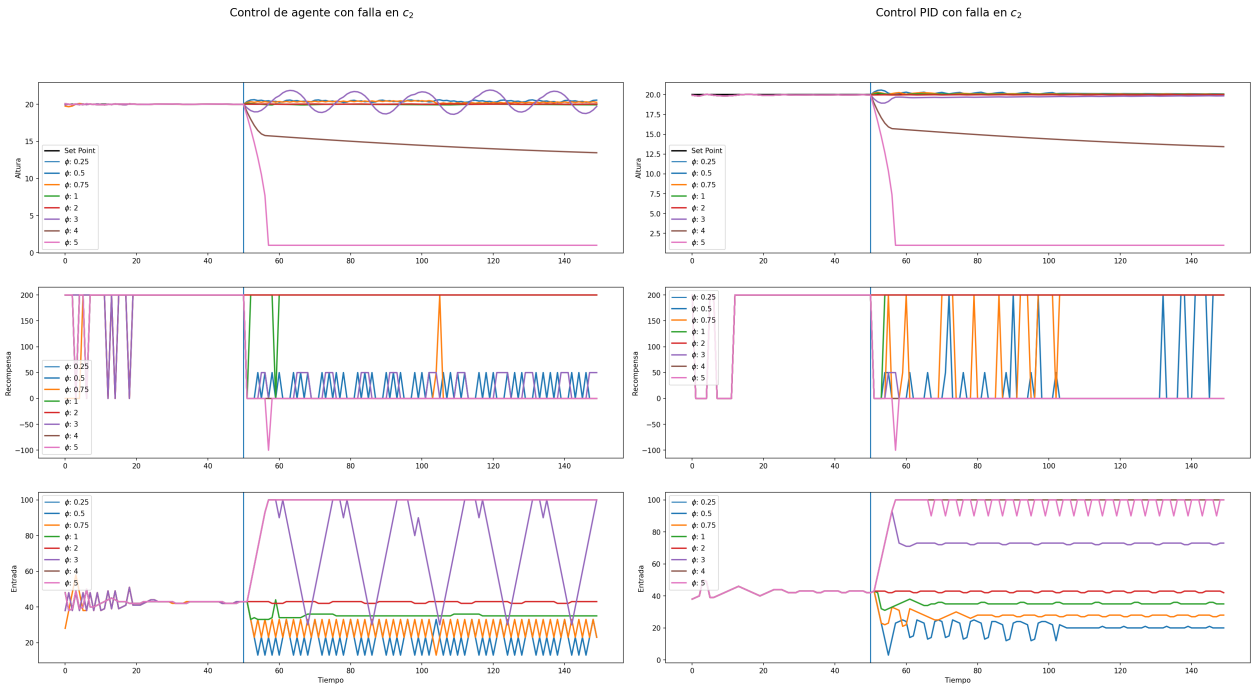
Figura 4.14: Seguimiento de referencia a 30cm de altura.

de mejor manera por las líneas rosada y azul ($\phi = 5$ y $\phi = 0.5$).

La Figura 4.17 muestra como reaccionan los controladores frente a la eventualidad en una altura de menos de veinte centímetros. Es posible observar que el controlador PID tiene problemas con los valores altos, siendo esto más notorio con $\phi = 5$ o $\phi = 4$ donde se nota una oscilación en torno al punto de equilibrio. Para el resto de los ϕ el controlador es capaz de mantener la altura dentro de un rango adecuado, apenas saliendo del rango de recompensa máxima. Por el otro lado el seguimiento de referencia mediante el agente REM-DQN comienza a tener problemas serios cuando el multiplicador alcanza el triple, debido en parte porque este agente no es muy bueno controlando en esta altura y esto se ve exacerbado por el fenómeno discutido anteriormente. Se puede observar que para $\phi < 3$ se sigue el mismo control que se tiene antes de la falla, pero sobre eso comienza a tener oscilaciones importantes.

4.4.3. Análisis del desempeño

El análisis de la presente sección se hace con la idea de llegar a comparar el desempeño de un agente entrenado mediante el uso de información de operación sub óptima. Bajo esta lógica se obvia el uso de un PID y se compara su utilidad frente a políticas de uso subóptimas. El Agente es capaz de aprender una política operacional que le permite obtener mejores resultados netos en la función de recompensa utilizada, a costo del esfuerzo de control realizado. Lo anterior puede ser mitigado si se cambia la función de recompensa para incorporar la intención de minimizar a su vez el esfuerzo de control, para aprendizaje reforzado online es necesario tunear bien este parámetro debido al dilema de la exploración versus la explotación, ya que al explorar se realizan cambios bruscos al intentar explorar distintas formas



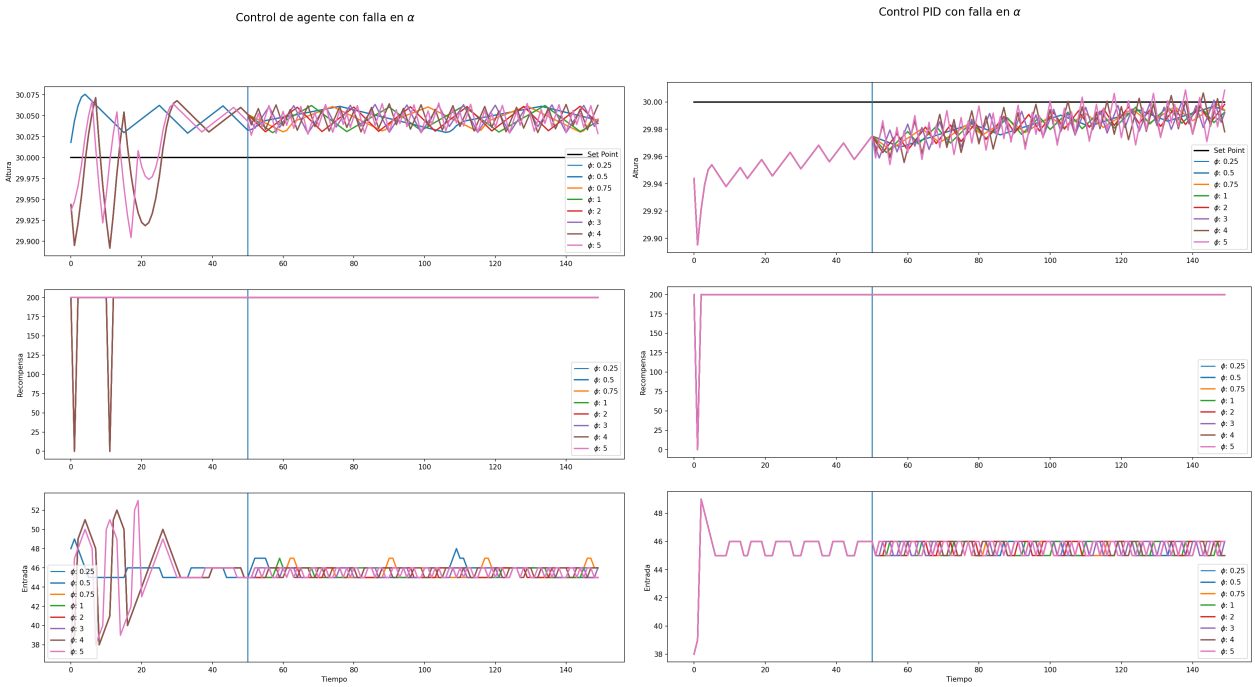
(a) Seguimiento mediante agente REM-DQN.

(b) Seguimiento mediante PID

Figura 4.15: Seguimiento de referencia a 20cm de altura.

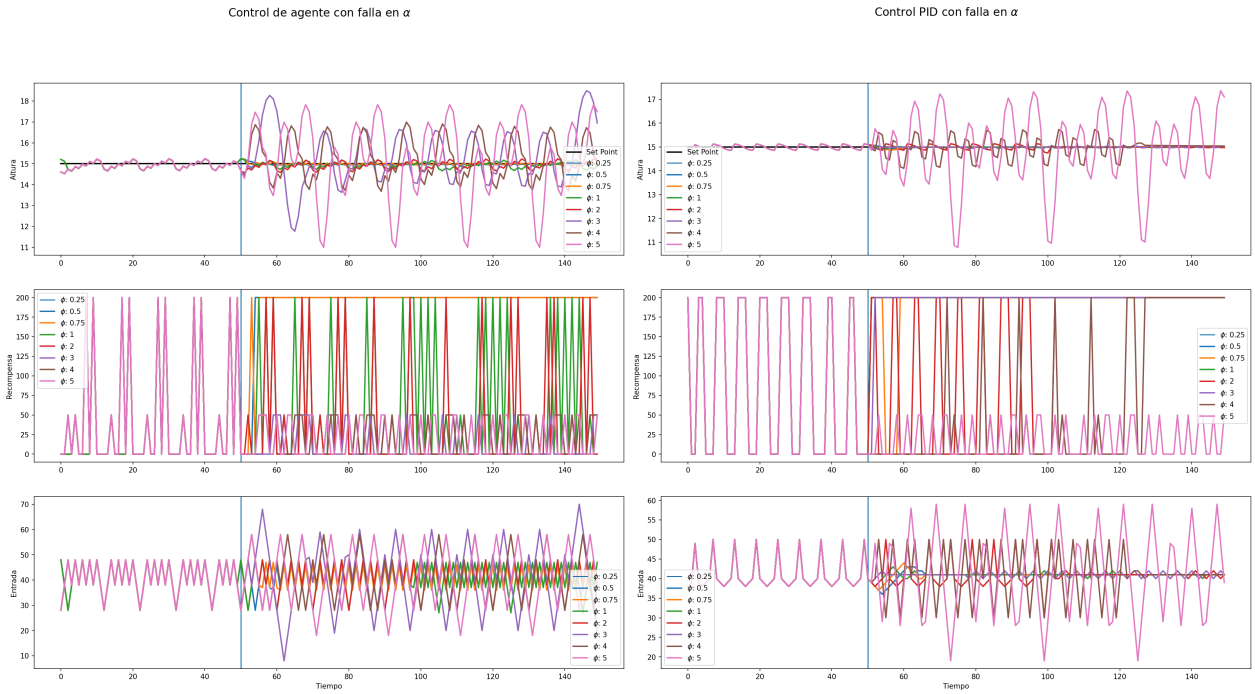
de operar [56]. Ya que el aprendizaje fuera de línea no necesita explorar datos, ese tipo de problemáticas no se aplican.

La capacidad del algoritmo de adaptarse a fallas en el sistema es satisfactoria. Si bien no es capaz de seguir la referencia en casos extremos se mantiene oscilante en torno a la referencia, esto muestra la capacidad de resistir eventos verdaderamente catastróficos donde el control se aleje demasiado del punto de referencia. Es posible mejorar los resultados del agente si se le entrega data de como se ha comportado anteriormente frente a ese estado.



(a) Seguimiento mediante agente REM-DQN. (b) Seguimiento mediante PID

Figura 4.16: Seguimiento de referencia a 30cm de altura.



(a) Seguimiento mediante agente REM-DQN. (b) Seguimiento mediante PID

Figura 4.17: Seguimiento de referencia a 15cm de altura.

5. Conclusiones

El aprendizaje fuera de línea es un concepto relativamente nuevo y prometedor, en la presente investigación se intentó acercar esta nueva tecnología a un caso de control específico con un ambiente típico para control no lineal, usando métodos de control imperfectos. La metodología presentada muestra que es posible aplicar aprendizaje reforzado fuera de línea a sistemas de control reales con buenos resultados. Se muestra que es posible la aplicación de aprendizaje reforzado fuera de línea a problemas de control discreto con la necesidad de una base de datos que soporte el aprendizaje y una función de mérito que plasme los objetivos específicos del problema que se lleva a cabo.

En cuanto a los objetivos, se cumple el objetivo general con cada uno de los agentes entrenados y en función del objetivo general se cumplen los objetivos específicos, lográndose agentes capaces de controlar las diversas instancias.

Durante la sección del Caso de estudio se deja en evidencia la capacidad que tiene el método para poder adaptarse de manera satisfactoria a un control con distinta calidad en los datos de entrenamiento. Los datos recopilados mediante el entrenamiento del agente DQN usando la función de mérito **FPID** nos permite observar como el algoritmo REM-DQN es capaz de obtener información sobre como controlar al sistema, de mejor manera que el agente original (con el cual fue generada la base de datos). Esta mejoría se logra sin necesidad de obtener información adicional del ambiente además de las interacciones guardadas del agente DQN. Mediante el entrenamiento con la función de mérito **CRF** vemos que el agente REM-DQN es capaz de aprender del comportamiento de un agente cercano al óptimo y mejorar su desempeño.

Las interacciones con los datasets generados por los agentes DQN nos demuestran la capacidad que tiene el algoritmo REM-DQN de mejorar la política actual, buscando mejorías de esta en acciones anteriores sin necesidad de invertir en modelación o tiempo de interacción. Siempre y cuando existan suficientes transiciones de estado para poder generar una buena aproximación de la función de recompensa sin *distribution shift*. Resulta interesante el hecho que el agente fuera de linea pueda obtener mejores resultados que los presentes en el dataset de entrenamiento. Esto se debe a que el agente usado para generar el dataset de entrenamiento puede explorar nuevas transiciones de estado, mientras que el agente fuera de linea esta acotado a las transiciones de estado observadas en el dataset de entrenamiento.

Mediante los experimentos de desempeño y el comportamiento del controlador frente a eventualidades, se demuestra que el control realizado por el agente entrenado fuera de línea es resistente frente a perturbaciones sustanciales y se mantiene dentro de márgenes razonables cerca del valor de referencia para fallas catastróficas. Demostrando que el control es lo suficientemente robusto para controlar el sistema con un margen de error importante.

Si bien los resultados presentados dan prueba de la factibilidad de usar estos métodos, es importante remarcar que es importante testear la calidad del aprendizaje realizado. A priori no es posible saber si el aprendizaje realizado tuvo buenos resultados. Para suplir esta desventaja se puede utilizar en conjunto con operadores como sistema de recomendación. Dando una segunda opinión sobre como operar y guiando al agente a un entrenamiento satisfactorio.

En la actualidad se utilizan sistemas de recomendación para mejorar o disminuir la variabilidad de distintos procesos que no están bien mapeados.

Como trabajo futuro se propone utilizar la misma metodología sobre una base de datos de recomendaciones historicas. Identificando si esto supone una mejoría del sistema recomendador anterior y si estas le hacen más sentido a los operadores que las utilizan.

Bibliografía

[1] Neurips, "Acceptedpapersinitial, 2017," 2017, https://web.archive.org/web/20170914060103/nips.cc/Conferences/2017/AcceptedPapersInitial.

[2] Neurips, "Acceptedpapersinitial, 2018," 2018, https://neurips.cc/Conferences/2018/Schedule?type=Poster.

[3] Neurips, "Acceptedpapersinitial, 2019," 2019, https://web.archive.org/web/20190906013341/http://neurips.cc/Conferences/2019/AcceptedPapersInitial.

[4] Neurips, "Acceptedpapersinitial, 2020," 2020, https://neurips.cc/Conferences/2020/AcceptedPapersInitial.

[5] Ivanov, S., "Neurips 2020. comprehensive analysis of authors, organizations, and countries.," 2020, https://medium.com/criteo-engineering/neurips-2020-comprehensive-analysis-of-authors-organizations-and-countries-a1b55a08132e.

[6] Banking, G. y Finance, "Machine learning (ml) market size, share covid-19 impact analysis, by component (solution, and services), by enterprise size (smes, and large enterprises), by deployment (cloud and on-premise), by industry (healthcare, retail, it and telecommunication, bfsi, automotive and transportation, advertising and media, manufacturing, and others), and regional forecast, 2021-2028," 2020, https://www.fortunebusinessinsights.com/machine-learning-market-102226.

[7] Gajurel, A., Louis, S. J., Wu, R., Barford, L., y Harris, F. C., "Gpu acceleration of sparse neural networks," en ITNG 2021 18th International Conference on Information Technology-New Generations (Latifi, S., ed.), (Cham), pp. 323–330, Springer International Publishing, 2021.

[8] Dally, W. J., Keckler, S. W., y Kirk, D. B., "Evolution of the graphics processing unit (gpu)," IEEE Micro, vol. 41, no. 6, pp. 42–51, 2021, doi:10.1109/MM.2021.3113475.

[9] H., S. I., "Machine learning: Algorithms, real-world applications and research directions," SN Comput Sci., doi:10.1007/s42979-021-00592-x.

[10] Levine, S., Kumar, A., Tucker, G., y Fu, J., "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," CoRR, vol. abs/2005.01643, 2020, https://arxiv.org/abs/2005.01643.

[11] Jaques, N., Ghandeharioun, A., Shen, J. H., Ferguson, C., Lapedriza, A., Jones, N., Gu, S., y Picard, R., "Way off-policy batch deep reinforcement learning of implicit human preferences in dialog," 2019, doi:10.48550/ARXIV.1907.00456.

[12] Ebert, F., Finn, C., Dasari, S., Xie, A., Lee, A., y Levine, S., "Visual foresight: Model-based deep reinforcement learning for vision-based robotic control," 2018, doi:10.48550

/ARXIV.1812.00568.

[13] Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., y Levine, S., "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," 2018, doi:10.48550/ARXIV.1806.10293.

[14] Kahn, G., Abbeel, P., y Levine, S., "Badgr: An autonomous self-supervised learning-based navigation system," 2020, doi:10.48550/ARXIV.2002.05700.

[15] Zhan, X., Xu, H., Zhang, Y., Huo, Y., Zhu, X., Yin, H., y Zheng, Y., "Deepthermal: Combustion optimization for thermal power generating units using offline reinforcement learning," 2021, doi:10.48550/ARXIV.2102.11492.

[16] Ai, M., Xie, Y., Tang, Z., Zhang, J., y Gui, W., "Deep learning feature-based setpoint generation and optimal control for flotation processes," Information Sciences, vol. 578, pp. 644–658, 2021, doi:https://doi.org/10.1016/j.ins.2021.07.060.

[17] Gawłowicz, P. y Zubow, A., "ns3-gym: Extending openai gym for networking research," 2018, doi:10.48550/ARXIV.1810.03943.

[18] Stančin, I. y Jović, A., "An overview and comparison of free python libraries for data mining and big data analysis," en 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 977–982, 2019, doi:10.23919/MIPRO.2019.8757088.

[19] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., y Chintala, S., "Pytorch: An imperative style, high-performance deep learning library," en Advances in Neural Information Processing Systems (Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., y Garnett, R., eds.), vol. 32, Curran Associates, Inc., 2019, https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

[20] Åström, K. y Hägglund, T., Control PID avanzado. Pearson, Madrid, 2009.

[21] Knospe, C., "Pid control," IEEE Control Systems Magazine, vol. 26, no. 1, pp. 30–31, 2006.

[22] Johnson, M. A. y Moradi, M. H., PID control. Springer, 2005.

[23] Åström, K. J. y Hägglund, T., "Pid control," IEEE Control Systems Magazine, vol. 1066, 2006.

[24] Swaminathan, A., Krishnamurthy, A., Agarwal, A., Dudik, M., Langford, J., Jose, D., y Zitouni, I., "Off-policy evaluation for slate recommendation," en Advances in Neural Information Processing Systems (Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., y Garnett, R., eds.), vol. 30, Curran Associates, Inc., 2017, https://proceedings.neurips.cc/paper/2017/file/5352696a9ca3397beb79f116f3a33991-Paper.pdf.

[25] Sutton, R. S. y Barto, A. G., Reinforcement learning: An introduction. MIT press, 2018.

[26] Bellman, R., "A markovian decision process," Journal of mathematics and mechanics, pp. 679–684, 1957.

[27] Sutton, R. S. y Barto, A. G., "Introduction to reinforcement learning. mit press," Cam-

bridge, MA, 1998.

[28] Thomas, P., "Bias in natural actor-critic algorithms," en Proceedings of the 31st International Conference on Machine Learning (Xing, E. P. y Jebara, T., eds.), vol. 32 de Proceedings of Machine Learning Research, (Bejing, China), pp. 441–448, PMLR, 2014, https://proceedings.mlr.press/v32/thomas14.html.

[29] Dabney, W., Ostrovski, G., y Barreto, A., "Temporally-extended ϵ-greedy exploration," CoRR, vol. abs/2006.01782, 2020, https://arxiv.org/abs/2006.01782.

[30] Afifi, H. y Karl, H., "Reinforcement learning for virtual network embedding in wireless sensor networks," en 2020 16th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 123–128, 2020, doi: 10.1109/WiMob50308.2020.9253442.

[31] Sutton, R. S., McAllester, D., Singh, S., y Mansour, Y., "Policy gradient methods for reinforcement learning with function approximation," Advances in neural information processing systems, vol. 12, 1999.

[32] Kakade, S. M., "A natural policy gradient," Advances in neural information processing systems, vol. 14, 2001.

[33] Schulman, J., Levine, S., Abbeel, P., Jordan, M., y Moritz, P., "Trust region policy optimization," en International conference on machine learning, pp. 1889–1897, PMLR, 2015.

[34] Cormen, T. H., Leiserson, C. E., Rivest, R. L., y Stein, C., Introduction to Algorithms, Third Edition. The MIT Press, 3rd ed., 2009.

[35] Bellman, R., "Dynamic programming. princeton, nj: Princeton universitypress," BellmanDynamic Programming, 1957.

[36] Fu, J., Kumar, A., Soh, M., y Levine, S., "Diagnosing bottlenecks in deep q-learning algorithms," en International Conference on Machine Learning, pp. 2021–2030, PMLR, 2019.

[37] Tassa, Y., Erez, T., y Todorov, E., "Synthesis and stabilization of complex behaviors through online trajectory optimization," en 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4906–4913, IEEE, 2012.

[38] Nagabandi, A., Kahn, G., Fearing, R. S., y Levine, S., "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," en 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 7559–7566, IEEE, 2018.

[39] Chua, K., Calandra, R., McAllister, R., y Levine, S., "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," Advances in neural information processing systems, vol. 31, 2018.

[40] Gu, S., Holly, E., Lillicrap, T. P., y Levine, S., "Deep reinforcement learning for robotic manipulation," CoRR, vol. abs/1610.00633, 2016, http://arxiv.org/abs/1610.00633.

[41] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., y Riedmiller, M., "Playing atari with deep reinforcement learning," arXiv preprint arXiv:1312.5602, 2013.

[42] 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, vol. 518, no. 7540, pp. 529–533, 2015.

[43] Agarwal, R., Schuurmans, D., y Norouzi, M., "An optimistic perspective on offline reinforcement learning," en International Conference on Machine Learning, 2020.

[44] Ernst, D., Geurts, P., y Wehenkel, L., "Tree-based batch mode reinforcement learning," J. Mach. Learn. Res., vol. 6, p. 503–556, 2005.

[45] Fujimoto, S., Meger, D., y Precup, D., "Off-policy deep reinforcement learning without exploration," 2018, doi:10.48550/ARXIV.1812.02900.

[46] Kumar, A., Fu, J., Tucker, G., y Levine, S., "Stabilizing off-policy q-learning via bootstrapping error reduction," 2019, doi:10.48550/ARXIV.1906.00949.

[47] Cabi, S., Colmenarejo, S. G., Novikov, A., Konyushkova, K., Reed, S., Jeong, R., Zolna, K., Aytar, Y., Budden, D., Vecerik, M., *et al.*, "Scaling data-driven robotics with reward sketching and batch reinforcement learning," arXiv preprint arXiv:1909.12200, 2019.

[48] Fujimoto, S., Conti, E., Ghavamzadeh, M., y Pineau, J., "Benchmarking batch deep reinforcement learning algorithms," 2019, doi:10.48550/ARXIV.1910.01708.

[49] Papers with code, l., "Atari games on atari 2600 breakout," 2021-04, https://paperswithcode.com/sota/atari-games-on-atari-2600-breakout.

[50] Fu, J., Kumar, A., Nachum, O., Tucker, G., y Levine, S., "D4rl: Datasets for deep data-driven reinforcement learning," arXiv preprint arXiv:2004.07219, 2020.

[51] Seno, T. y Imai, M., "d3rlpy: An offline deep reinforcement learning library," arXiv preprint arXiv:2111.03788, 2021.

[52] Krizhevsky, A., Hinton, G., *et al.*, "Learning multiple layers of features from tiny images," Citeseer, 2009.

[53] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., y Fei-Fei, L., "ImageNet Large Scale Visual Recognition Challenge," International Journal of Computer Vision (IJCV), vol. 115, no. 3, pp. 211–252, 2015, doi:10.1007/s11263-015-0816-y.

[54] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., y Fei-Fei, L., "Imagenet: A large-scale hierarchical image database," en 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255, 2009, doi:10.1109/CVPR.2009.5206848.

[55] Nagahara, M., Quevedo, D. E., y Nešić, D., "Maximum hands-off control: A paradigm of control effort minimization," IEEE Transactions on Automatic Control, 2016.

[56] Oestreich, C. E., Linares, R., y Gondhalekar, R., "Autonomous six-degree-of-freedom spacecraft docking maneuvers via reinforcement learning," arXiv preprint arXiv:2008.03215, 2020.