La investigación reportada en esta tesis es parte de los programas de investigación del CICESE (Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California).
La investigación fue financiada por el CONAHCYT (Consejo Nacional de Humanidades, Ciencias y Tecnologías).

Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California

Maestría en Ciencias
en Electrónica y Telecomunicaciones

Control de sistemas usando aprendizaje de máquina

Tesis

para cubrir parcialmente los requisitos necesarios para obtener el grado de

Maestro en Ciencias

Presenta:

Jesús Martín Miguel Martínez

Ensenada, Baja California, México

2023

Tesis defendida por

Jesús Martín Miguel Martínez

y aprobada por el siguiente Comité

Dr. Carlos Alberto Brizuela Rodríguez

Codirector de tesis

Dr. Luis Alejandro Márquez Martínez

Codirector de tesis

Dr. Cesar Raúl García Jacas

Dr. José Ricardo Cuesta García

Dra. María del Carmen Maya Sánchez

Coordinadora del Posgrado en Electrónica y Telecomunicaciones

Dra. Ana Denise Re Araujo

Directora de Estudios de Posgrado

Resumen de la tesis que presenta Jesús Martín Miguel Martínez como requisito parcial para la obtención del grado de Maestro en Ciencias en Electrónica y Telecomunicaciones.

Control de sistemas usando aprendizaje de máquina

Resumen aprobado por:

Dr. Carlos Alberto Brizuela Rodríguez	Dr. Luis Alejandro Márquez Martínez
Codirector de tesis	Codirector de tesis

El aprendizaje por refuerzo es un paradigma del aprendizaje de máquina con un amplio desarrollo y una creciente demanda en aplicaciones que involucran toma de decisiones y control. Es un paradigma que permite el diseño de controladores que no dependen directamente del modelo que describe la dinámica del sistema. Esto es importante ya que en aplicaciones reales es frecuente que no se disponga de dichos modelos de manera precisa. Esta tesis tiene como objetivo implementar un controlador óptimo en tiempo discreto libre de modelo. La metodología elegida se basa en algoritmos de aprendizaje por refuerzo, enfocados en sistemas con espacios de estado y acción continuos a través de modelos discretos. Se utiliza el concepto de función de valor (Q-función y función V) y la ecuación de Bellman para resolver el problema del regulador cuadrático lineal para un sistema mecánico masa-resorte-amortiguador, en casos donde se tiene conocimiento parcial y desconocimiento total del modelo. Para ambos casos las funciones de valor son definidas explícitamente por la estructura de un aproximador paramétrico, donde el vector de pesos del aproximador es sintonizado a través de un proceso iterativo de estimación de parámetros. Cuando se tiene conocimiento parcial de la dinámica se usa el método de aprendizaje por diferencias temporales en un entrenamiento episódico, que utiliza el esquema de mínimos cuadrados con mínimos cuadrados recursivos en la sintonización del crítico y descenso del gradiente en la sintonización del actor, el mejor resultado para este esquema es usando el algoritmo de iteración de valor para la solución de la ecuación de Bellman, con un resultado significativo en términos de precisión en comparación a los valores óptimos (función `DLQR`). Cuando se tiene desconocimiento de la dinámica se usa el algoritmo Q-learning en entrenamiento continuo, con el esquema de mínimos cuadrados con mínimos cuadrados recursivos y el esquema de mínimos cuadrados con descenso del gradiente. Ambos esquemas usan el algoritmo de iteración de política para la solución de la ecuación de Bellman, y se obtienen resultados de aproximadamente 0.001 en la medición del error cuadrático medio. Se realiza una prueba de adaptabilidad considerando variaciones que puedan suceder en los parámetros de la planta, siendo el esquema de mínimos cuadrados con mínimos cuadrados recursivos el que tiene los mejores resultados, reduciendo significativamente el número de iteraciones necesarias para la convergencia a valores óptimos.

Palabras clave: aprendizaje por refuerzo, control óptimo, control adaptativo, sistemas mecánicos, libre de modelo, dinámica totalmente desconocida, aproximación paramétrica, Q-learning, iteración de política

Abstract of the thesis presented by Jesús Martín Miguel Martínez as a partial requirement to obtain the Master of Science degree in Electronics and Telecommunications.

Systems control using machine learning

Abstract approved by:

Dr. Carlos Alberto Brizuela Rodríguez
Thesis Co-Director

Dr. Luis Alejandro Márquez Martínez
Thesis Co-Director

Reinforcement learning is a machine learning paradigm with extensive development and growing demand in decision-making and control applications. This technique allows the design of controllers that do not directly depend on the model describing the system dynamics. It is useful in real-world applications, where accurate models are often unavailable. The objective of this work is to implement a model-free discrete-time optimal controller. Through discrete models, we implemented reinforcement learning algorithms focused on systems with continuous state and action spaces. The concepts of value-function, Q-function, V-function, and the Bellman equation are employed to solve the linear quadratic regulator problem for a mass-spring-damper system in a partially known and utterly unknown model. For both cases, the value functions are explicitly defined by a parametric approximator's structure, where the weight vector is tuned through an iterative parameter estimation process. When partial knowledge of the dynamics is available, the temporal difference learning method is used under episodic training, utilizing the least squares with a recursive least squares scheme for tuning the critic and gradient descent for the actor´s tuning. The best result for this scheme is achieved using the value iteration algorithm for solving the Bellman equation, yielding significant improvements in approximating the optimal values (`DLQR` function). When the dynamics are entirely unknown, the Q-learning algorithm is employed in continuous training, employing the least squares with recursive least squares and the gradient descent schemes. Both schemes use the policy iteration algorithm to solve the Bellman equation, and the system's response using the obtained values was compared to the one using the theoretical optimal values, yielding approximately zero mean squared error between them. An adaptability test is conducted considering variations that may occur in plant parameters, with the least squares with recursive least squares scheme yielding the best results, significantly reducing the number of iterations required for convergence to optimal values.

Keywords: reinforcement learning, optimal control, adaptive control, mechanical systems, model-free, utterly unknown dynamics, parametric approximation, Q-learning, policy iteration

Dedicatoria

A YHWH y a mi madre.

Agradecimientos

Al CICESE (Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California) por darme la oportunidad realizar mis estudios de posgrado.

Al CONAHCYT (Consejo Nacional de Humanidades, Ciencias y Tecnologías) por brindarme el apoyo económico para realizar mis estudios de maestría.

A mis directores de tesis, el Dr. Carlos Alberto Brizuela Rodríguez por su eterna paciencia y su capacidad de hacerme pensar más allá de la obviedad, pero sobre todo por darme la confianza y aceptar este reto, y el Dr. Luis Alejandro Márquez Martínez por su confianza y sus instrucciones constantes para ser un mejor estudiante, por aceptarme como su asesorado, pero sobre todo por su sinceridad en todas y cada una de sus correcciones a lo largo de la maestría, a ambos entero y eternamente agradecido.

A los miembros que integran el comité evaluador de esta tesis, el Dr. José Ricardo Cuesta García y el Dr. Cesar Raúl García Jacas por el apoyo y disposición constante, también por las contribuciones en la realización de la tesis.

A mi madre †...

A mis tías, Irma y Rosa, por darme el cariño y todo el apoyo que puede venir del alma, para ustedes, por lo que representan en mi vida.

A mi familia, por todo las enseñanzas y amor brindado a lo largo de mi vida, gracias por nunca dejarme. A todos mis primos y hermanos; José, Francisco, Luis, Mayra y Dany.

A mis compañeros dentro del CICESE, por siempre darme el apoyo necesario en cada una de mis dudas, en especial al grupo de control, mi amiga Diana y mi amigo Ángel, la gente más talentosa que he conocido, unos verdaderos animales.

A mi amigo Isra, por sus conocimientos compartidos y por ser un apoyo para concluir este trabajo.

A todos mis profesores del departamento de Electrónica y Telecomunicaciones con orientación en Control, por compartir sus conocimientos acerca de tan bella disciplina de manera entusiasta y sincera, al Dr. Jonatán Peña, al Dr. Javier Pliego y en especial al Mtro. Ricardo Núñez.

Y más importante aún, a *YHWH*, por todo lo que hiciste por mí y que no pude ver.

"... y sabrá su majestad que yendo por occidente y regresando por oriente, hemos descubierto y circunnavegado el mundo entero".

Juan Sebastián Elcano

Tabla de contenido

Página

Resumen en español ... ii

Resumen en inglés ... iii

Dedicatoria ... iv

Agradecimientos ... v

Lista de figuras ... ix

Lista de tablas ... xiii

Capítulo 1. Introducción

1.1. Definiciones básicas ... 2
1.2. Antecedentes y motivación ... 5
1.3. Definición del problema ... 9
1.4. Objetivos ... 9
1.4.1. Objetivo general ... 9
1.4.2. Objetivos específicos ... 10
1.5. Metodología de trabajo ... 10
1.6. Estructura de la tesis ... 11

Capítulo 2. Marco teórico

2.1. Proceso de decisión de Markov ... 13
2.1.1. Modelo de optimalidad y retornos ... 15
2.1.2. Funciones de valor y ecuaciones de Bellman ... 16
2.2. Programación dinámica: algoritmos dependientes del modelo ... 19
2.2.1. Iteración de valor ... 20
2.2.2. Iteración de política ... 21
2.2.3. Programación dinámica determinista ... 22
2.2.3.1. Análisis no lineal en tiempo discreto ... 24
2.2.3.2. Control óptimo: LQR en tiempo discreto ... 25
2.2.3.3. Q-función para el LQR en tiempo discreto ... 28
2.3. Aprendizaje por refuerzo: algoritmos libres de modelo ... 29
2.3.1. Aprendizaje por diferencias temporales ... 29
2.3.2. Algoritmo Q-learning para control óptimo adaptativo ... 31
2.3.3. Actor-crítico ... 32
2.3.4. Aproximación de la función de valor ... 34
2.3.4.1. Aproximadores paramétricos ... 35
2.3.4.2. Aproximación de la Q-función a través de la iteración ... 36
2.3.5. Estimación de parámetros ... 38
2.3.5.1. El modelo ... 38
2.3.5.2. Mínimos cuadrados ordinarios ... 39
2.3.5.3. Mínimos cuadrados recursivos ... 40
2.3.5.4. Descenso del gradiente ... 41
2.3.5.5. Excitación persistente del sistema ... 42
2.4. Iteración de valor e iteración de política para el problema del LQR en tiempo discreto 43

2.4.1. Iteración de valor con TD . . . 43
2.4.2. Iteración de política con TD . . . 44
2.4.3. Iteración de política con Q-learning . . . 44

Capítulo 3. Metodología
3.1. Formulación del modelo en tiempo discreto . . . 45
3.1.1. Sistemas de estudio . . . 47
3.1.1.1. Caso 1. Sistema lineal estable de segundo orden . . . 48
3.1.1.2. Caso 2. Sistema lineal inestable de segundo orden . . . 49
3.2. Control óptimo adaptativo para un sistema con dinámica parcialmente conocida . . . 50
3.2.1. Aproximación de la función de valor usando conjunto base polinomial para el crítico . . . 50
3.2.2. Iteración de valor usando aprendizaje por diferencias temporales en línea . . . 52
3.2.3. Iteración de política usando aprendizaje por diferencias temporales en línea 60
3.3. Control óptimo para un sistema con dinámica totalmente desconocida . . . 63
3.3.1. Iteración de política para algoritmo de Q-learning en línea . . . 63
3.4. Sumario . . . 66

Capítulo 4. Simulaciones numéricas y resultados
4.1. Especificaciones de hardware y software . . . 70
4.2. Dinámica parcialmente conocida . . . 70
4.2.1. Ejemplo 1: Sistema estable usando el Algoritmo 3 . . . 71
4.2.2. Ejemplo 2: Sistema inestable usando el Algoritmo 3 . . . 73
4.2.3. Ejemplo 3: Sistema estable usando el Algoritmo 4 . . . 75
4.2.4. Ejemplo 4: Sistema inestable usando el Algoritmo 4 . . . 77
4.2.5. Comparación de resultados con entrenamiento continuo . . . 79
4.3. Dinámica totalmente desconocida . . . 84
4.3.1. Ejemplo 5: Sistema estable usando el Algoritmo 5 (LS+GD) . . . 84
4.3.2. Ejemplo 6: Sistema estable usando el Algoritmo 5 (LS+RLS) . . . 87
4.4. Prueba de adaptabilidad a variaciones con dinámica totalmente desconocida . . . 90
4.4.1. Ejemplo 7: Sistema estable con dos variaciones (LS+GD) . . . 91
4.4.2. Ejemplo 8: Sistema estable con dos variaciones (LS+RLS) . . . 92

Capítulo 5. Discusión

Capítulo 6. Conclusiones
6.1. Conclusiones generales . . . 96
6.2. Aportaciones realizadas . . . 97
6.3. Trabajo futuro . . . 97

Literatura citada . . . 99

Anexos . . . 103

Lista de figuras

Figura Página

1. Flujo de señales en un sistema. 2
2. Esquema de un sistema de control clásico en lazo abierto. Diagrama tomado y modificado de Miranda (2012). 2
3. Esquema de un sistema de control clásico en lazo cerrado con realimentación de estado. Diagrama tomado y modificado de Miranda (2012). 3
4. Configuración básica de control adaptativo de acuerdo a Rubio & Sánchez (1996). . . . 4
5. Esquema de control adaptativo con controlador autoajustable (STR) de acuerdo a Rubio & Sánchez (1996). 4
6. Esquema de control basado en aprendizaje por refuerzo de acuerdo a Yu & Perrusquía (2021). Permite unir las características óptimas y adaptativas en un esquema simple que funcione en lazo cerrado y en línea con el sistema. 5
7. Sesgo disciplinario del aprendizaje por refuerzo desde la teoría de control y el aprendizaje de máquina explicado por Recht (2018). 6
8. Flujo de la interacción en DP y RL de acuerdo a Busoniu et al. (2017). 13
9. Recompensas descontadas durante la transición de estados para un horizonte de tiempo definido por N. 16
10. Ilustración conceptual del principio de optimalidad de Bellman mostrada en Bertsekas (2019). La cola $\{u_k^*, \ldots, u_{N-1}^*\}$ de una secuencia óptima $\{u_0^*, \ldots, u_{N-1}^*\}$, es óptima para la subtrayectoria que inicia en el estado x_k^* de la trayectoria óptima $\{x_1^*, \ldots, x_N^*\}$. . 23
11. El aprendizaje por diferencias temporales describe como la ecuación de Bellman captura el funcionamiento de la acción, la observación, la evaluación y el mejoramiento, ilustración tomada de Lewis et al. (2012a). 31
12. El método de aprendizaje por TD usando iteración de política puede ser representado en el esquema de actor-crítico de acuerdo Vrabie et al. (2013). Las estructuras de los aproximadores son representadas por los rectángulos verdes. 33
13. La representación del algoritmo Q-learning para el LQR, en el esquema de actor-crítico solo requiere un aproximador debido a la naturaleza de las Q-funciones. Considérese la representación de un aproximador como la unión de las tareas desempeñadas por el actor y el crítico, además, $u' = h_j(x_{k+1})$. 33
14. La forma funcional de la VFA para las funciones V, consiste en un mapeo de extracción que va del espacio de estado al espacio de funciones, con el objetivo de generar la entrada a un mapeo lineal que sirva para generar la aproximación descrita por $W^T\Phi(x_k)$. 34
15. La forma funcional de la VFA para las Q-funciones, consiste en un mapeo de extracción que va del espacio de estado al espacio de funciones, y una política como mapeo sobre el estado para definir la acción de control hacia el espacio de funciones, con el objetivo de generar la entrada a un mapeo lineal que sirva para generar la aproximación descrita por $W^T\Phi(x_k, u_k)$. 35

Figura Página

16. Ilustración conceptual de la aproximación usando la iteración de la Q-función. Idealmente, el algoritmo de RL usado converge a un punto fijo W^* a través de la composición de mapeos descrita por (63). Obsérvese que el resultado del mapeo $\mathcal{H}$ es proyectado de regreso al espacio de parámetros usando el mapeo $\mathcal{P}$ indicado con las flechas rojas. . . . 37

17. Ilustración conceptual del entrenamiento episódico usando funciones V. Se consideran las mismas condiciones iniciales x_0 para cada episodio, el horizonte de tiempo definido por N para la trayectoria. Este esquema no requiere tener la misma longitud de trayectoria para todos los episodios. La línea cortada roja indica el proceso realizado por la RNA. . . 46

18. Ilustración conceptual del entrenamiento continuo usando funciones V. Este esquema considera realizar la evaluación de la política (o actualización de valor) y el mejoramiento de la política en un solo episodio, donde las iteraciones indicadas por j pueden tener diferentes cantidades de datos muestreados. Este esquema de entrenamiento puede ser implementado mientras el sistema está en operación. Considérese que la red neuronal del actor requiere todo el largo de la trayectoria usada por el crítico conforme se realizan los mejoramientos. 46

19. Sistema mecánico masa-resorte-amortiguador. 48

20. Señales utilizadas como PE. 55

21. Señales multisenoidales con decaimiento exponencial probadas en la metodología para entrenamiento continuo. 56

22. Comportamiento natural del sistema masa-resorte-amortiguador usando un vector de ganancias de entrada $K = [1, -0.3]$ para unas condiciones iniciales de $x_1 = 0.5$ y $x_2 = -0.4$. 56

23. Estado x_1 del sistema afectado por la entrada de control con PE. De una señal con $\omega_i = 0.005\, rad/seg$ y $\omega_f = 345,000\, rad/seg$ con una baja amplitud controlada por la constante de amplificación $c_r = 0.09$ (gráfica superior) y señal multisenoidal con decaimiento exponencial $0.1e^{-0.022k}$ (gráfica inferior). Se usa un vector de ganancias de entrada de control $K = [1, -0.5]$, para una condición inicial de $x_1 = 0.5$. 57

24. Trayectorias de los estados x_1 y x_2, del sistema estable en entrenamiento episódico, con un $n_{ep} = 3$ usando el Algoritmo 3. 59

25. Ilustración del funcionamiento a bloques del crítico para un episodio usando el Algoritmo 3. 59

26. Ilustración del funcionamiento a bloques del actor para un episodio usando el Algoritmo 3. 60

27. Ilustración conceptual del esquema para entrenamiento continuo usando el algoritmo Q-learning. El color magenta indica el bloque de muestras medidas de la trayectoria de estado y el control usados por LS para sintonizar los parámetros de W^{j+1}. El color azul indica la operación de RLS o GD usando las muestras medidas para la sintonización del crítico en línea. 67

28. Descripción a bloques del Algoritmo 5, se toma como base la metodología descrita en Vrabie et al. (2013). 68

29. Convergencia de los parámetros del vector W hacia valores óptimos usando el Algoritmo 3. 72

30. Convergencia de los parámetros del vector K hacia valores óptimos usando el Algoritmo 3. 72

Figura Página

31. Norma euclidiana de la diferencia medida entre P_{j+1} y P_j (gráfica superior) y de la diferencia medida entre K_{j+1} y K_j (gráfica inferior) usando el Algoritmo 3. 73

32. Convergencia de los parámetros del vector W hacia valores óptimos usando el Algoritmo 3 - sistema inestable. 74

33. Convergencia de los parámetros del vector K hacia valores óptimos usando el Algoritmo 3 - sistema inestable. 74

34. Norma euclidiana de la diferencia medida entre P_{j+1} y P_j (gráfica superior) y de la diferencia medida entre K_{j+1} y K_j (gráfica inferior) usando el Algoritmo 3 - sistema inestable. 75

35. Convergencia de los parámetros del vector W hacia valores óptimos usando el Algoritmo 4. 76

36. Convergencia de los parámetros del vector K hacia valores óptimos usando el Algoritmo 4. 76

37. Norma euclidiana de la diferencia medida entre P_{j+1} y P_j (gráfica superior) y la diferencia medida entre K_{j+1} y K_j (gráfica inferior) usando el Algoritmo 4. 77

38. Convergencia de los parámetros del vector W hacia valores óptimos usando el Algoritmo 4 - sistema inestable. 78

39. Convergencia de los parámetros del vector K hacia valores óptimos usando el Algoritmo 4 - sistema inestable. 78

40. Norma euclidiana de la diferencia medida entre P_{j+1} y P_j (gráfica superior) y de la diferencia medida entre K_{j+1} y K_j (gráfica inferior) usando el Algoritmo 4 - sistema inestable. 79

41. Convergencia de los parámetros del vector W hacia valores óptimos usando el Algoritmo 6. Sintonización del crítico usando LS+RLS con los parámetros iniciales: $n_{ls} = 3$, $\lambda = 0.99$ y $\delta = 10^{-2}$. 80

42. Convergencia de los parámetros del vector K hacia valores óptimos usando el Algoritmo 6. Sintonización del actor usando GD con $\beta = 0.3939$. 81

43. Comportamiento de los estados y la salida durante el entrenamiento continuo con condiciones iniciales $x_0 = [0.5 \quad -4]^T$. Los parámetros generales para el entrenamiento continuo usando LS+RLS y GD en el Algoritmo 6 son: $N = 400$, $\gamma = 0.999$, $Q = \mathbb{I}_{2\times 2}$ y $R = 2$. Tiempo de ejecución ≈ 3.07284 segundos. 81

44. Convergencia de los parámetros del vector W hacia valores óptimos usando el Algoritmo 6. Sintonización del crítico usando LS+GD con los parámetros iniciales: $n_{ls} = 4$, $\alpha = 2.01988$ y $\delta = 10^{-2}$. 82

45. Convergencia de los parámetros del vector K hacia valores óptimos usando el Algoritmo 6 (LS+GD). Sintonización del actor usando GD con $\beta = 0.007339$. 82

46. Comportamiento de los estados y la salida durante el entrenamiento continuo con condiciones iniciales $x_0 = [0.5 \quad -4]^T$. Los parámetros generales para el entrenamiento continuo usando LS+GD y GD en el Algoritmo 6 son: $N = 400$, $\gamma = 0.999$, $Q = \mathbb{I}_{2\times 2}$ y $R = 2$. Tiempo de ejecución ≈ 3.05006 segundos. 83

Figura Página

47. Convergencia de los parámetros del vector W hacia valores óptimos usando el Algoritmo 5 con LS+GD. 85

48. Convergencia de los parámetros del vector K hacia valores óptimos usando el Algoritmo 5. 86

49. Comportamiento de los estados y la salida durante el entrenamiento continuo usando Algoritmo 5 con LS+GD. Tiempo de ejecución ≈ 3.83227 segundos. 86

50. Convergencia de los parámetros del vector W hacia valores óptimos usando el Algoritmo 5 con LS+RLS. La matriz $\Gamma(n_{ls})$ es escalada por 1000. 87

51. Convergencia de los parámetros del vector K hacia valores óptimos usando el Algoritmo 5. 88

52. Comportamiento de los estados y la salida durante el entrenamiento continuo usando Algoritmo 5 con LS+RLS. Tiempo de ejecución ≈ 1.1352 segundos. 88

53. Convergencia de los parámetros del vector W hacia valores óptimos usando el Algoritmo 5 con GD para la sintonización del crítico. Con los parámetros $N = 4000$, $\gamma = 0.999$, $\alpha = 120.02$, $\delta = 10^{-2}$ y $c_r = 0.202$. Tiempo de ejecución ≈ 0.90386 segundos. 89

54. Convergencia de los parámetros del vector W hacia valores óptimos usando el Algoritmo 5 con RLS para la sintonización del crítico. Con los parámetros $N = 4000$, $\gamma = 0.999$, $\lambda = 0.99$, $\delta = 10^{-2}$ y $c_r = 0.202$. La matriz $\Gamma(k) = \mathbb{I}_{6\times 6} \times 1000$. Tiempo de ejecución ≈ 1.2035 segundos. 90

55. Convergencia del parámetro w_1 con dos variaciones usando el Algoritmo 5, con LS+GD para la sintonización del crítico. Con los parámetros $N = 11000$, $\gamma = 0.999$, $n_{ls} = 21$, $\alpha = 70.02$, $\delta = 10^{-2}$ y $c_r = 0.202$. Tiempo de ejecución ≈ 11.2009 segundos. 92

56. Convergencia del parámetro w_1 con dos variaciones usando el Algoritmo 5, con LS+RLS para la sintonización del crítico. Con los parámetros $N = 11000$, $\gamma = 0.999$, $n_{ls} = 7$, $\lambda = 0.99$, $\delta = 10^{-2}$ y $\Gamma(n_{ls})$ es escalada por 1000. Tiempo de ejecución ≈ 4.4748 segundos. 93

Lista de tablas

Tabla Página

1. Relación entre los elementos de un sistema de control y la DP/RL. 14
2. Parámetros del sistema masa-resorte-amortiguador considerado para las pruebas de los algoritmos. 49
3. Valores de los parámetros requeridos por el Algoritmo 3, para el actor $\beta = 0.0102$. 71
4. Valores óptimos y estimados de la matriz P y el vector K - sistema inestable. . . 79
5. Comparación de valores óptimos y estimados del vector W y el vector K para el sistema estable. 83
6. Valores de los parámetros requeridos por el Algoritmo 5 usando LS+GD para la sintonización del crítico. 85
7. Comparación de valores óptimos y estimados del vector W y el vector K para el sistema estable. 89
8. Parámetros del sistema masa-resorte-amortiguador nominal, variación intermedia y peor variación. 91
9. Comparación de valores óptimos y estimados del vector W y el vector K para la peor variación. 93

Capítulo 1. Introducción

El aprendizaje por refuerzo (RL, por sus siglas en inglés), es principalmente considerado un área del aprendizaje de máquina ampliamente estudiado en la comunidad de ciencias de la computación y usado en diferentes disciplinas como método para encontrar acciones en ambientes inciertos que resuelvan un problema de optimización (Lewis et al., 2012a). Siendo más específico, el aprendizaje por refuerzo es un conjunto de algoritmos aplicables a sistemas que evolucionan en el tiempo, inspirados en la teoría de la psicología del comportamiento animal, donde se da una interpretación de cómo un ser vivo aprende y toma decisiones (Sutton & Barto, 2018). Dos categorías de experimentos son conocidas: los de condicionamiento clásico y los de condicionamiento instrumental. Dentro de la primera categoría, uno muy conocido fue realizado por Pavlov en 1927 para entrenar a sus perros y la respuesta que daban a estímulos provocados en situaciones similares después de un condicionamiento. En sus monografías se menciona el concepto de reforzamiento, un concepto introducido por un experimento de condicionamiento instrumental en el aprendizaje animal realizado años atrás, en 1911 por Edward Thorndike, descrito como "Ley del efecto", debido a que describe eventos repetidos como resultado de la tendencia a seleccionar ciertas acciones (Sutton & Barto, 2018). Pavlov retomó la idea y describió el concepto de reforzamiento como el fortalecimiento o debilitamiento de un patrón de comportamiento dependiendo de los estímulos recibidos como recompensa o castigo.

Años después, en 1957, en el área de las matemáticas aplicadas, Richard Bellman propone el concepto de función de valor, buscando resolver el problema de control óptimo en tiempo discreto, creando las bases de la programación dinámica (DP, por sus siglas en inglés). En esa misma década se introduce una tercera idea que serviría para unir las ideas inspiradas en la psicología del comportamiento animal con los problemas de control óptimo y programación dinámica: el aprendizaje por diferencias temporales. Este concepto es propuesto en diferentes trabajos (Sutton & Barto, 2018) como un procedimiento de aprendizaje incremental especializado para la predicción, es decir, usando experiencia pasada en un sistema de información incompleta se predice su comportamiento futuro (Sutton, 1988). Este contexto permitió un amplio desarrollo teórico del aprendizaje por refuerzo para sistemas de control moderno, que toman como base el control óptimo en tiempo discreto, donde resolver el problema de optimización consiste en maximizar o minimizar una función objetivo.

El aprendizaje por refuerzo tiene la dicotomía de ser dependiente o libre de modelo, en los últimos años los trabajos, específicamente en el área de aprendizaje por refuerzo libre de modelo, han incrementado su popularidad en robótica, donde se busca brindar mayor autonomía a sistemas mecánicos y condiciones de optimalidad para la eficiencia en las tareas realizadas.

1.1. Definiciones básicas

Desde el contexto de la teoría de control, se definen los siguientes conceptos:

Sistema: conjunto de elementos relacionados que tienen un objetivo y transforman señales llamadas de entrada en otras señales llamadas de salida, como se ilustra en la Figura 1.

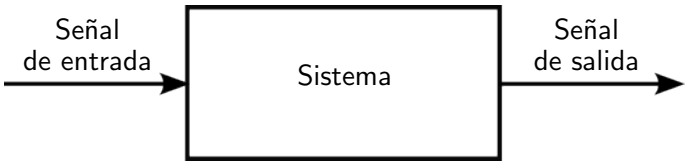

Figura 1. Flujo de señales en un sistema.

Modelo: conjunto de ecuaciones matemáticas que representan la dinámica del sistema.

Planta: objeto físico a controlar.

Sistema de control: los sistemas de control tienen como objetivo que las señales de salida de una planta sean capaces de ser gobernadas por las directrices marcadas por las señales de entrada, con independencia de las perturbaciones (Miranda, 2012). En esta tesis nos referimos a sistemas de control clásico como cualquier tipo de controlador diseñado desde la teoría de control lineal o no lineal.

Métodos en línea: son aquellos donde la mayoría de cálculos son desempeñados justo después de conocido el estado actual (Bertsekas, 2019).

La Figura 2 ilustra el esquema de un sistema de control en lazo abierto, donde x_{d_k} es el estado deseado en el instante k como señal de referencia de entrada al controlador, y x_k es el estado en el instante k como señal de salida de la planta. La Figura 3 ilustra el esquema de un sistema de control en lazo cerrado con realimentación del estado x_k.

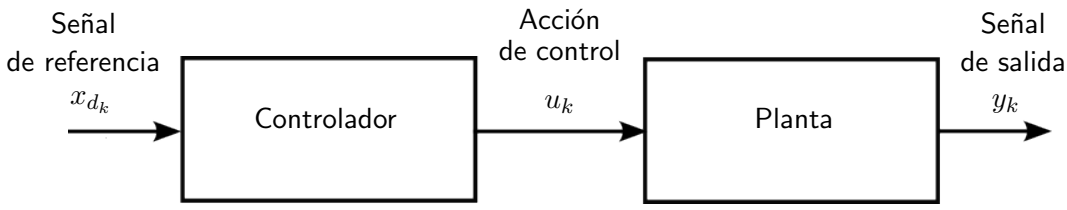

Figura 2. Esquema de un sistema de control clásico en lazo abierto. Diagrama tomado y modificado de Miranda (2012).

Los controladores clásicos para sistemas mecánicos se basan en el modelo que describe la dinámica del sistema, para el diseño de un controlador que cumpla una tarea que implica una señal de referencia o de seguimiento de trayectoria. El control óptimo busca diseñar controladores lineales a partir de la solución de la ecuación algebraica de Riccati (Lewis et al., 2012b), con el objetivo de resolver un problema de optimización. Sin embargo, estas metodologías dependen del conocimiento exacto del modelo en todo momento para el diseño del control. Aunque existen aplicaciones de control óptimo en línea con el sistema, generalmente su implementación es fuera de línea.

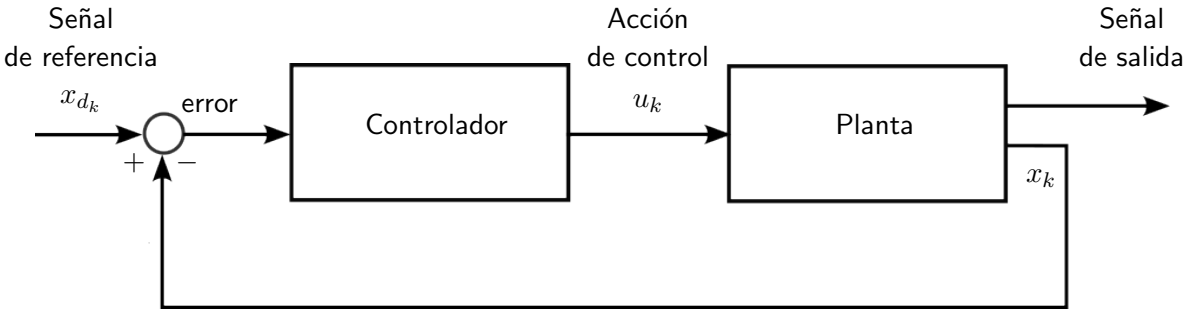

Figura 3. Esquema de un sistema de control clásico en lazo cerrado con realimentación de estado. Diagrama tomado y modificado de Miranda (2012).

Por otro lado, los controladores adaptativos son implementados en línea y pueden ajustar su comportamiento como respuesta a variaciones que puedan suceder en la planta o en la dinámica completa del sistema, a través de un segundo lazo de control que mide un índice de desempeño, dicho índice es comparado con el desempeño deseado, esta comparación actúa como un error sobre un mecanismo de adaptación, el cual actúa sobre el controlador o directamente sobre la señal de control como se ilustra en la Figura 4.

Según sean diseñados los bloques descritos en la Figura 4, podemos dividir el diseño de controles adaptativos en dos grupos principales de acuerdo a Rubio & Sánchez (1996).

- Por modelo de referencia, (MRAC, por sus siglas en inglés): intentan alcanzar un comportamiento, dando una señal de entrada definida y un modelo de referencia.
- Controladores (reguladores) autoajustables (STR, sus siglas en inglés): intentan alcanzar un control óptimo, sujeto al tipo de controlador y la obtención de información que viene de la planta. Dicha información es usada en un bloque de estimación de parámetros.

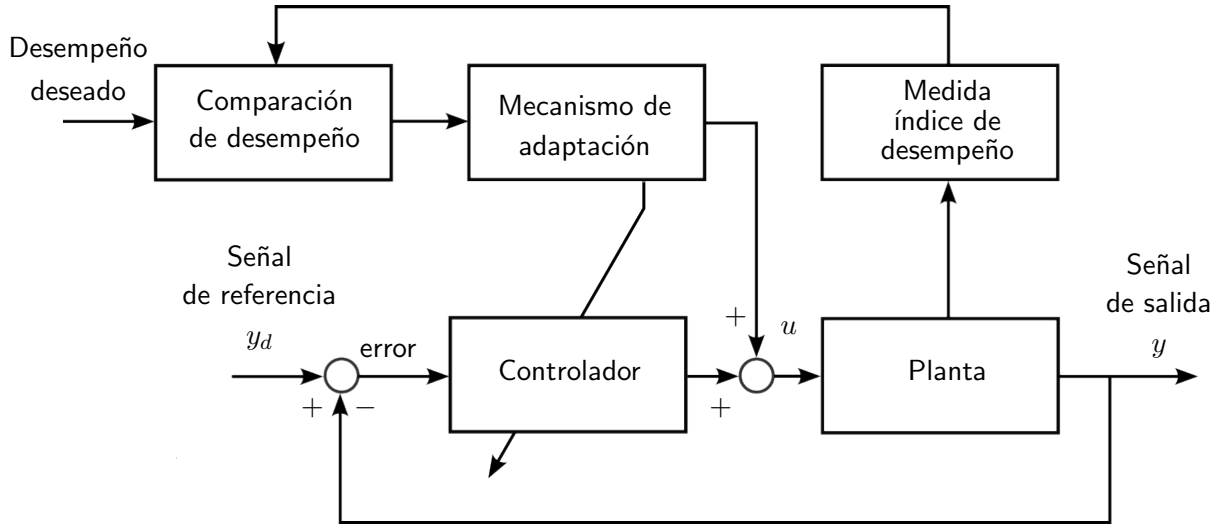

Figura 4. Configuración básica de control adaptativo de acuerdo a Rubio & Sánchez (1996).

El segundo tipo, ilustrado en la Figura 5, tiene menos limitaciones y mayor modularidad en el diseño (Rubio & Sánchez, 1996), pero la planta debe ser identificada en todo momento. El control adaptativo generalmente no es óptimo, aunque considera un criterio de optimización que está en relación a la calidad de la identificación de la planta, pero este criterio no es dado a priori por el usuario. Algunos trabajos consideran un tipo especial de optimalidad (Miroslav & Zhong-Hua, 1997), pero esto aumenta la dificultad en el diseño y requiere la intervención de un experto.

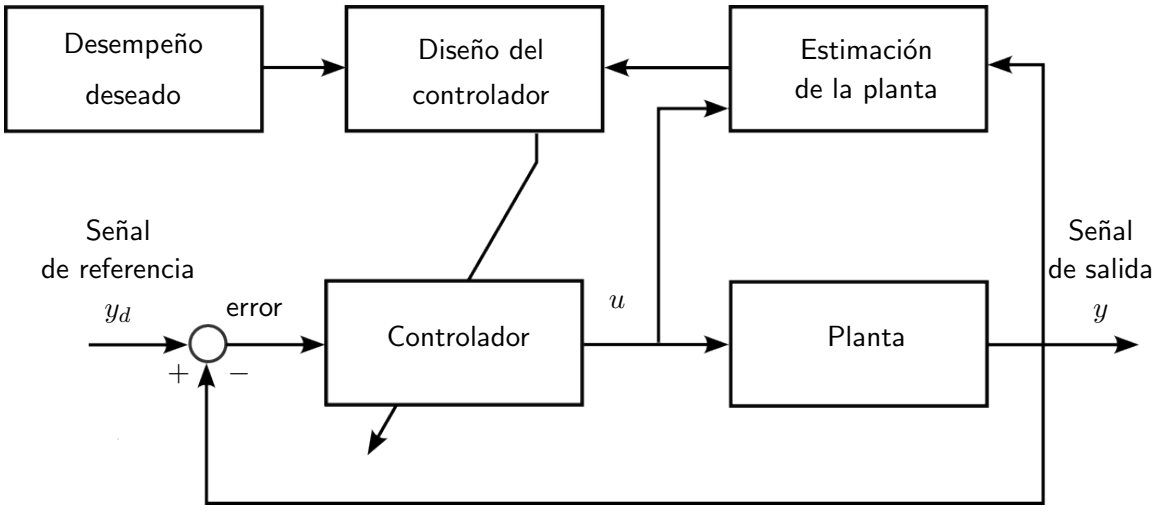

Figura 5. Esquema de control adaptativo con controlador autoajustable (STR) de acuerdo a Rubio & Sánchez (1996).

El aprendizaje por refuerzo es una alternativa para el diseño de controladores que cumplan un criterio de

optimización propuesto por el usuario y que presenten características adaptativas a variaciones, debido a que pueden ser implementados en línea usando solo los datos medidos sobre las trayectorias del sistema, es decir, no requieren del conocimiento del modelo para el diseño del controlador. La Figura 6 muestra el esquema básico de un sistema de control en lazo cerrado basado en aprendizaje por refuerzo.

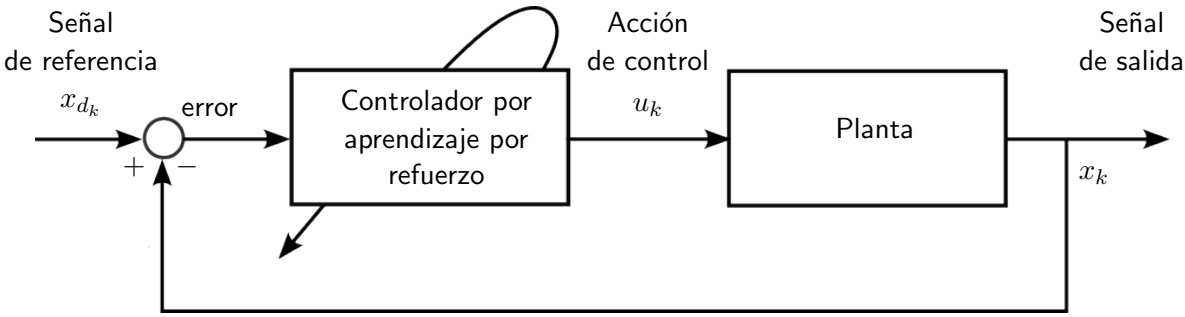

Figura 6. Esquema de control basado en aprendizaje por refuerzo de acuerdo a Yu & Perrusquía (2021). Permite unir las características óptimas y adaptativas en un esquema simple que funcione en lazo cerrado y en línea con el sistema.

Dos tipos de problemas son conocidos para un controlador óptimo:

Regulación óptima: diseñar un controlador que asegure que los estados o la salida del sistema convergen a cero o muy cercano a cero, minimizando un índice de desempeño predefinido.

Seguimiento óptimo de trayectoria: diseñar un controlador que haga que la salida del sistema siga una trayectoria de referencia, minimizando un índice de desempeño predefinido.

1.2. Antecedentes y motivación

Para motivar la relevancia de la presente tesis, es necesario conocer de manera concisa algunos trabajos realizados usando aprendizaje por refuerzo, partiendo desde el sesgo disciplinario existente en los campos de estudio de la teoría de control y el aprendizaje de máquina, ilustrado en la Figura 7.

El área de ciencias de la computación ha estudiado ampliamente el aprendizaje por refuerzo como un paradigma del aprendizaje de máquina localizado entre el aprendizaje supervisado y el aprendizaje no supervisado. El aprendizaje supervisado consiste en aprender de un conjunto de entrenamiento de ejemplos etiquetados proporcionados por un supervisor externo informado (Sutton & Barto, 2018). Este es un tipo de aprendizaje importante y el más estudiado en el campo del aprendizaje de máquina, pero no

es adecuado para el aprendizaje desde la interacción con un ambiente desconocido que puede cambiar. Por otro lado, el aprendizaje no supervisado trata de encontrar estructuras ocultas en colecciones de datos no etiquetados (Sutton & Barto, 2018). Tomando en cuenta lo anterior, el aprendizaje por refuerzo sí recibe un tipo de información que proviene de la interacción entre los elementos que lo conforman, lo cual le permite mejorar su desempeño conforme aumenta su interacción con un ambiente que no necesariamente debe ser conocido. Esta información ayuda al cumplimiento de su objetivo, el cual consiste en maximizar recompensas o minimizar castigos.

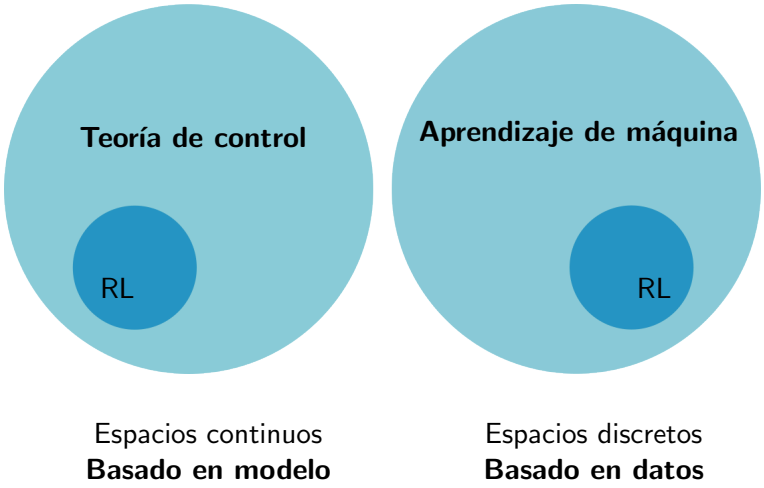

Figura 7. Sesgo disciplinario del aprendizaje por refuerzo desde la teoría de control y el aprendizaje de máquina explicado por Recht (2018).

El aprendizaje por refuerzo desarrollado como subconjunto del aprendizaje de máquina, trabaja principalmente con espacios de estado y acción discretos, para problemas de decisión secuencial que usan tablas o árboles de búsqueda, en combinación con un modelo conocido y una solución global de acuerdo a Moerland et al. (2022). Bajo el enfoque anterior algunos trabajos realizados permiten interactuar con los conceptos de aprendizaje por refuerzo y probar algoritmos en diferentes ambientes simulados, particularmente el trabajo realizado por OpenAI con su biblioteca Gym (Gym, 2022), y el módulo Pygame (Pygame, 2023) fueron de utilidad en el primer contacto con el aprendizaje por refuerzo para esta tesis. Sin embargo, este enfoque también aborda problemas en tareas de control con espacios de estado y acción continuos, como en aplicaciones de robótica (Kober et al., 2013), donde la búsqueda de acciones óptimas y la exploración de estados son un reto complejo debido a la alta dimensión de los datos (provenientes de diferentes sensores) que presentan los sistemas de interés (e.g., visión por computadora, señales de radar, número de grados de libertad), permitiendo el desarrollo del aprendizaje por refuerzo profundo (Hao et al., 2022) como una alternativa de aproximación para espacios discretos

y continuos de alta dimensión. Otra característica importante de este enfoque es que basa la toma de decisiones en los datos (data-driven) observados durante la interacción. Trabajos para aplicaciones de control que utilizan este enfoque pueden verse en Cheng et al. (2013), donde el desempeño del algoritmo conocido como actor crítico es mejorado usando aprendizaje por diferencias temporales, implementado en línea con el sistema a través de entrenamientos denominados episódicos, para el diseño de un control en problemas con espacios continuos, tomando como ejemplos al péndulo invertido y al carro en la montaña. Puriel Gil et al. (2019) proponen la implementación de un controlador con una compensación proporcional-derivativa (PD), agregando una compensación por aprendizaje por refuerzo para el problema del péndulo doble sobre un carro, obteniendo mejores resultados en comparación con los mismos métodos implementados por separado.

En un trabajo reciente, Siraskar (2021) realiza un estudio de los algoritmos de aprendizaje por refuerzo aplicados a una válvula no lineal, usando como herramienta el toolbox de reinforcement learning introducido por Matlab 2019, integrando Simulink para la simulación del comportamiento no lineal de la válvula, obteniendo buenos resultados en el seguimiento de trayectoria, comparado con un controlador basado en una estrategia proporcional-integral-derivativa (PID).

Antes de continuar con un enfoque desde la teoría de control, es necesario dar la definición de aprendizaje por refuerzo elegida y usada para este proyecto de tesis, la cual fue propuesta en Recht (2019):

"El aprendizaje por refuerzo es el estudio de cómo usar los datos pasados para mejorar la manipulación futura de un sistema dinámico".

Tomando en consideración la definición anterior, podemos decir que el aprendizaje por refuerzo también ha sido estudiado en la teoría de control, no solo porque parte de su base es el análisis no lineal para tareas de control óptimo, sino también, porque la idea de aprendizaje también está relacionada a tareas de control predictivo basado en modelo (MPC, por sus siglas en inglés) (Grzedzinski & Trodden, 2018) y tareas de control adaptativo (Lewis et al., 2012a), con espacios de estado y acción continuos para ambos casos. Generalmente, ambas estrategias de control toman las decisiones basadas en el modelo o una aproximación del mismo. Sin embargo, un esfuerzo desarrollado desde el área de control óptimo permite unir en un solo esquema, técnicas de control clásico y algoritmos que usan el concepto de reforzamiento, para evitar el uso del modelo en el diseño del control y resolver problemas en espacios continuos de una manera sencilla y elegante. Partiendo del desarrollo propuesto por Bradtke et al. (1994), donde se define el método de estimación directa de la Q-función a través de la ecuación de Bellman, se

desarrollan trabajos para la regulación y el seguimiento de trayectoria en control óptimo. Particularmente, Neto et al. (2013) usan la ecuación de Bellman descrita en Bradtke et al. (1994) con el diseño de una recompensa cuadrática, para estimar los parámetros de una matriz P como solución a la ecuación algebraica de Riccati en tiempo discreto. La estimación de la matriz P se realiza con programación dinámica heurística, como se le conoce a la iteración de valor en los métodos críticos adaptativos de acuerdo a Landelius & Knutsson (1996), y mínimos cuadrados recursivos. Sin embargo, aún se requiere el conocimiento del modelo del sistema en el mejoramiento de la acción de control. Al-Tamimi y Lewis, enfocan sus esfuerzos en resolver los problemas con espacios continuos partiendo desde un enfoque en tiempo discreto ((Al-Tamimi et al., 2008) y (Al-Tamimi, 2007)) cuando se tiene conocimiento parcial de la dinámica del sistema, es decir, solo se conoce una parte de la ecuación que describe la dinámica del sistema. Al-Tamimi y Lewis introducen redes neuronales como aproximadores para la sintonización de parámetros que describan explícitamente la función de valor y la acción de control. Lo anterior, abordando el problema del regulador cuadrático lineal y usando polinomios como funciones de activación en las redes neuronales, para sistemas lineales y no lineales. Con el mismo enfoque, se aborda el problema del seguidor cuadrático lineal (LQT, por sus siglas en inglés) en Kiumarsi-Khomartash et al. (2013) para el seguimiento óptimo de referencia, utilizando un comando generador para la definición de la referencia cuando se tiene conocimiento parcial de la dinámica del sistema. Posteriormente, Kiumarsi et al. (2014) proponen el uso de la Q-función para abordar el problema del LQT, basado en la estimación directa de la Q-función, como alternativa de solución a problemas en donde se tenga desconocimiento total del modelo que describe la dinámica del sistema para el diseño de las políticas de control, usando únicamente los datos medidos a lo largo de las trayectorias.

En Kiumarsi et al. (2018) se da una revisión de los algoritmos de aprendizaje por refuerzo para regulación y seguimiento de trayectoria óptima en tiempo discreto y continuo, tomando como base el algoritmo Q-learning y el aprendizaje por refuerzo integral para los algoritmos en tiempo discreto. Los trabajos mencionados anteriormente enfocan sus esfuerzos en la estructura cuadrática que se da a la función de valor, con mayor cantidad de trabajos utilizando el algoritmo Q-learning, incluyendo pruebas de convergencia. Algunos enfoques basados en control descentralizado, como el caso del regulador cuadrático lineal distribuido, aún se encuentran sin pruebas de convergencia, sin embargo, trabajos como Görges (2019) revisan y comparan algoritmos propuestos por Vrabie et al. (2013) contra un enfoque distribuido, usando simulaciones numéricas como estudio, obteniendo resultados cercanos a los óptimos.

La idea de usar aprendizaje por refuerzo en aplicaciones de control comienza a ser altamente deseada debido a la cantidad de aplicaciones reales sobre las cuales no conocemos el modelo que describe la

dinámica del sistema, el reto sigue siendo diseñar controles óptimos y adaptativos que brinden mayor autonomía en diferentes aplicaciones. Wang & Hong (2020) dan una revisión sobre el diseño de controladores y hacen un breve análisis de los retos que implica el desarrollo de controladores basados en aprendizaje por refuerzo, donde la gran mayoría de los controladores diseñados son enfocados al sector de la construcción y la energía (e.g., HVAC, iluminación, almacenamiento de energía, calentamiento de agua doméstica).

1.3. Definición del problema

Con base en lo anterior, este proyecto de tesis realiza un trabajo exploratorio en el uso de algoritmos de aprendizaje por refuerzo para el diseño de un controlador lineal óptimo. El controlador es implementado en tiempo discreto para tareas que implican espacios de trabajo continuos. Se usa un aproximador para la parametrización de la función de valor empleada para resolver un problema de optimización en el diseño de un controlador en un sistema determinista[1], lineal e invariante en el tiempo, el cual corresponde a un sistema mecánico de un grado de libertad y de segundo orden. Los algoritmos de aprendizaje por refuerzo elegidos deben considerar escenarios donde se tiene conocimiento parcial o desconocimiento total del modelo que describe la dinámica del sistema, con el objetivo de concluir en una metodología que permita tener poca dependencia de la intervención de un experto en el diseño de controladores automáticos. La parte central del proyecto se lleva acabo en la estimación de parámetros a través de aproximadores, buscando reducir el número de iteraciones necesarias para la convergencia a valores óptimos. Se toma como referencia principal la literatura dirigida por Frank L. Lewis y la metodología descrita en dos de sus trabajos, *"Control óptimo"*, (Lewis et al., 2012b) y *"Control óptimo adaptativo y juegos diferenciales por aprendizaje por refuerzo"*, (Vrabie et al., 2013).

1.4. Objetivos

1.4.1. Objetivo general

Diseñar, implementar y analizar un controlador libre de modelo a través de algoritmos de aprendizaje por refuerzo para sistemas lineales e invariantes en el tiempo.

[1] Caso determinista: es aquel en el que el azar no está involucrado.

1.4.2. Objetivos específicos

- Determinar los algoritmos y la estructura a utilizar para el diseño del controlador.
- Implementar los algoritmos para el diseño de controladores en tiempo real.
- Definir los aproximadores paramétricos para la definición de las funciones de valor.
- Implementar los algoritmos en las plantas seleccionadas.
- Comparar el desempeño de los controladores implementados usando diferentes estimadores.

1.5. Metodología de trabajo

Este proyecto de tesis se desarrolla en dos posibles casos, cuando se tiene un conocimiento parcial del modelo que describe la dinámica del sistema y cuando no se tiene conocimiento de dicho modelo.

Dinámica parcialmente conocida: los problemas con dinámica parcialmente conocida permiten el uso de funciones de valor V. Sin embargo, para usar las funciones de valor en espacios de trabajo continuos, se deben aproximar las funciones de valor (Busoniu et al., 2017). Por ello se usan aproximadores paramétricos, cuya estructura consiste de un vector de pesos y de un vector de regresión.

Se plantea una solución a través de un experimento llamado entrenamiento episódico, donde la solución es posible usando dos aproximadores paramétricos. Es necesario el uso de estimadores de parámetros para la actualización de los aproximadores. El primer estimador se centra en la actualización de la función de valor V, es decir, es el encargado de sintonizar el vector de pesos del primer aproximador. Para ello se selecciona mínimos cuadrados recursivos, debido a que es un estimador de parámetros que no presenta pérdida de rango de la matriz de covarianza en entrenamientos prolongados. Además, supone ser una mejor alternativa para tareas que se desarrollan en tiempo real (Lewis et al., 2012a), donde se prefiere que la información sea actualiza de manera constante y no por lotes. El segundo estimador de parámetros usa descenso del gradiente con paso fijo, y se encarga de mejorar la política de control mediante la actualización del vector de pesos del segundo aproximador.

La ecuación de Bellman es necesaria para la solución óptima, donde dos algoritmos son usados para la solución de la ecuación: el algoritmo de iteración de valor y el de iteración de política. Para reducir

el número de iteraciones necesarias para la convergencia y reducir el número de parámetros iniciales a sintonizar de manera arbitraria, se adhiere un bloque de mínimos cuadrados por lotes al inicio del primer estimador.

Para medir la calidad de los resultados, se realiza la simulación del comportamiento del sistema al aplicar directamente el vector de ganancias K estimado por cada algoritmo, y se hace la medición del error cuadrático medio respecto a la simulación del comportamiento al aplicar el vector de ganancias K óptimo.

Dinámica totalmente desconocida: cuando no se tiene conocimiento del modelo, se puede hacer uso de las Q-funciones. Esta metodología se basa en la estimación directa de la Q-función, conocida como el algoritmo Q-learning. Se plantea la solución a través de un experimento llamado entrenamiento continuo.

Este algoritmo solo requiere de un aproximador paramétrico, por lo que es necesario solo un estimador de parámetros encargado de la sintonización de pesos del aproximador. Se usan dos esquemas como estimadores de parámetros: mínimos cuadrados por lotes más mínimos cuadrados recursivos, y mínimos cuadrados con descenso del gradiente. Para ambos esquemas se hace uso del algoritmo de iteración de política para la solución de la ecuación de Bellman. Adicionalmente, se realizan pruebas de adaptabilidad tomando en cuenta variaciones que puedan ocurrir en los parámetros de la planta durante el entrenamiento.

Para medir la calidad de los resultados, se realiza la simulación del comportamiento del sistema al aplicar directamente el vector de ganancias K estimado por el algoritmo, al usar cada uno de los estimadores. Se hace la medición del error cuadrático medio respecto a la simulación del comportamiento al aplicar el vector de ganancias K óptimo. La adaptabilidad se analiza en función de dos variaciones, tomando en cuenta la medición del error cuadrático medio y el número de iteraciones necesarias para la convergencia a valores óptimos.

1.6. Estructura de la tesis

- **Capítulo 1. Introducción:** muestra los aspectos generales de la tesis, antecedentes y trabajos relacionados que justifican el objetivo general del trabajo. Posteriormente se muestran los objetivos específicos necesarios para cubrir el objetivo general.
- **Capítulo 2. Marco teórico:** muestra los conceptos propios de RL y DP, así como la relación

analítica con el caso más representativo de control óptimo; el regulador cuadrático lineal, que será la base para resolver el problema de optimización para el sistema de interés. Además, partiendo de la dicotomía de RL, se explican los algoritmos seleccionados de DP, dependientes del modelo, así como los seleccionados de RL, libres de modelo.

- **Capítulo 3. Metodología:** se describe la metodología usada para la implementación de los algoritmos seleccionados libres de modelo, aplicados a dos sistemas (estable e inestable) como casos de estudio. Se detalla el uso de las herramientas de aprendizaje de máquina e identificación de sistemas utilizadas.

- **Capítulo 4. Experimentos y resultados:** muestra los resultados obtenidos para ejemplos con dinámica parcialmente conocida y dinámica totalmente desconocida considerando diferentes escenarios para cada ejemplo. Además, se muestra el desempeño adaptativo del controlador diseñado para el sistema de interés de la tesis.

- **Capítulo 5. Discusión:** se presentan los puntos destacados de este trabajo de tesis basados en los resultados obtenidos por el uso de los algoritmos de RL.

- **Capítulo 6. Conclusión:** muestra las conclusiones generales de este trabajo de tesis respecto a los objetivos y se proponen líneas de investigación para trabajo a futuro.

Capítulo 2. Marco teórico

El propósito de este capítulo es presentar los conceptos básicos de programación dinámica y aprendizaje por refuerzo como una alternativa para encontrar soluciones a problemas de control óptimo. Dichos conceptos serán de utilidad para el desarrollo de la metodología (Capítulo 3) en esta tesis.

2.1. Proceso de decisión de Markov

Un proceso de decisión de Markov (MDP, por sus siglas en inglés) de acuerdo con Busoniu et al. (2017) es definido por la cuadrupla (X, U, f, ρ), donde X es el espacio de estados, U es el espacio de acciones que puede realizar el controlador, f es la función que determina la dinámica del sistema y ρ es la función de recompensa, la cual evalúa el desempeño del control aplicado. Para esta tesis se considera un MDP con configuración determinista.

En un MDP existe un controlador (agente o tomador de decisiones), que interactúa con el sistema (ambiente), por medio de tres señales (mediciones); la señal de estado, que brinda información actual sobre el sistema, la señal de acción (control), que permite al controlador modificar el estado y una señal de recompensa escalar, que da una realimentación al controlador sobre su desempeño al aplicar una acción. La recompensa ayuda al controlador a mejorar o cambiar las acciones desempeñadas, al aplicar alguna acción se produce una transición a un nuevo estado y el controlador recibe un nuevo grupo de mediciones (estado y recompensa), y el proceso se repite como se ilustra en la Figura 8.

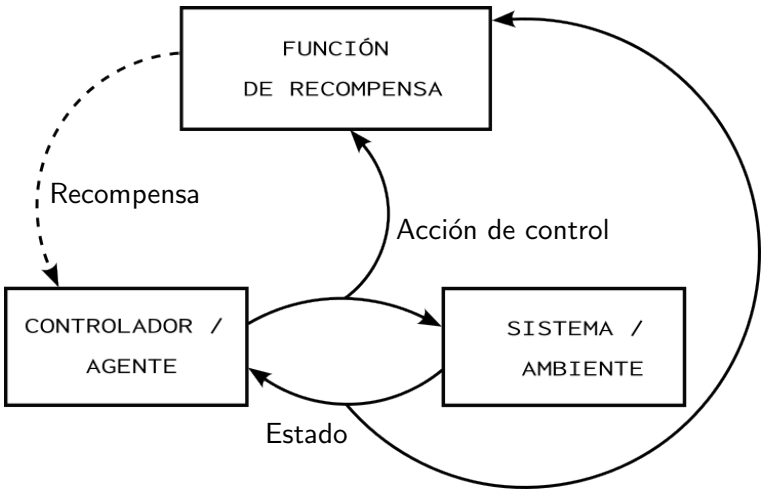

Figura 8. Flujo de la interacción en DP y RL de acuerdo a Busoniu et al. (2017).

La Tabla 1 muestra la relación conceptual de los elementos de un sistema de control en tiempo discreto y la teoría de DP y RL.

Tabla 1. Relación entre los elementos de un sistema de control y la DP/RL.

Señal	Sistema de control	DP/RL
	Controlador	Agente
	Sistema	Ambiente
x_k	Estado	Estado
u_k	Señal de control	Acción
$f(x_k, u_k)$	Dinámica	Función de transición
$\rho(x_k, u_k)$	Índice de desempeño	Recompensa inmediata
$\mathcal{J}_k = V(x_k)$	Función de costo	Función de valor
$h(x_k)$	Control	Política

Las siguientes definiciones fueron tomadas directamente de Busoniu et al. (2017). El cual describe que cuando la acción de control u_k es aplicada en el estado x_k, en el instante de tiempo discreto k, existe una transición de estado a x_{k+1} de acuerdo a la función $f : X \times U \to X$:

$$x_{k+1} = f(x_k, u_k). \tag{1}$$

Al mismo tiempo, el controlador recibe una señal de recompensa escalar r_{k+1}, de acuerdo a la función de recompensa $\rho : X \times U \to \mathbb{R}$:

$$r_{k+1} = \rho(x_k, u_k). \tag{2}$$

La recompensa evalúa el efecto inmediato de la acción u_k, de pasar del estado x_k a x_{k+1}, pero en general no dice nada acerca de los efectos de u_k a largo plazo.

El controlador elige las acciones acorde a su política $h : X \to U$, usando:

$$u_k = h(x_k). \tag{3}$$

Dados f, ρ, el estado actual x_k y el control actual u_k, se puede determinar el estado siguiente x_{k+1} y la recompensa r_{k+1}.

2.1.1. Modelo de optimalidad y retornos

En la teoría de DP y RL, el termino óptimo se asocia al objetivo de encontrar una política que maximice un retorno R desde cualquier estado inicial x_0. La noción de retorno es formalizada en términos de una señal llamada recompensa (Sutton & Barto, 2018). Dicho retorno es resultado de la suma de recompensas recibidas cuando se sigue una política a lo largo de la trayectoria que inicia en x_0. Existen varios tipos de retornos que Kober & Peters (2012) refieren como modelos de optimalidad, su distinción depende de la manera en la que acumulan la recompensa, como lo menciona Busoniu et al. (2017).

Un retorno R que use el concepto de descuento es ideal para las tareas de control continuas en un espacio de tiempo finito. En el enfoque de la tesis, el controlador diseñará acciones de control que minimicen una función de costo descrita por la sumatoria de recompensas, por lo tanto, se supone que la función de recompensa $\rho(x_k, u_k)$ es convexa (e.g., una función cuadrática en x_k y u_k). Dicha sumatoria es conocida como retorno descontado en horizonte infinito, descrito por Busoniu et al. (2017) como:

$$R^h(x_0) = \sum_{i=0}^{\infty} \gamma^i r_{i+1}, \tag{4}$$

donde γ es un factor de descuento. Con la definición de la función de recompensa (2), el retorno dado por (4) y considerando que Lewis et al. (2012a) define una función de costo determinista como la sumatoria de recompensas desde cualquier instante k. Para esta tesis se usa la siguiente descripción de retorno como modelo de optimalidad:

$$R^h(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} r_{i+1} = \sum_{i=k}^{\infty} \gamma^{i-k} \rho(x_i, h(x_i)). \tag{5}$$

Se observa que la configuración está en tiempo infinito, debido a que es ideal para analizar y derivar en métodos conocidos como hacia delante en el tiempo, para encontrar valores y políticas óptimas de acuerdo con Lewis et al. (2012a). El factor de descuento $\gamma \in [0, 1)$ puede ser interpretado como una medida de qué tanto el controlador está considerando sus recompensas; inmediata y futuras, como menciona Busoniu et al. (2017). El factor de descuento determina los valores presentes de las futuras recompensas, i.e., un retorno recibido desde un estado x_k, su recompensa inmediata tendrá un valor condicionado por γ^{k-k} y la recompensa para un estado futuro x_{k+1} tendrá un valor de γ^{k+1-k} veces si fuera recibida de manera inmediata. La Figura 9 ilustra la configuración del retorno descontado y se muestra en que instante aparecen los elementos que componen (5).

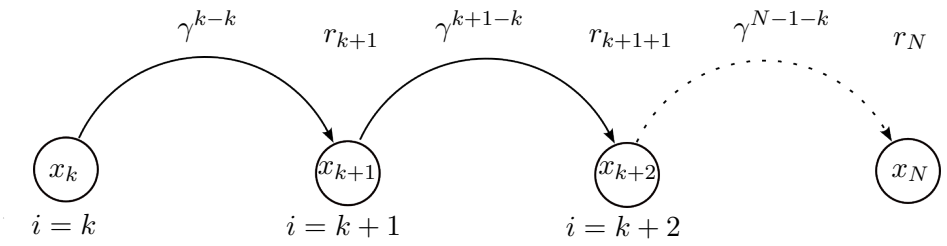

Figura 9. Recompensas descontadas durante la transición de estados para un horizonte de tiempo definido por N.

Se considera que en la práctica las recompensas no se extienden infinitamente, sino que se detienen en un tiempo N conocido como horizonte, esta idea de acotar el retorno de las tareas continuas de control en un tiempo N es conocido como episodio (Sutton & Barto, 2018).

$$R^h(x_k) = r_{k+1} + \gamma r_{k+2} + \gamma^2 r_{k+3} + \gamma^3 r_{k+4} + \ldots + \gamma^{N-1-k} r_N. \tag{6}$$

Si $\gamma = 0$, el agente es llamado miope ya que sólo se enfoca en la recompensa inmediata. Si $\gamma = 1$ se obtiene un retorno $R^h(x_k)$ sin descuento. Un análisis más extenso sobre diferentes modelos de optimalidad puede ser estudiado en Koenig & Liu (2002).

2.1.2. Funciones de valor y ecuaciones de Bellman

Busoniu et al. (2017) menciona que una función de valor es una estimación de la bondad que supone para un agente estar en determinado estado cuando se sigue un política. El concepto de que tan buena es la política se refleja en el modelo de optimalidad, es decir, en el retorno, de acuerdo a Koenig & Liu (2002). Por lo anterior las funciones de valor nos ayudan a relacionar el sistema y el modelo de optimalidad elegido.

Las siguientes definiciones son tomadas directamente de Busoniu et al. (2017). Existen dos tipos de funciones de valor, las funciones de estado-acción llamadas Q-funciones y las funciones de estado, llamadas funciones V.

Definición 1 *La Q-función de una política π, $Q^\pi : X \times U \to \mathbb{R}$, es escrita como la suma descontada de recompensas cuando desde un estado x se toma una acción u, y a partir de entonces se sigue una política π:*

$$Q^{\pi}(x,u) = \sum_{i=k}^{\infty} \gamma^{i-k} \rho(x_i, u_i), \tag{7}$$

cuando $(x,u) = (x_k, u_k)$, $u_k = \pi(x_k)$ *y* $x_{k+1} = f(x_k, u_k)$.

La expresión anterior puede ser escrita usando el retorno (5) y $\pi = h$, como:

$$Q^{h}(x,u) = \rho(x,u) + \gamma R^{h}(f(x,u)). \tag{8}$$

La Q-función óptima se define como la Q-función mínima obtenida de entre todas las políticas h posibles:

$$Q^{*}(x,u) = \min_{h} Q^{h}(x,u). \tag{9}$$

Cualquier política h^* que elige en cada estado una acción con el mínimo valor de Q, satisface:

$$h^{*}(x) \in \arg\min_{u} Q^{*}(x,u), \tag{10}$$

y es óptima (optimiza el retorno $R^h(x)$).

Las funciones Q^h y Q^* pueden ser caracterizadas usando la ecuación de Bellman (Bellman, 1957). La ecuación de Bellman para una función Q^h establece el valor de tomar una acción u en un estado bajo una política h es igual a la suma de la recompensa inmediata y el valor descontado logrado por la política h en el estado siguiente:

$$Q^{h}(x,u) = \rho(x,u) + \gamma Q^{h}(f(x,u), h(f(x,u))). \tag{11}$$

La ecuación de optimalidad de Bellman satisface la Definición 1 y puede ser caracterizada usando la ecuación (9). Establece que el valor óptimo de una acción u tomada en el estado x es igual a la suma de la recompensa inmediata y el valor óptimo descontado obtenido al tomar la mejor acción en el estado siguiente $u' = h(f(x,u))$:

$$Q^{*}(x,u) = \rho(x,u) + \gamma \min_{u'} Q^{*}(f(x,u), u'). \tag{12}$$

Definición 2 *La función V de una política h, $V^h : X \to \mathbb{R}$, es el retorno obtenido al iniciar en un cierto estado x y seguir la política h, la función V puede ser calculada a partir de la Q-función, siguiendo la política h:*

$$V^h(x) = Q^h(x, u). \tag{13}$$

La relación mostrada en (13) se puede calcular a partir de (8). La función V óptima es la función V mínima que puede ser obtenida por cualquier política, y puede ser calculada a partir de la Q-función óptima:

$$V^*(x) = \min_h V^h(x) = \min_u Q^*(x, u). \tag{14}$$

Con base en (11), (13) y (14) se puede calcular h^*:

$$h^*(x) \in \arg\min_u \left\{\rho(x, u) + \gamma V^*(f(x, u))\right\}. \tag{15}$$

De acuerdo con Busoniu et al. (2017) las funciones V solo brindan información de la calidad de los estados. Las funciones V^h y V^* satisfacen las siguientes ecuaciones de Bellman:

$$V^h(x) = \rho(x, h(x)) + \gamma V^h(f(x, h(x))), \tag{16}$$

$$V^*(x) = \min_u \left\{\rho(x, u) + \gamma V^*(f(x, u))\right\}. \tag{17}$$

La idea fundamental de las ecuaciones de Bellman consiste en dividir un problema en diferentes partes, resolver cada uno y combinar las soluciones para formar una sola solución global. A diferencia de otros métodos más simples donde algunos subproblemas se resuelven múltiples veces, los métodos que utilizan las ecuaciones de Bellman para caracterizar funciones de valor, resuelven cada subproblema una única vez ((Bertsekas, 2019) y (Skiena, 2008)).

A continuación se muestra una clasificación de los algoritmos de DP y RL propuesta en Busoniu et al. (2017) basada en cómo se obtiene la política óptima.

- **Iteración de valor:** Busca la función de valor óptima, que consiste en el retorno mínimo de cada estado o estado-acción. La función de valor óptima es usada para el cálculo de la política óptima.
- **Iteración de política:** Evalúa la política construyendo sus propias funciones de valor en lugar de buscar la función de valor óptima. Usa las funciones de valor para encontrar mejores políticas.

- **Búsqueda de política:** Usa técnicas de optimización para la búsqueda directa de una política óptima.

Los algoritmos de DP requieren del conocimiento del modelo del sistema, mientras que los algoritmos de RL pueden ser libres de modelo (Sutton & Barto, 2018), y solo requieren los estados, la recompensa y el control (para las Q-funciones). A continuación se muestra una taxonomía tomando en cuenta la clasificación mencionada anteriormente de acuerdo a la forma en la que operan los algoritmos (Busoniu et al., 2017). Se muestran subrayados los algoritmos usados en este proyecto de tesis.

- **DP, algoritmos dependientes del modelo**
  - Iteración de valor
  - Iteración de política
  - Búsqueda de política
- **RL, algoritmos libres de modelo**
  - Iteración de valor
    - En línea
    - Fuera de línea
  - Iteración de política
    - En línea
    - Fuera de línea
  - Búsqueda de política
    - En línea
    - Fuera de línea

2.2. Programación dinámica: algoritmos dependientes del modelo

El término de programación dinámica consiste en una colección de algoritmos para obtener la solución óptima a un problema de decisión secuencial (e.g., MDP) (Sutton & Barto, 2018), del cual se tiene conocimiento del modelo de comportamiento del mismo (i.e., $f(x_k, u_k)$). En consecuencia, queda fuera del alcance de DP todos aquellos problemas en los cuales no se pueda acceder a un modelo exacto. La programación dinámica se basa en el *principio de optimalidad de Bellman* (Bellman, 1957), que de acuerdo a lo que se menciona en Lewis et al. (2012b), establece lo siguiente:

"Una política óptima tiene la propiedad de que no importan cuales hayan sido las acciones de control previas, las acciones de control restantes constituyen una política óptima con respecto al estado resultante de esas acciones previas".

A continuación se presenta el análisis de los algoritmos de iteración de valor e iteración de política dependientes del modelo desde un punto de vista determinista usando Q-funciones, tomado directamente de Busoniu et al. (2017).

2.2.1. Iteración de valor

El algoritmo de iteración de valor (ver Algoritmo 1) usa la ecuación de optimalidad de Bellman para calcular de forma iterativa la función de valor óptima, desde la cual se construirá la política óptima.

Sea $\mathcal{Q}$ el espacio de las Q-funciones. Se define el operador de Bellman[1] $\mathcal{H} : \mathcal{Q} \to \mathcal{Q}$ como:

$$\left[\mathcal{H}(Q^{h_j})\right](x,u) = \rho(x,u) + \gamma \min_{u'} Q^{h_j}(f(x,u),u'), \tag{18}$$

donde $\mathcal{H}$ es un mapeo que calcula iterativamente el lado derecho de la ecuación de optimalidad de Bellman para cualquier Q-función. Observe que $[\mathcal{H}(Q^h)]$ denota aplicar el operador de Bellman a una Q-función, y evaluar un par estado-acción disponible.

El algoritmo de iteración para la Q-función inicia desde una Q-función arbitraria Q_0^h, y en cada iteración j actualiza la Q-función usando:

$$Q^{h_{j+1}} = \mathcal{H}(Q^{h_j}). \tag{19}$$

Se puede demostrar que $\mathcal{H}$ es una contracción con factor $\gamma < 1$ en la norma infinito de acuerdo a Busoniu et al. (2017), es decir, para cualquier par de funciones $Q, Q' \in \mathcal{Q}$, es cierto que:

$$\|\mathcal{H}(Q) - \mathcal{H}(Q')\|_\infty \leq \gamma \|Q - Q'\|_\infty, \tag{20}$$

debido a que $\mathcal{H}$ es una contracción, tiene un único punto fijo, además, la ecuación de optimalidad de Bellman (12) establece que Q^* es el punto de fijo de $\mathcal{H}$, es decir,

$$Q^* = \mathcal{H}(Q^*). \tag{21}$$

Por lo tanto, el único punto fijo de $\mathcal{H}$ es Q^* y la iteración de la Q-función converge asintóticamente a

[1] Llamado *mapeo* $\mathcal{H}$ en Busoniu et al. (2017) y como *operador de Bellman* en Perrusquia et al. (2019).

Q^* cuando $j \to \infty$. Además, la iteración de la Q-función converge en una tasa de γ, en el sentido que $\|Q^{h_{j+1}} - Q^*\|_\infty \leq \gamma \|Q^{h_j} - Q^*\|_\infty$, esto último se deduce de (19) y (20). Una política óptima puede ser calculada desde Q^* usando (10).

Algoritmo 1: Iteración de valor (Q-función), tomado de Busoniu et al. (2017)

Entrada: Estado inicial x_0, dinámica $f(x,u)$, recompensa $\rho(x,u)$ y factor de descuento γ;
Inicializa la Q-función, e.g., $Q^{h_0} \leftarrow 0$;
En cada iteración $j = 0, 1, 2, \ldots$ **repetir**
    % Actualización de valor;
    para *cada* (x,u) **hacer**

$$Q^{h_{j+1}}(x,u) \leftarrow \rho(x,u) + \gamma \min_{u'} Q^{h_j}(f(x,u),u') \tag{22}$$

    fin
hasta que $Q^{h_{j+1}} = Q^{h_j}$;
Salida: $Q^* = Q^{h_j}$

2.2.2. Iteración de política

El algoritmo de iteración de política, constituye la segunda mayor clase de algoritmos de DP y RL de acuerdo a Busoniu et al. (2017). El algoritmo de iteración de política evalúa sus políticas mediante la construcción de sus funciones de valor y utiliza estas funciones para encontrar nuevas y mejores políticas. Este algoritmo inicia con una política arbitraria h_0 y en cada iteración j, la función de valor Q^{h_j} de la política h_j es determinada (Busoniu et al., 2017), a este paso se le conoce como: *evaluación de la política* (predicción). La evaluación de la política se realiza mediante la resolución de la ecuación de Bellman (11). A continuación se describe el algoritmo iterativo de evaluación de política de manera similar a la iteración de valor. Sea $\mathcal{Q}$ el espacio de las Q-funciones. Se define el operador de Bellman bajo una política $\mathcal{H}^{h_j} : \mathcal{Q} \to \mathcal{Q}$ como:

$$\left[\mathcal{H}^{h_j}(Q^{h_j})\right](x,u) = \rho(x,u) + \gamma Q^{h_j}(f(x,u), h(f(x,u))), \tag{23}$$

Cuando la evaluación de la política es completada, una nueva política h_{j+1} es encontrada mediante:

$$h_{j+1}(x) \in \arg\min_u Q^{h_j}(x,u). \tag{24}$$

El paso descrito por (24) se le conoce como *mejoramiento de la política* de acuerdo con Busoniu et al. (2017). El procedimiento completo es mostrado en el Algoritmo 2. La secuencia de Q-funciones producidas por la iteración de política converge asintóticamente a Q^* cuando $j \to \infty$ y simultáneamente, una política h^* es obtenida.

Algoritmo 2: Iteración de política (Q-función), tomado de Busoniu et al. (2017)

Entrada: Estado inicial x_0, dinámica $f(x,u)$, recompensa $\rho(x,u)$ y factor de descuento γ;
Inicializa la política h_0;
En cada iteración $j = 0, 1, 2, \ldots$ **repetir**
    % Evaluación de la política;
        Encuentra Q^{h_j}, la Q-función de h_j;
    % Mejoramiento de la política;

$$h_{j+1}(x) \in \arg\min_u Q^{h_j}(x,u)$$

hasta que $h_{j+1} = h_j$;
Salida: $h^* = h_j$, $Q^* = Q^{h_j}$

El Algoritmo 1 considera solo la etapa de la actualización de la Q-función, mientras que en el Algoritmo 2 las etapas de evaluación y mejoramiento se repiten alternativamente hasta la convergencia a una política óptima.

2.2.3. Programación dinámica determinista

Es común que el análisis sea más compacto para trabajar cuando está en tiempo continuo, sin embargo, los problemas en tiempo discreto permiten una comprensión simple de la programación dinámica, y para sistemas de control digital en aplicaciones reales, el tratamiento en tiempo discreto resulta suficiente (Tedrake, 2009).

El problema involucra la forma de una función de costo como recursión, que indica el índice de desempeño, como la suma del costo por etapa en el instante k descrito por $L(x_k, u_k)$, el cual se debe especificar al momento de definir la función de costo. Para un estado inicial x_0, la función de costo total de una secuencia de control $\{u_0, \ldots, u_{N-1}\}$ es:

$$\mathcal{J}(x_0; u_0, \ldots, u_{N-1}) = h(x_N) + \sum_{k=0}^{N-1} L(x_k, u_k), \tag{25}$$

donde $h(x_N)$ es el costo terminal. Este costo es un número bien definido debido a la secuencia de control $\{u_0, \ldots, u_{N-1}\}$, que junto con x_0 determina la secuencia de estados $\{x_1, \ldots x_N\}$ a través de la ecuación (1) de acuerdo a Bertsekas (2019).

Siguiendo el principio de optimalidad de Bellman es posible encontrar una secuencia óptima de control para un problema que involucre una función de costo como (25). La Figura 10 ilustra del concepto del principio de optimalidad de Bellman para una trayectoria que en x_0 selecciona u_0^*.

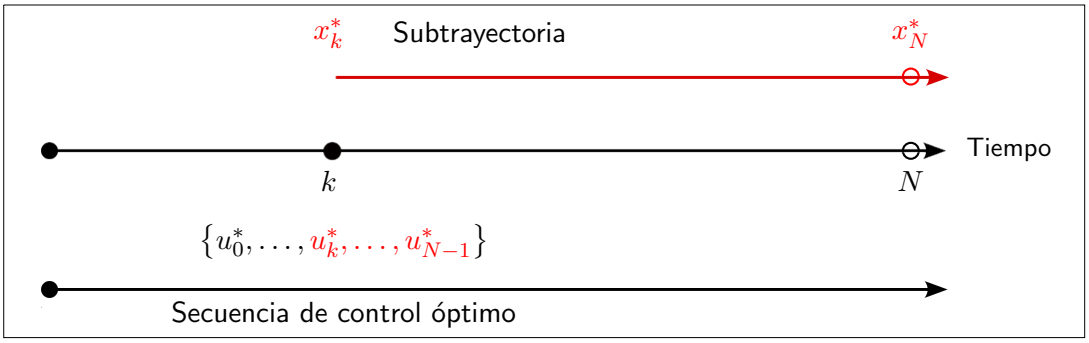

Figura 10. Ilustración conceptual del principio de optimalidad de Bellman mostrada en Bertsekas (2019). La cola $\{u_k^*, \ldots, u_{N-1}^*\}$ de una secuencia óptima $\{u_0^*, \ldots, u_{N-1}^*\}$, es óptima para la subtrayectoria que inicia en el estado x_k^* de la trayectoria óptima $\{x_1^*, \ldots, x_N^*\}$.

Considérese un sistema descrito por una dinámica no lineal usando ecuaciones en diferencias de la forma:

$$x_{k+1} = f(x_k) + g(x_k)u_k. \tag{26}$$

El sistema descrito por (26) es un caso particular de (1), donde $x_k \in \mathbb{R}^n$, $u_k \in \mathbb{R}^m$, $f : \mathbb{R}^n \to \mathbb{R}^n$ y $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$. Suponemos que $f + gu$ es continua y Lipschitz en un conjunto $\Omega \in \mathbb{R}^n$, que además, es controlable en el sentido de que existe un control en Ω que en lazo cerrado estabiliza asintóticamente al sistema en cuestión.

El objetivo es encontrar un control realimentado u_k para minimizar la energía del estado final y toda la secuencia de estados intermedios, así como los controles para cada uno de estos estados. Minimizar la energía corresponde a mantener el estado y el control muy cercanos a cero, y son conocidos como problemas de mínima energía de acuerdo a Lewis et al. (2012b). Se define la función de costo (25) de forma cuadrática matricial como sigue:

$$\mathcal{J}_k = x_N^T P x_N + \sum_{k=0}^{N-1} (x_k^T Q x_k + u_k^T R u_k), \tag{27}$$

donde $Q \in \mathbb{R}^{n\times n}$ satisface $Q = Q^T \geq 0$, $R \in \mathbb{R}^{m\times m}$ satisface $R = R^T > 0$ y alguna matriz $P \in \mathbb{R}^{n\times n}$ satisface $P = P^T > 0$.

2.2.3.1. Análisis no lineal en tiempo discreto

El análisis que se muestra a continuación es propuesto en Brunton & Kutz (2019). La función V que toma los valores de la trayectoria con respecto a los estados x_k con $k \in [0, N]$, es definida como los costos acumulados iniciando de un estado x_0 tomando una política de control que se supone es óptima:

$$V(x_0, N) = \min_{\{u_k\}_{k=0}^{N-1}} \mathcal{J}(x_0; u_0, \ldots, u_{N-1}). \tag{28}$$

Tomando un punto intermedio de la trayectoria, es decir, x_k como un estado inicial de la nueva trayectoria $k \in (0, N]$, la secuencia de controles restantes es óptima con respecto a este nuevo estado inicial:

$$V(x_0, N) = V(x_0, k) + V(x_k, N). \tag{29}$$

El valor de este nuevo estado inicial, considerando la definición cuadrática de (27), se define como:

$$V(x_k, N) = \min_{u_k} \left\{ x_k^T Q x_k + u_k^T R u_k \right\} + V(x_{k+1}, N), \qquad k = 1, \ldots, N-1. \tag{30}$$

Usando la dinámica no lineal descrita en (26) como la función de transición para x_{k+1}, se puede obtener el valor óptimo para toda la trayectoria:

$$V^*(x_k, N) = \min_{u_k} \left\{ x_k^T Q x_k + u_k^T R u_k + V((f(x_k) + g(x_k)u_k), N) \right\}. \tag{31}$$

Debido a la recursividad de la función de valor para algún estado intermedio, la dependencia del tiempo final N para la función V en (31) no es necesaria (Brunton & Kutz, 2019), por tanto,

$$V^*(x_k) = \min_{u_k} \left\{ x_k^T Q x_k + u_k^T R u_k + V(f(x_k) + g(x_k)u_k) \right\}. \tag{32}$$

La ecuación (32) corresponde a la ecuación Hamilton-Jacobi-Bellman (HJB, por sus siglas en inglés) en tiempo discreto y es similar a la ecuación de Bellman (17). Se supone que se encuentra la u_k^* para reescribir la ecuación (32)[2]:

$$x_k^T Q x_k + (u_k^*)^T R u_k^* + V^*(f(x_k) + g(x_k)u_k^*) - V^*(x_k) = 0, \tag{33}$$

con su solución u_k^* como:

$$\frac{\partial x_k^T Q x_k}{\partial u_k^*} + \frac{\partial (u_k^*)^T R u_k^*}{\partial u_k^*} + \frac{\partial V^*(x_{k+1})}{\partial u_k^*} - \frac{\partial V^*(x_k)}{\partial u_k^*} = 0, \tag{34}$$

$$\frac{\partial V^*(x_{k+1})}{\partial x_{k+1}} g(x_k) + 2(u_k^*)^T R = 0, \tag{35}$$

$$u_k^* = -\frac{1}{2} R^{-1} g^T(x_k) \left(\frac{\partial V^*(x_{k+1})}{\partial x_{k+1}} \right)^T. \tag{36}$$

Particularmente la ecuación (36) es de importancia en la metodología que se usará en esta tesis.

2.2.3.2. Control óptimo: LQR en tiempo discreto

Con la finalidad de entender la relación del aprendizaje por refuerzo con el control óptimo, se estudia un caso característico de programación dinámica, el regulador cuadrático lineal (LQR, por sus siglas en inglés), el cual es claramente el caso más representativo de control óptimo y requiere del conocimiento del modelo que describe la dinámica del sistema. En esta subsección se presentan ecuaciones y el procedimiento tomado directamente de Lewis et al. (2012b).

Considérese el problema del LQR en tiempo discreto, donde el objetivo es diseñar un controlador que minimice una función de costo, donde el MDP es determinista y satisface la ecuación de transición de estados:

$$x_{k+1} = Ax_k + Bu_k, \tag{37}$$

[2]La representación de la ecuación HJB en tiempo discreto, descrita en (33) es presentada en Zheng Chen & Jagannathan (2008).

el índice de tiempo discreto es representado por k y el índice de desempeño de horizonte infinito asociado, tiene un costo por etapa que se representa como:

$$\mathcal{J}_k = V^h(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} \rho(x_i, u_i).$$

Definiendo

$$\rho(x_i, u_i) = x_i^T Q x_i + u_i^T R u_i \tag{38}$$

tenemos:

$$\mathcal{J}_k = \sum_{i=k}^{\infty} \gamma^{i-k} (x_i^T Q x_i + u_i^T R u_i), \tag{39}$$

con el espacio de estados $x \in \mathbb{R}^n$ y el espacio de acciones $u \in \mathbb{R}^m$, donde $\gamma \in [0, 1)$ es el factor de descuento. Una ecuación en diferencias equivalente a (39) es:

$$V^h(x_k) = x_k^T Q x_k + u_k^T R u_k + \sum_{i=k+1}^{\infty} \gamma^{i-k} (x_i^T Q x_i + u_i^T R u_i), \tag{40}$$

se reescribe (40) como la ecuación de Bellman (16):

$$V^h(x_k) = x_k^T Q x_k + u_k^T R u_k + \gamma V^h(x_{k+1}). \tag{41}$$

Aunque el análisis anterior se realiza con funciones V como en Lewis et al. (2012a), puede ser fácilmente extendido a Q-funciones debido a la relación descrita por (13). Suponiendo que la función de valor es cuadrática en el estado de acuerdo a Lewis et al. (2012b):

$$V^h(x_k) = x_k^T P x_k, \tag{42}$$

la ecuación (41) será:

$$V^h(x_k) = x_k^T Q x_k + u_k^T R u_k + x_{k+1}^T \gamma P x_{k+1}, \tag{43}$$

la cual, usando la ecuación de transición de estados (37), puede ser reescrita como:

$$V^h(x_k) = x_k^T Q x_k + u_k^T R u_k + (A x_k + B u_k)^T \gamma P (A x_k + B u_k). \tag{44}$$

Ahora, considerando la siguiente política realimentada:

$$u_k^* = -Kx_k, \tag{45}$$

donde $K \in \mathbb{R}^{m \times n}$, y usando (45) en (44) e igualando con (42), se obtiene:

$$V^h(x_k) = x_k^T P x_k = x_k^T Q x_k + (-Kx_k)^T R(-Kx_k) + (Ax_k + B(-Kx))^T \gamma P(Ax_k + B(-kx_k)),$$

$$x_k^T P x_k = x_k^T Q x_k + x_k^T K^T R K x_k + x_k^T (A - BK)^T \gamma P (A - BK) x_k,$$

$$0 = -x_k^T P x_k + x_k^T Q x_k + x_k^T K^T R K x_k + x_k^T (A - BK)^T \gamma P (A - BK) x_k,$$

$$0 = x_k^T (-P + Q + K^T R K + (A - BK)^T \gamma P (A - BK)) x_k. \tag{46}$$

Debido a que la ecuación (46) se mantiene para todas las trayectorias del estado, se obtiene:

$$(A - BK)^T \gamma P (A - BK) - P + Q + K^T R K = 0, \tag{47}$$

la cual cumple con ser una ecuación de Lyapunov como se indica en Lewis et al. (2012a), por lo que la ecuación de Bellman para el problema del LQR en tiempo discreto es equivalente a una ecuación de Lyapunov (ver e.g., Khalil (2015)).

La función Hamiltoniano Ha del LQR en tiempo discreto se describe en Lewis et al. (2012a) como:

$$Ha(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k + (Ax_k + Bu_k)^T \gamma P (Ax_k + Bu_k) - x_k^T P x_k. \tag{48}$$

En Lewis et al. (2012a) se menciona que una condición necesaria para la optimalidad es la condición estacionaria: $\frac{\partial Ha(x_k, u_k)}{\partial u_k} = 0$, por tanto, resolviendo (48) bajo esta condición se obtiene el control óptimo:

$$u_k^* = -Kx_k = -\left(B^T P B + \frac{1}{\gamma} R\right)^{-1} B^T P A x_k. \tag{49}$$

Usando (49) en (47) reproducimos la ecuación algebraica de Riccati en tiempo discreto (DARE, por sus

siglas en ingles) como se indica en Lewis et al. (2012a):

$$A^T PA - P + Q - A^T PB(B^T PB + \frac{1}{\gamma}R)^{-1}B^T PA = 0. \tag{50}$$

La DARE, descrita por (50), es exactamente la ecuación de optimalidad de Bellman (17), para el problema del LQR en tiempo discreto.

2.2.3.3. Q-función para el LQR en tiempo discreto

Considerando la siguiente ecuación de Bellman:

$$Q^h(x_k, u_k) = \rho(x_k, u_k) + \gamma Q^h(x_{k+1}, u_{k+1}), \tag{51}$$

a partir de (13) se tiene:

$$Q^h(x_k, u_k) = \rho(x_k, u_k) + \gamma V^h(x_{k+1}). \tag{52}$$

Tomando V^h dada por (42), junto con (37) y (38) se obtiene:

$$Q^h(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k + (Ax_k + Bu_k)^T \gamma P(Ax_k + Bu_k) = \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T H \begin{bmatrix} x_k \\ u_k \end{bmatrix}, \tag{53}$$

donde

$$H = \begin{bmatrix} H_{xx} & H_{ux}^T \\ H_{ux} & H_{uu} \end{bmatrix} = \begin{bmatrix} Q + \gamma A^T PA & \gamma A^T PB \\ \gamma B^T PA & R + \gamma B^T PB \end{bmatrix}. \tag{54}$$

Resolviendo $\frac{\partial Q^h(x_k, u_k)}{\partial u_k} = 0$, se obtiene el control óptimo:

$$u_k^* = -H_{uu}^{-1} H_{ux} x_k \tag{55}$$

La ecuación (49) requiere del conocimiento del modelo del sistema para calcular una política óptima. Por otro lado (55) depende de las submatrices H_{uu} y H_{ux} de la Q-función, mismas que son estimadas en tiempo real por el algoritmo de aprendizaje, conocido como Q-learning, el cual necesita únicamente

los datos medidos a lo largo de las trayectorias, eliminando la necesidad del conocimiento del modelo. De acuerdo a Lewis et al. (2012a) esto es equivalente a resolver la ecuación algebraica de Riccati (50), pero sin conocer el modelo del sistema.

Es importante mencionar que el algoritmo de iteración de valor e iteración de política pueden desarrollarse para el caso del LQR cuando se tiene conocimiento del modelo del sistema, a estos métodos se les conoce como *Recursión de Lyapunov* y *Algoritmo de Hewer*, respectivamente, un tratamiento más detallado y formal de esto puede consultarse en Lewis & Vamvoudakis (2011), Lewis et al. (2012a) y Vrabie et al. (2013).

2.3. Aprendizaje por refuerzo: algoritmos libres de modelo

En esta sección abordamos los algoritmos de RL implementados en la metodología de esta tesis, describiendo el funcionamiento y las características que los diferencia. Características que los vuelven idóneos para aplicaciones de control realimentado, donde se tiene poco o nulo conocimiento del modelo que describe la dinámica del sistema. Considere que la manera de describir los algoritmos puede variar dependiendo del autor (ver (Sutton & Barto, 2018), (Bertsekas, 2019) y (Busoniu et al., 2017)) y la aplicación de interés (i.e., MDPs deterministas, MDPs estocásticos, espacios de estado y acción discretos o contables, espacios de estado y acción continuos).

Los dos principales métodos de RL para desempeñar los algoritmos de iteración de política e iteración de valor son; *Monte Carlo* y *Aprendizaje por Diferencias Temporales* (Sutton & Barto, 2018). Ambos métodos pueden ser implementados sin conocer el modelo del sistema, es decir, solamente requieren de los datos medidos (observados) a lo largo de las trayectorias del sistema x_{k+1}. Esta tesis abarca el método de diferencias temporales y un algoritmo derivado del mismo.

2.3.1. Aprendizaje por diferencias temporales

El método de aprendizaje por diferencias temporales (TD[3], por sus siglas en inglés), lidera una familia de controladores adaptativos, debido a que se desempeña completamente en línea, dando solución a la ecuación de Bellman y estimando funciones de valor sin conocer el modelo del sistema, únicamente

[3]Definido como TD(0) en (Sutton & Barto, 2018).

usando los datos medidos a lo largo de las trayectorias del sistema[4]. Por lo anterior, los métodos basados en aprendizaje por TD son ideales para tareas de control realimentado que buscan leyes de control óptimo, además, estos métodos son relacionados con control adaptativo en el sentido de que ajustan sus valores y acciones en línea con el sistema y pueden ser implementados en tiempo real (Lewis et al., 2012a).

La iteración de política requiere la solución de N ecuaciones lineales con respecto a la ecuación de Bellman, en cada paso. La iteración de valor requiere desempeñar una recursión de la ecuación de Bellman, en cada paso. El aprendizaje por TD actualiza la función de valor en cada instante de tiempo y la medición de datos es realizada a lo largo de las trayectorias del sistema y el nuevo valor es usado periódicamente para el mejoramiento de la política. El aprendizaje por diferencias temporales puede ser considerado una técnica de aproximación estocástica (Yu & Perrusquía, 2021), donde su versión de la ecuación de Bellman es remplazada por la evaluación en un solo camino muestreado del MDP, que posteriormente pasa a ser la ecuación de Bellman determinista (Lewis et al., 2012a).

El *error de diferencias temporales* para la ecuación de Bellman determinista (11) puede ser definido de acuerdo a Lewis et al. (2012a), como:

$$e_{TD} = -V^h(x_k) + \rho(x_k, h(x_k)) + \gamma V^h(x_{k+1}). \tag{56}$$

El método de aprendizaje por TD se muestra en la Figura 11, como mecanismo a la solución de la ecuación de Bellman. La idea es usar la ecuación de Bellman (11) en el algoritmo de iteración de valor o en el de iteración de política para minimizar el error de diferencias temporales (56).

En la Figura 11 se muestra que a partir de una recompensa $\rho(x_k, h(x_k))$ como consecuencia de aplicar una acción de control en el estado actual, podemos calcular el valor $V^h(x_k)$ para el estado actual. La transición al estado siguiente nos permite calcular el valor $V^h(x_{k+1})$ para el estado siguiente, a lo que se conoce como valor estimado. Está idea permite actualizar la función de valor para cada estado visitado usando la suma de la recompensa $\rho(x_k, h(x_k))$ más el valor estimado $V^h(x_{k+1})$, sin necesidad de esperar el final del episodio.

[4]El término *trayectorias del sistema*, se refiere al conjunto de trayectorias con respecto a los estados. Estas trayectorias son medidas solo una vez para TD, esta idea hace referencia a usar solo un camino muestreado de un MDP (Bertsekas, 2019).

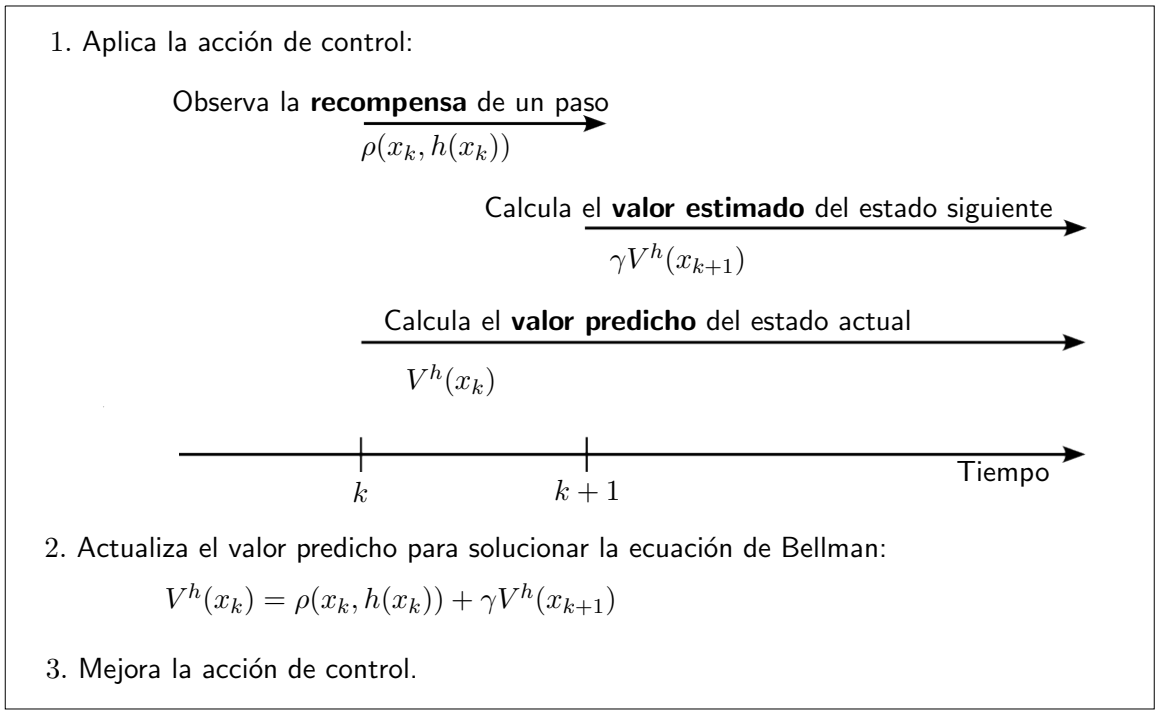

Figura 11. El aprendizaje por diferencias temporales describe como la ecuación de Bellman captura el funcionamiento de la acción, la observación, la evaluación y el mejoramiento, ilustración tomada de Lewis et al. (2012a).

2.3.2. Algoritmo Q-learning para control óptimo adaptativo

El principal algoritmo de RL es conocido como Q-learning de acuerdo con Wang & Hong (2020), desarrollado por Christopher J. Watkins (ver (Watkins, 1989) y (Watkins & Dayan, 1992)) y Paul J. Werbos (ver (Werbos, 1987) y (Werbos et al., 1991)). Es conocido como un método simple para trabajar con MDPs desconocidos, es decir, que no se conoce el modelo del sistema o la función de transición. El algoritmo Q-learning es un método de TD llamado por Werbos como *Programación Dinámica Heurística - Dependiente de la Acción* (ADHDP, por sus siglas en inglés) (Wang et al., 2009). El algoritmo Q-learning recibe dicho nombre debido a que las Q-funciones son dependientes de la acción de control.

El algoritmo Q-learning es considerado un método *fuera de política* (off-policy[5]) y puede ser implementado completamente en línea debido a que se desempeña como el aprendizaje por TD (Lewis et al., 2012a). De hecho, la descripción dada por Watkins (1989) para Q-learning es basada en el Algoritmo 1,

[5]El término *off-policy* se refiere a los métodos que evalúan o mejoran una política de control diferente a la que usaron para la generación de los datos observados (Sutton & Barto, 2018). Los métodos *on-policy* evalúan o mejoran la misma política usada para la generación de los datos observados, los algoritmos Monte Carlo ES o SARSA generalmente son un ejemplo de este término (Sutton & Barto, 2018).

y es posible observar que consiste en un método iterativo para la actualización de la Q-función:

$$Q_{k+1}^h(x_k, u_k) = Q_k^h(x_k, u_k) + \alpha_k \left[\rho(x_k, u_k) + \gamma \min_{u'} Q_k^h(x_{k+1}, u') - Q_k^h(x_k, u_k) \right]. \tag{57}$$

Algunos trabajos recientes relacionados al diseño de controladores basados en la ecuación (57) para sistemas mecánicos y robóticos, se pueden ver en Puriel Gil et al. (2019) y Perrusquía et al. (2021).

Una particularidad que menciona Watkins (1989) es que una Q-función para un control u estabilizante se define como (51). La metodología propuesta por Bradtke et al. (1994) es ideal para la solución al problema del LQR usando Q-funciones sin conocer el modelo del sistema a través de lo que él llama *estimación directa de Q-funciones*. Las pruebas de convergencia para (51) se pueden ver en Bradtke et al. (1994) y Al-Tamimi (2007).

2.3.3. Actor-crítico

El método *actor-crítico* es una técnica de aprendizaje por refuerzo que usa aprendizaje por TD para la toma de decisiones secuenciales, e introduce dos componentes para mejorar el entrenamiento de un agente (Sutton & Barto, 2018). Resultando en una estructura que provee el mejor esquema de trabajo para espacios de estado y acción continuos en tareas de control, debido a que mantiene separada la estructura para evaluar o actualizar la función de valor, llamada crítico, y la estructura correspondiente a mejorar la política de control, llamada actor.

El esquema de la Figura 12 es ideal para algoritmos que usan funciones V, debido a que en las funciones V cada espacio continuo necesita un aproximador; uno para el crítico y otro para el actor. Los algoritmos que usen Q-funciones (e.g., Q-learning) solo requieren de un aproximador. En la Figura 13 se muestra como la actualización de Q-función y el mejoramiento de la política se pueden implementar en la estructura de un aproximador.

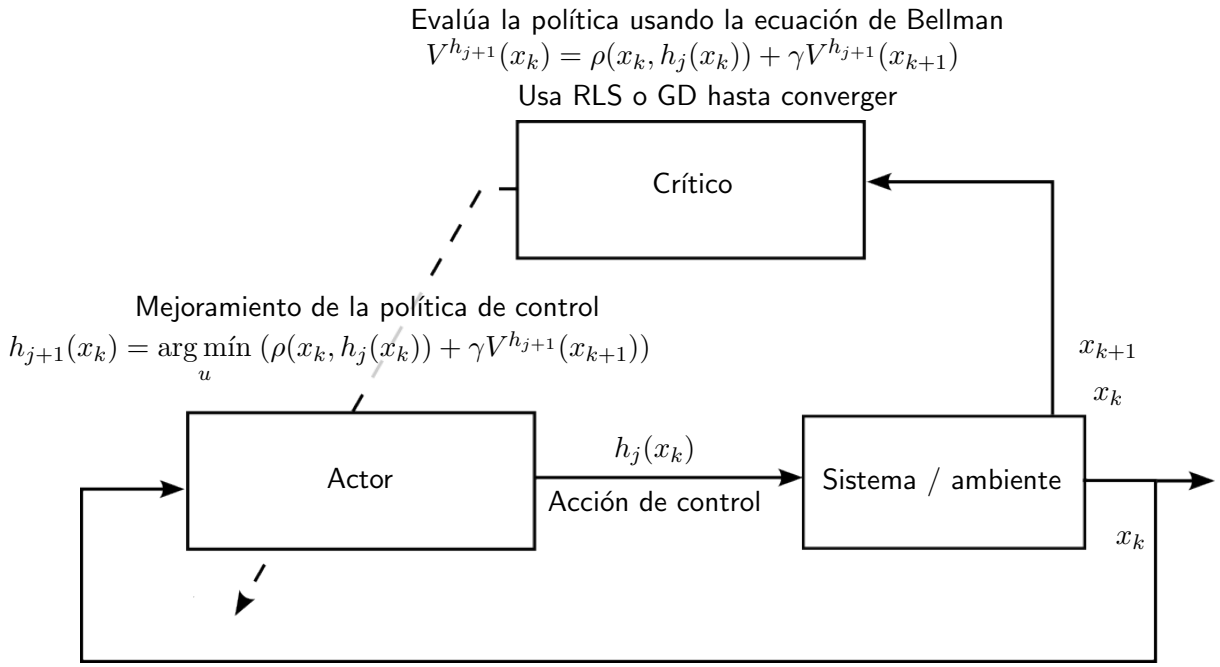

Figura 12. El método de aprendizaje por TD usando iteración de política puede ser representado en el esquema de actor-crítico de acuerdo Vrabie et al. (2013). Las estructuras de los aproximadores son representadas por los rectángulos verdes.

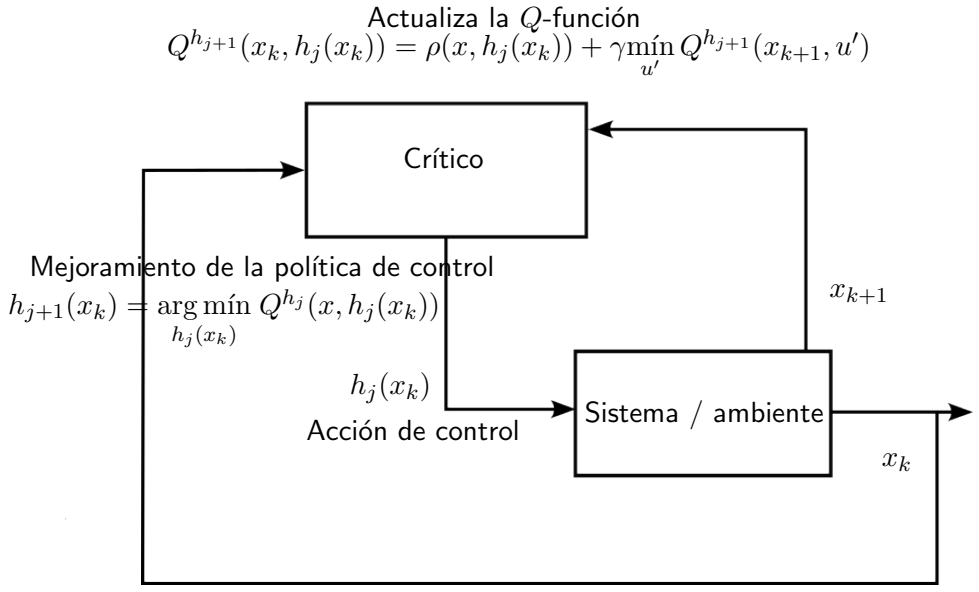

Figura 13. La representación del algoritmo Q-learning para el LQR, en el esquema de actor-crítico solo requiere un aproximador debido a la naturaleza de las Q-funciones. Considérese la representación de un aproximador como la unión de las tareas desempeñadas por el actor y el crítico, además, $u' = h_j(x_{k+1})$.

2.3.4. Aproximación de la función de valor

En DP, cuando el estado de un sistema y el control aplicable pertenecen a conjuntos finitos, se puede hacer una búsqueda exhaustiva y saber cuál acción de control optimiza un criterio de desempeño. Cuando el estado o el control pertenecen a espacios continuos la solución exacta ya no es posible, por lo que es necesario usar soluciones aproximadas para las funciones de valor y las políticas. En Busoniu et al. (2017) se menciona que para la implementación de los algoritmos de iteración de valor o iteración de política vistos en la secciones 2.2.1 y 2.2.2 respectivamente, es imposible recorrer los espacios de estados y acciones infinitos en un tiempo finito, como lo requiere la ecuación (22) o la ecuación (6).

Una alternativa a la búsqueda exhaustiva es la *aproximación de la función de valor* (VFA, por sus siglas en inglés) (Lewis & Vrabie, 2009). Esta forma de aprendizaje provee una aproximación del espacio continuo a partir de un cojunto discreto muestreado (Fernandez-Gauna et al., 2018). La VFA es esencial en DP y RL, pero requiere del muestreo del espacio continuo de las trayectorias del sistema para ser representada, incluidos los controles u_k para el caso de aproximación de Q-funciones.

La Figura 14 ilustra los mapeos involucrados en la aproximación de la función V, donde x_k representa las muestras discretas medidas en el espacio continuo de estados. De forma similar, la Figura 15 muestra los mapeos de la aproximación de la Q-función. El uso de la VFA es llamado por Werbos como *Programación Dinámica Aproximada* (ADP, por sus siglas en inglés). En Wang et al. (2009) se da una revisión a la literatura de programación dinámica aproximada como introducción, donde se mencionan los diferentes nombres que recibe.

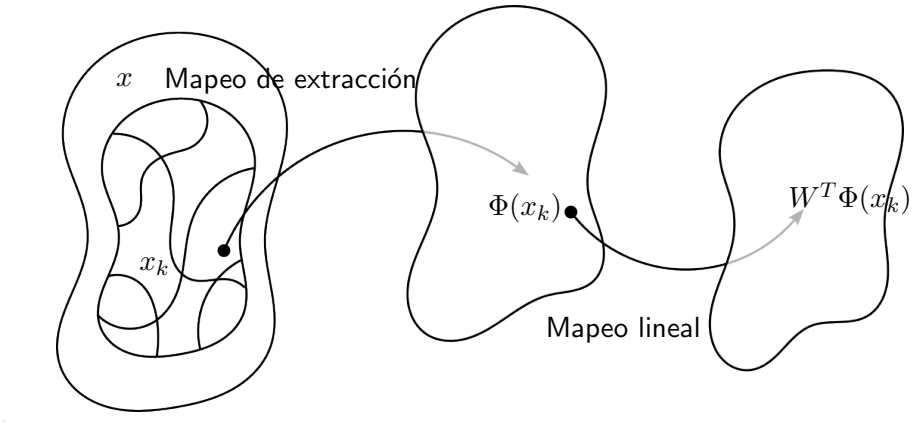

Figura 14. La forma funcional de la VFA para las funciones V, consiste en un mapeo de extracción que va del espacio de estado al espacio de funciones, con el objetivo de generar la entrada a un mapeo lineal que sirva para generar la aproximación descrita por $W^T\Phi(x_k)$.

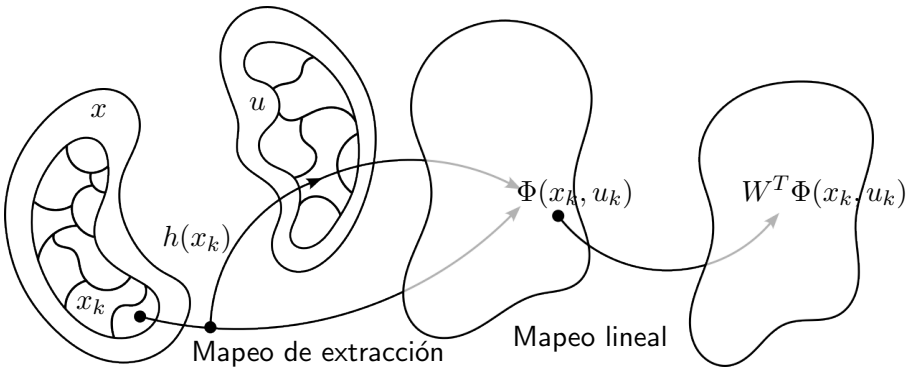

Figura 15. La forma funcional de la VFA para las Q-funciones, consiste en un mapeo de extracción que va del espacio de estado al espacio de funciones, y una política como mapeo sobre el estado para definir la acción de control hacia el espacio de funciones, con el objetivo de generar la entrada a un mapeo lineal que sirva para generar la aproximación descrita por $W^T\Phi(x_k, u_k)$.

La siguiente sección se enfocará en describir el tipo de aproximador usado en este proyecto de tesis, conocido como *aproximador paramétrico*. Posteriormente, describimos las técnicas de identificación de sistemas usadas para sintonizar los parámetros de W.

2.3.4.1. Aproximadores paramétricos

En Busoniu et al. (2017) se define a un aproximador paramétrico como un mapeo del espacio de parámetros al espacio de funciones, cuyo objetivo es representar funciones de valor. En un aproximador paramétrico, el número de parámetros y la forma son definidos a priori y no dependen de los datos, por lo tanto, la estructura del aproximador paramétrico es fija. La estructura consiste de los siguientes elementos:

- $\hat{W}$: vector de parámetros aproximado. Por simplicidad en la notación se usará W.
- Φ: vector de funciones base.
- x_k: estado al instante k.
- u_k: control al instante k.

Se considera un aproximador de la Q-función como la parametrización dada por un vector $W \in \mathbb{R}^s$, donde $\mathbb{R}^s$ es el espacio de los parámetros. El aproximador es denotado por $\hat{Q}^h$ y está dado por el mapeo $\mathcal{F} : \mathbb{R}^s \to \mathcal{Q}$:

$$\hat{Q}^h(x, u) = [\mathcal{F}(W)](x, u). \tag{58}$$

La ecuación (58) describe la Q-función aproximada como $[\mathcal{F}(W)]$ evaluado en x al tomar una acción u. Intuitivamente esta idea consiste en que en lugar de almacenar distintas Q-funciones para cada par estado acción disponible, solo será necesario almacenar s parámetros, que sirvan para describir la $\hat{Q}^h$ desde cualquier estado visitado y acción tomada. Un aproximador emplea s funciones base de acuerdo a Busoniu et al. (2017). Se define el cálculo de la aproximación como:

$$[\mathcal{F}(W)](x,u) = \sum_{i=1}^{s} w_i \phi_i(x,u) = W^T \Phi(x,u), \tag{59}$$

donde x y u contienen los componentes de x_k y u_k respectivamente, por lo que $\Phi : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^s$:

$$\Phi(x_k,u_k) = [\phi_1(x_k,u_k), \phi_2(x_k,u_k), \ldots, \phi_s(x_k,u_k)]^T \tag{60}$$

De acuerdo a lo mencionado por Busoniu et al. (2017) y Lewis et al. (2012a), el mapeo $\mathcal{F}$ puede ser no-lineal en los parámetros, (e.g., redes neuronales, con sus funciones de activación como funciones base: sigmoide, tangente hiperbólica, etc.). Sin embargo, los aproximadores linealmente parametrizables son preferibles en ADP y RL para tareas de control, debido a que hacen más fácil el análisis del resultado.

Este proyecto de tesis hace uso de un conjunto base polinomial como funciones base, idea descrita por Lewis et al. (2012a).

2.3.4.2. Aproximación de la Q-función a través de la iteración

A continuación se presenta un análisis tomado directamente de Busoniu et al. (2017), respecto al funcionamiento de la aproximación de la Q-función, que resulta adecuado para entender cómo trabajan los algoritmos de ADP y RL[6].

En la sección 2.2.1 mencionamos que el Algoritmo 1 comienza con una Q-función arbitraria Q^{h_0} y en cada iteración j actualiza la Q-función. Se describe el paso de actualización usando el operador de Bellman visto en la sección 2.2.1, ecuación (18):

$$Q^{h_{j+1}} = \mathcal{H}(Q^{h_j}). \tag{61}$$

[6] Aunque el análisis es con respecto a la iteración de valor usando la Q-función, la idea puede ser extendida a la iteración de política, o considerando funciones V.

La función Q^{h_j} no puede ser almacenada de manera exacta, por ello se usa el mapeo $\mathcal{F}$ visto en la sección 2.3.4.1 para describir la Q-función con su versión aproximada:

$$\hat{Q}^{h_j} = \mathcal{F}(W_j). \tag{62}$$

La función $Q^{h_{j+1}}$ de la ecuación (61) tampoco puede ser almacenada exactamente, por lo que su representación es a través de un nuevo vector de parámetros W_{j+1}. Este vector se obtiene usando un nuevo mapeo de proyección $\mathcal{P} : \mathcal{Q} \to \mathbb{R}^s$:

$$W_{j+1} = \mathcal{P} \circ \mathcal{H} \circ \mathcal{F}(W_j). \tag{63}$$

La Figura 16 muestra el funcionamiento de la aproximación usando la Q-iteración, la composición de mapeos se detiene en un vector óptimo aproximado de parámetros W. W idealmente está cerca del punto fijo $W^* = \mathcal{P} \circ \mathcal{T} \circ \mathcal{F}(W^*)$, por tanto, se supone que $W \approx W^*$. En Busoniu et al. (2017) se detallan las condiciones bajo las cuales existe un único punto fijo, y cómo es encontrado asintóticamente si las iteraciones $j \to \infty$.

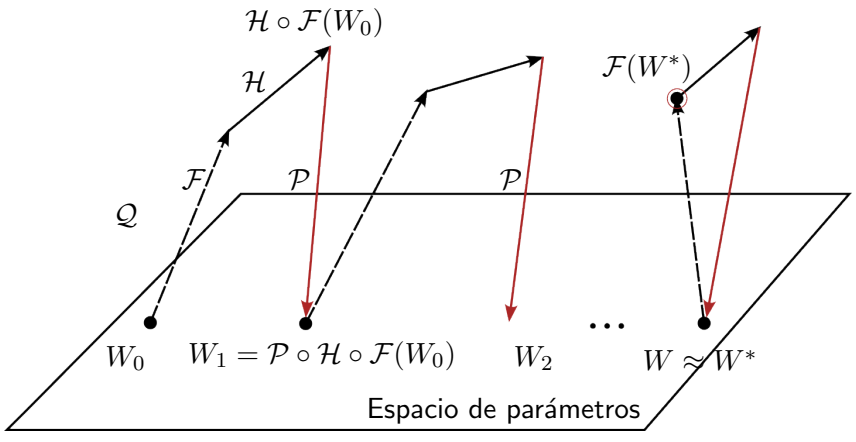

Figura 16. Ilustración conceptual de la aproximación usando la iteración de la Q-función. Idealmente, el algoritmo de RL usado converge a un punto fijo W^* a través de la composición de mapeos descrita por (63). Obsérvese que el resultado del mapeo $\mathcal{H}$ es proyectado de regreso al espacio de parámetros usando el mapeo $\mathcal{P}$ indicado con las flechas rojas.

Una elección común para describir $\mathcal{P}$ es usando la regresión de mínimos cuadrados (Busoniu et al., 2017), obteniendo así la estimación de parámetros $W^\dagger = \mathcal{P}(Q^h)$, considerando:

$$W^\dagger \in \arg\min_{W^\dagger} \sum_{k=1}^{n_{ls}} \left(Q^h(x_k, u_k) - [F(W)](x_k, u_k) \right)^2, \tag{64}$$

donde n_{ls} como el tamaño del lote que indica la cantidad de datos muestreados x_k y u_k (Franklin et al., 1998) usados para formar las funciones base requeridas en la estimación.

2.3.5. Estimación de parámetros

Existe una amplia literatura sobre estimación de parámetros, muchos de los métodos que se estudian tienen aplicación en identificación de sistemas ((Ljung, 1999) y (Franklin et al., 1998)), control adaptativo (Rubio & Sánchez, 1996) y control predictivo basado en modelo ((Grzedzinski & Trodden, 2018) y (Marafioti et al., 2010)). En esta sección se describen de manera general las herramientas usadas en esta tesis referente a la identificación de sistemas y estimación de parámetros, aplicando mínimos cuadrados, mínimos cuadrados recursivos y descenso del gradiente.

2.3.5.1. El modelo

Consideraremos como modelo la siguiente ecuación lineal en diferencias tomada de Ljung (1999):

$$y(k) + a_1 y(k-1) + \cdots + a_n y(k-n) = b_1 u(k-1) + \cdots + b_m u(k-m), \tag{65}$$

donde y representa la salida del sistema, u representa la entrada del sistema, n y m son enteros positivos, a_i y b_j son coeficientes reales.

Definiendo los siguientes vectores:

$$\theta = \Big[a_1, \ldots, a_n, b_1, \ldots, b_m\Big]^T, \tag{66}$$

$$\phi(k) = \Big[-y(k-1), \ldots, -y(k-n), u(k-1), \ldots, u(k-m)\Big]^T, \tag{67}$$

la ecuación (65) se puede escribir como:

$$y(k) = \phi^T(k)\theta. \tag{68}$$

La salida ahora depende de los datos pasados en $\phi(k)$, conocido como vector de regresión y del vector de parámetros θ.

La ecuación (68) resulta en una estructura funcional que puede ser extentida fácilmente como un modelo lineal de regresión simple ((Muller & Guido, 2017) y (Raschka, 2016)):

$$y(k) = \phi^T(k)\theta + e(k,\theta) \tag{69}$$

2.3.5.2. Mínimos cuadrados ordinarios

Los mínimos cuadrados ordinarios o solo mínimos cuadrados (LS, por sus siglas en inglés), son un método no recursivo de estimación de parámetros basado en lotes. Se considera que el problema consiste en encontrar los valores de los parámetros de θ, que corresponden a los coeficientes dados en (66). Al ser encontrados los parámetros podemos calcular la salida $\hat{y}(k,\theta)$, conocida como salida estimada. La salida medida es descrita como $y(k)$.

El método de mínimos cuadrados consiste minimizar una función de costo $\mathcal{J}(\theta)$ que describe el cuadrado del error. Dicho error refleja la diferencia entre la salida medida y la salida estimada (Rubio & Sánchez, 1996):

$$\mathcal{J}(\theta) = \sum_{k=n}^{n_{ls}} e^2(k,\theta) = \sum_{k=n}^{n_{ls}} (y(k) - \phi^T(k)\theta)^2. \tag{70}$$

Definiendo:

$$Y(k) = \begin{bmatrix} y(n) \\ \vdots \\ y(k) \end{bmatrix}, \qquad \Phi(k) = \begin{bmatrix} \phi^T(n) \\ \vdots \\ \phi^T(k) \end{bmatrix}^T, \qquad E(k,\theta) = \begin{bmatrix} e(n,\theta) \\ \vdots \\ e(k,\theta) \end{bmatrix}.$$

La función de costo en su forma matricial, es representada como:

$$\mathcal{J}(\theta) = \sum_{k=n}^{n_{ls}} e^2(k,\theta) = E^T E, \tag{71}$$

$$\mathcal{J}(\theta) = (Y^T - \theta^T\Phi)(Y - \Phi^T\theta), \tag{72}$$

donde $Y = Y(n_{ls})$, $\Phi = \Phi(n_{ls})$ y $E = E(n_{ls}, \theta)$.

Desarrollando (72) y derivando la función de costo $\mathcal{J}$ con respecto a θ, obtenemos:

$$\frac{\partial \mathcal{J}}{\partial \theta} = 2\theta^T \Phi \Phi^T - 2Y^T \Phi^T. \tag{73}$$

El valor óptimo de θ satisface $\frac{\partial \mathcal{J}}{\partial \theta} = 0$, de donde:

$$\theta^\dagger = (\Phi \Phi^T)^{-1} \Phi Y, \tag{74}$$

la cual es conocida como ecuación normal para la estimación de los parámetros de θ (Ljung, 1999).

2.3.5.3. Mínimos cuadrados recursivos

Los métodos recursivos aprovechan parte de los cálculos realizados en un paso para definir el siguiente, agregando una corrección. Por lo tanto, el cálculo de los parámetros en un instante de tiempo se realiza como:

$$\theta(k+1) = \theta(k) + \textit{corrección}. \tag{75}$$

Para el desarrollo de los mínimos cuadrados recursivos (RLS, por sus siglas en inglés), se usa una solución conocida, definida por los siguientes pasos de acuerdo a Franklin et al. (1998). Se introduce la notación de la matriz:

$$\Gamma(k) = (\Phi^T(k) \Lambda \Phi(k))^{-1}, \tag{76}$$

donde Λ es una matriz de ponderación.

1. Seleccionar los valores de λ y la longitud de N, donde λ es un factor de olvido, $0 < \lambda < 1$ y $a = 1 - \lambda$.

2. Coleccionar N datos de entradas y salidas para construir el vector de regresión $\Phi(N+1)$. Define $k = N$.

3. Calcula los valores iniciales $\theta(N)$ y $\Gamma(N)$.

4.

$$L(k+1) = \frac{\Gamma(k)}{\lambda}\Phi(k+1)\left(\frac{1}{a} + \Phi^T(k+1)\frac{\Gamma(k)}{\lambda}\Phi(k+1)\right)^{-1}. \tag{77}$$

5. Coleccionar los datos de entrada $u(k+1)$ y salida $y(k+1)$ y hacer:

$$\theta(k+1) = \theta(k) + L(k+1)\left(y(k+1) - \Phi^T(k+1)\theta(k)\right). \tag{78}$$

$$\Gamma(k+1) = \frac{1}{\lambda}\left(\mathbb{I} - L(k+1)\Phi^T(k+1)\right)\Gamma(k). \tag{79}$$

6. Formar $\Phi(k+2)$. Define $k \leftarrow k+1$.

7. Ir al paso 4.

El factor de olvido λ proporciona un descuento exponencial de las mediciones antiguas cuando $0 < \lambda < 1$ (Ljung, 1999), priorizando u olvidando los datos antiguos para sistemas en los que se tienen variaciones.

2.3.5.4. Descenso del gradiente

Una forma de sintonizar los parámetros del vector de pesos θ de un aproximador, es a través del descenso del gradiente (GD, por sus siglas en inglés). Debido a que el GD ofrece un enfoque para la optimización de una función de costo $\mathcal{J}$ cuando se tiene una función convexa o no convexa $f(x_1, \ldots, x_n)$. El GD usa las derivadas parciales de la función de interés $\partial f/\partial x_i$ para encontrar la dirección que minimiza f.

Dicha dirección resulta ser la dirección en la que f decrece más rápido, y es dada por $-\nabla f$. Donde ∇f representa el vector gradiente de n derivadas parciales de f y se define como:

$$\nabla f(x_1, \ldots, x_n) = \left[\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}\right]^T. \tag{80}$$

En el GD la función de interés f en x como elemento de su dominio, se define como una función de costo $\mathcal{J}$ respecto a θ, debido a que es el argumento que se desea sintonizar. Lo que resulta en un algoritmo que actualiza los parámetros de un vector de pesos θ en cada iteración a través de:

$$\theta^{i+1} = \theta^i - \alpha \nabla \mathcal{J}(\theta), \tag{81}$$

donde $\mathcal{J}(\theta)$ es definida por (70). Además, α es un parámetro de sintonización que define el tamaño de paso del algoritmo, conocido como tasa de aprendizaje.

2.3.5.5. Excitación persistente del sistema

Es importante considerar que la señal de entrada sea de excitación persistente (PE, por sus siglas en inglés), es decir, que el sistema contenga la suficiente dinámica que posibilite la correcta identificación de los parámetros de θ. De acuerdo a Marafioti et al. (2010) tenemos la siguiente definición:

Definición 3 *Una señal de entrada escalar u es fuertemente persistentemente excitada de orden n; si para toda k existe un entero $N > 0$ y algún $\alpha > 0$ y $\beta > 0$, tales que:*

$$\alpha \mathbb{I} > \sum_{i=k}^{k+N} \begin{bmatrix} u_{i+n} \\ u_{i+n-1} \\ \vdots \\ u_{i+1} \end{bmatrix} \begin{bmatrix} u_{i+n} \\ u_{i+n-1} \\ \vdots \\ u_{i+1} \end{bmatrix}^T > \beta \mathbb{I}, \tag{82}$$

La notación $A > B$ donde A, B son matrices cuadradas de la misma dimensión significa que $A - B$ es definida positiva.

Considerando una descripción más compacta:

$$\Psi_{k+N} = \sum_{i=k}^{k+N} \begin{bmatrix} u_{i+n} \\ u_{i+n-1} \\ \vdots \\ u_{i+1} \end{bmatrix} \begin{bmatrix} u_{i+n} \\ u_{i+n-1} \\ \vdots \\ u_{i+1} \end{bmatrix}^T, \tag{83}$$

se tiene que $\alpha \mathbb{I} - \Psi_{k+N} > 0$ y simultáneamente $\Psi_{k+N} - \beta \mathbb{I} > 0$. Por tanto, la excitación persistentemente

fuerte se asegura si $\Psi_{k+N} > 0$.

2.4. Iteración de valor e iteración de política para el problema del LQR en tiempo discreto

A continuación se presentan los algoritmos de iteración de valor e iteración de política de acuerdo a Lewis et al. (2012a). Considérese que $u_{k+1} = h_j(x_{k+1})$ es la política de control realimentado para el estado siguiente. Además, una política de control se dice *admisible* si es estabilizante y produce un costo finito para la función de valor (Lewis et al., 2012a).

2.4.1. Iteración de valor con TD

1. **Inicialización**

   Seleccionar cualquier política de control no necesariamente admisible $h_0(x_k)$. Iniciando con $j = 0$, iterar sobre j hasta converger.

2. **Paso de actualización de la función de valor**

$$V^{h_{j+1}}(x_k) = \rho(x_k, h_j(x_k)) + \gamma V^{h_j}(x_{k+1}). \tag{84}$$

3. **Paso de mejoramiento de la política**

$$h_{j+1}(x_k) = \arg\min_{h(.)} \left(\rho(x_k, h(x_k)) + \gamma V^{h_{j+1}}(x_{k+1})\right). \tag{85}$$

   De manera explicita la representación de la ecuación (85), para una dinámica parcialmente conocida se describe como:

$$h_{j+1}(x_k) = -\frac{\gamma}{2} R^{-1} g^T(x_k) \nabla V^{h_{j+1}}(x_{k+1}). \tag{86}$$

2.4.2. Iteración de política con TD

1. **Inicialización**

   Seleccionar un política de control admisible $h_0(x_k)$. Iniciando con $j = 0$, iterar sobre j hasta converger.

2. **Paso de evaluación de la política**

$$V^{h_{j+1}}(x_k) = \rho(x_k, h_j(x_k)) + \gamma V^{h_{j+1}}(x_{k+1}). \tag{87}$$

3. **Paso de mejoramiento de la política**

   Usar ecuación (86).

2.4.3. Iteración de política con Q-learning

1. **Inicialización**

   Seleccionar cualquier política de control admisible $h_0(x_k)$. Iniciando con $j = 0$, iterar sobre j hasta converger.

2. **Paso de evaluación de la política**

$$Q^{h_{j+1}}(x_k, u_k) = \rho(x_k, h_j(x_k)) + \gamma Q^{h_{j+1}}(x_{k+1}, u_{k+1}). \tag{88}$$

3. **Paso de mejoramiento de la política**

$$h_{j+1}(x_k) = \underset{h(.)}{\arg\min}\ (\rho(x_k, h(x_k)) + \gamma Q^{h_{j+1}}(x_{k+1}, u_{k+1})). \tag{89}$$

Una representación de la ecuación (89) para una dinámica totalmente desconocida se da en (55).

Capítulo 3. Metodología

En este capítulo se explica la metodología seguida para la implementación de los algoritmos de RL en tiempo discreto, tomando como base los algoritmos descritos en la sección 2.4. Para ello, se considera el caso particular de un sistema masa-resorte-amortiguador, donde la dinámica del sistema es lineal. Además, se describe el uso de aproximadores paramétricos para las funciones de valor (funciones V y Q-funciones) descritas en el capítulo anterior.

Los algoritmos contemplan dos tipos de entrenamiento: episódico usando funciones V, y continuo usando primeramente funciones V y luego Q-funciones. Cuando hay conocimiento parcial de la dinámica del sistema para el problema de regulación, se usa el método de aprendizaje por diferencias temporales con el algoritmo de iteración de valor (sección 2.4.1) e iteración de política (sección 2.4.2), con una modificación a la metodología descrita por Lewis et al. (2012a) y Lewis & Vrabie (2009). El objetivo de la modificación es aprender leyes de control óptimas a través de un entrenamiento episódico.

La Figura 17 ilustra la idea de este entrenamiento, donde se aplica una política de control para toda una trayectoria de largo N, se mide x_{k+1}, indicado por los cuadros blancos y se actualiza la función de valor. Como resultado se obtiene un vector de parámetros W^{j+1}. Posteriormente, se usan los datos medidos y la función de valor actualizada para mejorar la política de control a través del entrenamiento de un aproximador llamado red neuronal del actor (RNA). Como resultado se obtiene un vector de ganancias de control K_{j+1}. Este enfoque es ideal para tareas de control donde no se puede realizar un entrenamiento mientras el sistema está en operación, debido a que se pueden comprometer los elementos físicos del sistema. La Figura 18 ilustra la idea del entrenamiento continuo usando funciones V.

Cuando haya desconocimiento completo de la dinámica del sistema, se aborda el problema de regulación haciendo uso del algoritmo Q-learning (sección 2.4.3). La forma de implementar este algoritmo en línea puede ser usada en tiempo real con el sistema mientras está en operación.

3.1. Formulación del modelo en tiempo discreto

Para un sistema lineal de tiempo continuo de la forma:

$$\dot{x} = A_c x + B_c u, \tag{90}$$

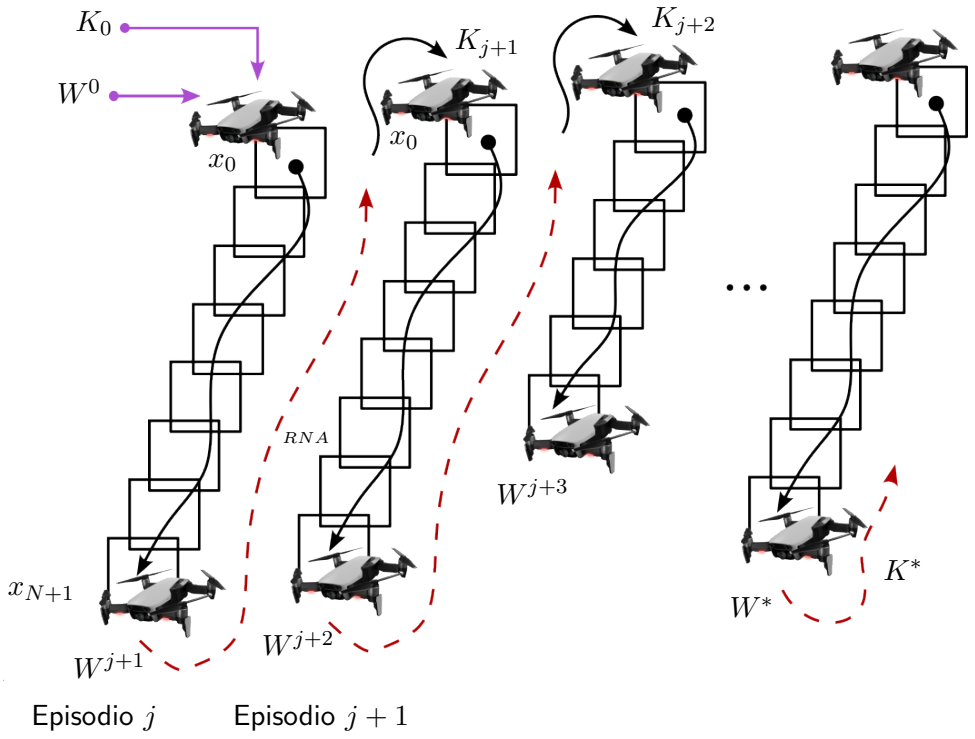

Figura 17. Ilustración conceptual del entrenamiento episódico usando funciones V. Se consideran las mismas condiciones iniciales x_0 para cada episodio, el horizonte de tiempo definido por N para la trayectoria. Este esquema no requiere tener la misma longitud de trayectoria para todos los episodios. La línea cortada roja indica el proceso realizado por la RNA.

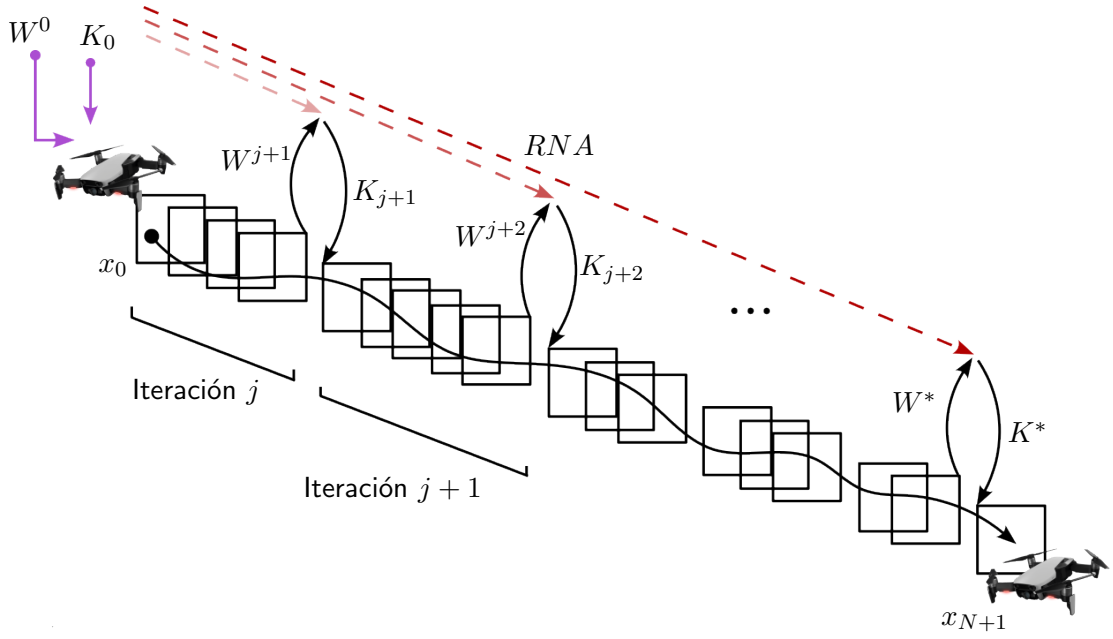

Figura 18. Ilustración conceptual del entrenamiento continuo usando funciones V. Este esquema considera realizar la evaluación de la política (o actualización de valor) y el mejoramiento de la política en un solo episodio, donde las iteraciones indicadas por j pueden tener diferentes cantidades de datos muestreados. Este esquema de entrenamiento puede ser implementado mientras el sistema está en operación. Considérese que la red neuronal del actor requiere todo el largo de la trayectoria usada por el crítico conforme se realizan los mejoramientos.

$$y = C_c x, \tag{91}$$

donde A_c, B_c y C_c son las matrices en el espacio de estados correspondientes a la matriz de dinámica, la matriz de control y la matriz de salida respectivamente.

El sistema se discretiza con un tiempo de muestreo T y un retenedor de orden zero, obteniendo un sistema en tiempo discreto de la forma:

$$x_{k+1} = Ax_k + Bu_k, \tag{92}$$

$$y_k = Cx_k, \tag{93}$$

el cual será utilizado en las simulaciones numéricas.

3.1.1. Sistemas de estudio

Con el fin de ilustrar el desempeño de los algoritmos al emplearse tanto en un sistema estable como en uno inestable en lazo abierto, se usarán como ejemplo del primero un sistema masa-resorte-amortiguador y para el segundo un sistema tomado de Kiumarsi et al. (2014), los cuales se presentan a continuación como Caso 1 y Caso 2, respectivamente.

En ambos casos se considera un índice de desempeño dado por (27) usando las matrices:

$$Q = \begin{bmatrix} q_i & 0 \\ 0 & q_i \end{bmatrix}, \qquad R = r_i. \tag{94}$$

Los valores q_i y r_i se describen en cada ejemplo del Capítulo 4, y se seleccionan de tal forma que ponderan el esfuerzo de control por encima de los estados para un problema de mínima energía.

3.1.1.1. Caso 1. Sistema lineal estable de segundo orden

Primero consideremos el modelo de un sistema mecánico de segundo orden masa-resorte-amortiguador de un grado de libertad, como el que se muestra en la Figura 19, donde la ecuación que describe su movimiento está dada por:

$$m\ddot{x} + b\dot{x} + kx = u, \tag{95}$$

donde $u = u(t)$ es la fuerza externa de entrada del sistema (control) y $x = x(t)$ es la posición del desplazamiento de la masa como la salida, b es el coeficiente de amortiguamiento, k es el coeficiente de elasticidad y m es la masa.

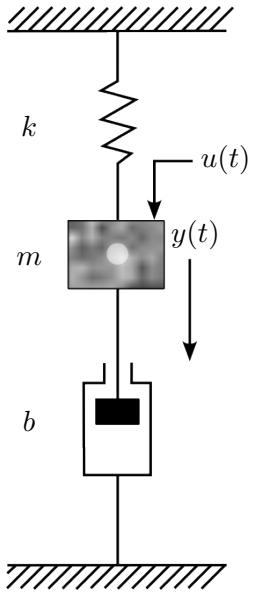

Figura 19. Sistema mecánico masa-resorte-amortiguador.

Definiendo las variables de estado $x_1 = x$ y $x_2 = \dot{x}$, la representación de (95) en espacio de estados bajo la forma canónica controlable es:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u, \tag{96}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \tag{97}$$

Tabla 2. Parámetros del sistema masa-resorte-amortiguador considerado para las pruebas de los algoritmos.

Símbolo	Descripción	Valor
m	Masa	$1\ kg$
b	Coeficiente de amortiguamiento	$1\ N \cdot s/m$
k	Coeficiente de elasticidad	$0.72\ N/m$

Sustituyendo con los valores mostrados en la Tabla 2:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -0.72 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u, \tag{98}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \tag{99}$$

discretizado con $T = 0.1\ seg$ corresponde a:

$$x_{k+1} = \begin{bmatrix} 1.8980 & -0.9048 \\ 1 & 0 \end{bmatrix} x_k + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u_k, \tag{100}$$

$$y_k = \begin{bmatrix} 0.0048 & 0.0047 \end{bmatrix} x_k. \tag{101}$$

Siendo (100) - (101) siendo un sistema con comportamiento estable, donde sus valores propios correspondientes a los polos en lazo abierto son $Z_{1,2} = 0.94900 \pm 0.06480i$.

3.1.1.2. Caso 2. Sistema lineal inestable de segundo orden

El sistema inestable seleccionado de entre los ejemplos dados en Kiumarsi et al. (2014), es el siguiente:

$$x_{k+1} = \begin{bmatrix} -1 & 2 \\ 2.2 & 1.7 \end{bmatrix} x_k + \begin{bmatrix} 2 \\ 1.6 \end{bmatrix} u_k, \tag{102}$$

$$y_k = \begin{bmatrix} 1 & 2 \end{bmatrix} x_k. \tag{103}$$

Los polos en lazo abierto del sistema descrito por (102) - (103) son $Z_1 = -2.1445$ y $Z_2 = 2.8445$.

3.2. Control óptimo adaptativo para un sistema con dinámica parcialmente conocida

El objetivo de usar el método de diferencias temporales con funciones de valor V es diseñar leyes de control óptimas cuando se tiene una dinámica parcialmente conocida. Por dinámica parcialmente conocida nos referimos a no tener conocimiento de la función $f(x)$ en la ecuación (26), que en el caso del LQR corresponde a la matriz de dinámica A.

3.2.1. Aproximación de la función de valor usando conjunto base polinomial para el crítico

Para esta tesis y los ejemplos realizados se considera el teorema de aproximación de Weierstrass mencionado por Lewis et al. (2012a). La idea del teorema radica en que cualquier función real continua, definida en un intervalo cerrado y acotado, es el límite uniforme de una sucesión de polinomios de acuerdo a Murillo (1997).

Teorema 1 (Aproximación de Weierstrass). *Si $f(x)$ es una función continua en un intervalo $[a, b]$ y $\epsilon > 0$, existe un polinomio $\mathcal{P}(x)$ tal que, para todo $x \in [a, b]$*

$$\|f(x) - \mathcal{P}(x)\| < \epsilon.$$

Este teorema tiene un gran uso debido a que los polinomios son generalmente fáciles de manejar. Sutton & Barto (2018) dan un estudio a la aplicación de los polinomios como funciones para métodos lineales de aproximación en RL, donde también se detalla la construcción de otro tipo de funciones (e.g., funciones base radial y funciones base Fourier) y se comparan los desempeños.

Por lo anterior, se considera la descripción del vector de funciones base definido en la sección 2.3.4.1. Sin embargo, para el uso de funciones V solo es necesario el estado x, por lo que la definición del vector de funciones es un mapeo $\Phi : \mathbb{R}^n \to \mathbb{R}^l$.

$$\Phi(x) = [\phi_1(x), \phi_2(x), \ldots, \phi_l(x)]^T, \tag{104}$$

El vector descrito por la ecuación (104) es un conjunto base polinomial, es decir, $\phi_i(x)$ corresponde a

un polinomio como término en los componentes del estado (Vrabie et al., 2013). Los pesos del vector $W \in \mathbb{R}^l$ consisten en los elementos independientes de la matriz P. Debido a que P es simétrica y tiene únicamente $l = n(n+1)/2$ elementos independientes, también hay l elementos en el vector $\Phi(x)$. Sea $x_k = [x_{1_k} \quad x_{2_k}]^T$, por simplicidad usaremos $x_k = [x_1 \quad x_2]^T$.

Para el sistema estable descrito en la sección 3.1.1.1, la representación de la función de valor $V(x_k)$ de acuerdo a la ecuación (42) es:

$$V(x_k) = x_k^T P x_k = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},$$

$$= p_{11}x_1^2 + 2x_1 p_{12} x_2 + p_{22}x_2^2 = \begin{bmatrix} p_{11} & p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} x_1^2 \\ 2x_1x_2 \\ x_2^2 \end{bmatrix}. \tag{105}$$

De lo anterior se define $\Phi(x_k)$ como:

$$\Phi(x_k) = \begin{bmatrix} x_1^2 \\ x_1x_2 \\ x_2^2 \end{bmatrix}, \tag{106}$$

lo que permite definir la función de valor $V(x)$ como:

$$V(x_k) = \begin{bmatrix} p_{11} & 2p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} x_1^2 \\ x_1x_2 \\ x_2^2 \end{bmatrix}. \tag{107}$$

Las descripciones dadas por (105) y (107) para la función de valor $V(x)$ son válidos. Se usa usa la descripción (105) para los algoritmos con dinámica parcialmente conocida y la descripción (107) para los algoritmos con dinámica totalmente desconocida.

Considerando la ecuación (105), y la definición de la función de valor $V(x)$ como un aproximador paramétrico $W^T\Phi(x)$, se obtiene:

$$V(x_k) = W^T \Phi(x_k) = \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} x_1^2 \\ 2x_1x_2 \\ x_2^2 \end{bmatrix}. \tag{108}$$

3.2.2. Iteración de valor usando aprendizaje por diferencias temporales en línea

Se considera una clase de sistemas no lineales descritos por la ecuación (26). Esta forma es usada como se indica en Lewis et al. (2012a) por la conveniencia en su análisis (ver sección 2.2.3 de este documento). Una política de control es definida como una función que va desde el espacio de estados al espacio de control $h : \mathbb{R}^n \to \mathbb{R}^m$, lo que significa que para cada vector de estados x_k la política define una entrada de control $u_k = h(x_k)$.

Usando la VFA, se puede definir la siguiente ecuación de Bellman:

$$W^T \Phi(k) = \rho(x_k, h(x_k)) + \gamma W^T \Phi(k+1). \tag{109}$$

Sustituyendo la ecuación de Bellman (109) en la actualización de valor y mejoramiento de la política del algoritmo de iteración de valor de la sección 2.4.1, se obtiene el Algoritmo 3. Las referencias que se toman como base para la implementación del Algoritmo 3 de esta sección son: Lewis et al. (2012a) y Sanusi et al. (2020). Se realizan adaptaciones al crítico tomadas de lo implementado por Sanusi et al. (2020) en su algoritmo de iteración de política para seguimiento de trayectoria.

En las referencias mencionadas se aborda el diseño del controlador en línea en tiempo real. El enfoque del Algoritmo 3 se basa en realizar el entrenamiento de manera episódica en línea con el sistema, pero no es aplicable cuando el sistema está en operación.

Se desarrolla como alternativa para evitar comprometer los elementos físicos que componen el sistema durante el proceso de entrenamiento, debido a que existe una señal entrada con excitación persistente y pueden aparecer variaciones en la planta que generen una política de control que en lazo cerrado produzca un comportamiento inestable.

Algoritmo 3: Iteración de valor - usando la VFA con diferencias temporales para el problema del LQR

Requeridos: Condición inicial x_0, política de control no necesariamente admisible $h_0(x)$, matrices de ponderación Q, R, $j = 0$, longitud de la trayectoria N, el tamaño del lote n_{ls} de LS, número de episodios a realizar n_{ep}, valor de δ, factor de olvido λ, tasa de aprendizaje β, factor de descuento γ;
mientras $j < n_{ep}$ **hacer**
  % Actualización de valor;
  para $k = 0$ **a** N **hacer**
    Calcular la acción de control u_k con el estado x_k y una excitación persistente ε como $u_k = h_j(x_k) + \varepsilon$;
    $\ell = k + 1$;
    Aplicar u_k al sistema y medir x_{k+1};
    si $k = n_{ls}$ **entonces**
      Determinar la solución con LS para el cálculo de W^0 de RLS;

$$W^0 = \Gamma(n_{ls})\Phi(n_{ls})Y(n_{ls})$$

    fin
    si $k > n_{ls}$ **entonces**
      Determinar la solución con RLS para;

$$\left(W^{j+1}\right)^T \Phi(k) = x_k^T Q x_k + u_k^T R u_k + \gamma \left(W^j\right)^T \Phi(k+1) \tag{110}$$

      si $e_{TD} \leq \delta$ **entonces**
        Ir al **Mejoramiento de la política**;
      fin
    fin
  fin
  % Mejoramiento de la política;
  para $i = 0$ **a** ℓ **hacer**
    Actualizar los parámetros de la política usando GD, mediante la ecuación tomada de Lewis et al. (2012a);

$$U_{j+1}^{i+1} = U_j^i - \beta\sigma(x_i)\left(2R(U_j^i)^T\sigma(x_i) + \gamma g^T(x_i)\nabla\Phi^T(i+1)W^{j+1}\right)^T \tag{111}$$

  fin
  $h_{j+1}(x_\ell) = \left(U_{j+1}^{\ell+1}\right)^T \sigma(x_\ell)$;
  $j \leftarrow j + 1$;
fin

En el lado derecho de la ecuación (110) aparece W^j, que corresponde al vector de pesos del valor estimado para el estado siguiente. Para desarrollar el Algoritmo 3 en línea usando RLS, se parte de la ecuación del error en TD en tiempo k, dicha ecuación necesita de los vectores de pesos iniciales W^0 y

W^1 para resolverse en la primera iteración:

$$e_{TD} = -\left(W^{j+1}\right)^T \Phi(k) + \rho(x_k, h_j(x_k)) + \gamma \left(W^j\right)^T \Phi(k+1), \tag{112}$$

además, se necesita una matriz $\Gamma(n_{ls})$ inicial para operar el algoritmo de RLS, por lo que es necesario sintonizar al inicio del algoritmo los parámetros de los vectores y matriz mencionados, lo cual se hace con un bloque de LS. En la Figura 25 se ilustra esta adaptación.

El uso de RLS permite la solución en línea de (110), además, RLS evita los problemas de pérdida de rango en la matriz de covarianza que se presenta con LS por lotes en trayectorias con tiempo N demasiado grande ((Franklin et al., 1998) y (Strang, 2019)).

El uso de LS por lotes previo a iniciar la estimación con RLS en el crítico se justifica como alternativa a lo siguiente:

1. Para mejorar la convergencia y evitar dar valores de manera arbitraria al vector de pesos W^0 de RLS.
2. Para el cálculo de la matriz $\Gamma(n_{ls})$ (sección 2.3.5.3) se recomienda una matriz de dimensiones adecuadas con valores en la diagonal principal, que generalmente es la identidad multiplicada por un escalar (Franklin et al., 1998). Con el objetivo de evitar dar valores de manera arbitraria a $\Gamma(n_{ls})$, se hace uso del bloque de LS, donde se aprovechan los datos del vector de regresión en el lote inicial de tamaño n_{ls} para el cálculo de $\Gamma(n_{ls})$ inicial.
3. Considere que todavía se necesitan los vectores de pesos W^0 de LS y un vector de pesos W^1 de RLS. Se pasa de necesitar la sintonización de los parámetros iniciales de dos vectores y una matriz, a solo necesitar los parámetros de dos vectores. Igualamos los valores de los pesos de W^0 de LS y W^1 de RLS con los valores de los elementos independientes de P_0. Además, se considera un tamaño de lote inicial $n_{ls} \geq 2l$ de acuerdo a Franklin et al. (1998).

En este punto se determina la solución a la ecuación (110) en línea con RLS para obtener los pesos del vector W^{j+1}, siguiendo los pasos mostrados en la sección 2.3.5.3.

Por lo anterior, se considera mantener el vector de regresión persistentemente excitado, es decir, mantener nuestra trayectoria de estados excitada usando técnicas de identificación de sistemas ((Ljung, 1999) y (Franklin et al., 1998)). Esta tesis usa dos señales distintas para mantener la PE:

- una señal aperiódica generada con la función chirp descrita en Franklin et al. (1998), pero usando una frecuencia final muy por encima de la frecuencia de Nyquist-Shannon lo que genera características frecuenciales que dieron buenos resultados en la práctica,
- una señal *multisenoidal* $f(k)$, que consiste en una combinación de senos y cosenos de diferentes frecuencias (Tangirala, 2018), con una ganancia con *decaimiento exponencial* $Ce^{-\xi k}$ como lo indica Vamvoudakis et al. (2014), donde c es una constante de amplificación y ξ es una tasa de cambio llamada constante de decaimiento. El número de frecuencias angulares depende del problema (Vamvoudakis et al., 2014).

La Figura 20 muestra las señales usadas como señales de PE durante el entrenamiento de algunos ejemplos del Capítulo 4. Todas estas señales fueron generadas con la función chirp reportada en el Anexo A, usando una tasa de muestreo de $T = 0.1\ seg$. Las señales 2 y 3 tienen las mismas frecuencias angulares ω_i y ω_f, siendo diferentes en la ventana de tiempo N. Como resultado las señales tienen distintos comportamientos que afectan la identificación de parámetros. Chirp 1: $\omega_i = 1\ rad/seg$, $\omega_f = 10\ rad/seg$, $N = 200$. Señal 2: $\omega_i = 0.005\ rad/seg$, $\omega_f = 345,000\ rad/seg$, $N = 200$. Señal 3: $\omega_i = 0.005\ rad/seg$, $\omega_f = 345,000\ rad/seg$, $N = 400$.

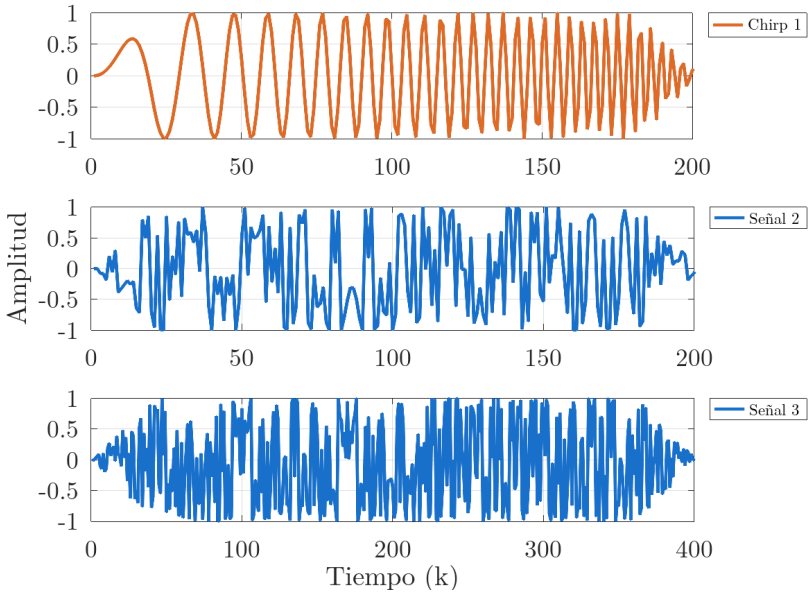

Figura 20. Señales utilizadas como PE.

La Figura 21 muestra dos señales multisenoidales con decaimiento exponencial usadas como ejemplo de una señal de PE. Al usar un C con valores de 0.1 y 0.198 para alguna señal $e^{-\xi k}f(k)$ se evita

inestabilizar el sistema y se obtienen buenos resultados en las simulaciones. Cada señal $f(k)$ es diferente dependiendo del escenario en el que se implementa el algoritmo. Se describe la $f(k)$ en cada ejemplo del Capítulo 4 donde sea usada. La Figura 22 muestra el comportamiento natural del sistema masa-resorte-amortiguador. La Figura 23 muestra como ejemplo el estado x_1 afectado por las señales de PE elegidas para esta tesis.

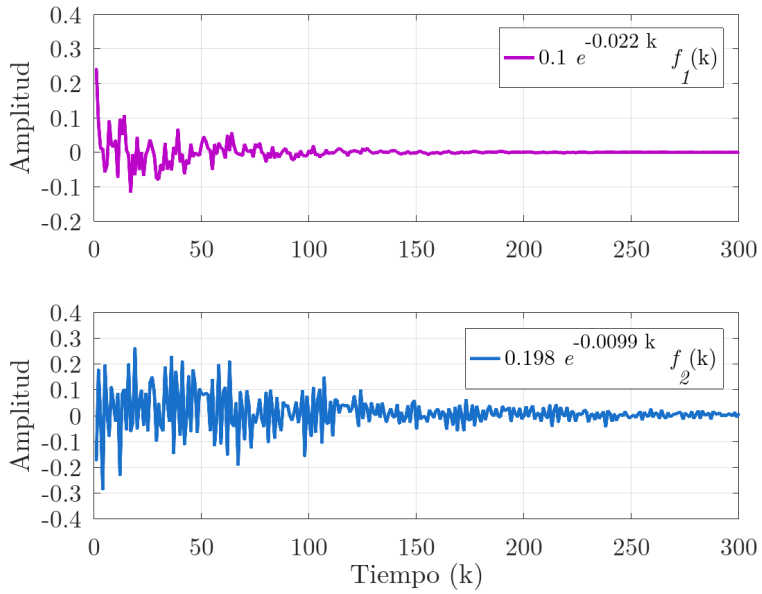

Figura 21. Señales multisenoidales con decaimiento exponencial probadas en la metodología para entrenamiento continuo.

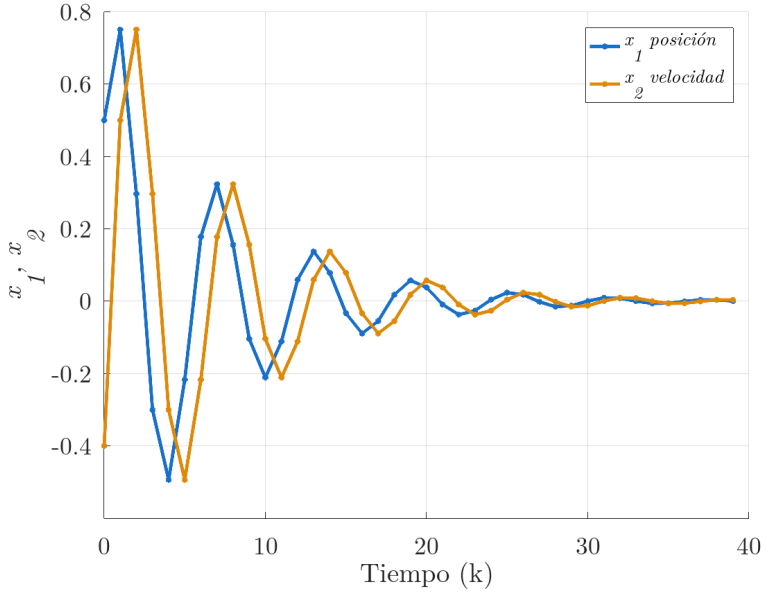

Figura 22. Comportamiento natural del sistema masa-resorte-amortiguador usando un vector de ganancias de entrada $K = [1, -0.3]$ para unas condiciones iniciales de $x_1 = 0.5$ y $x_2 = -0.4$.

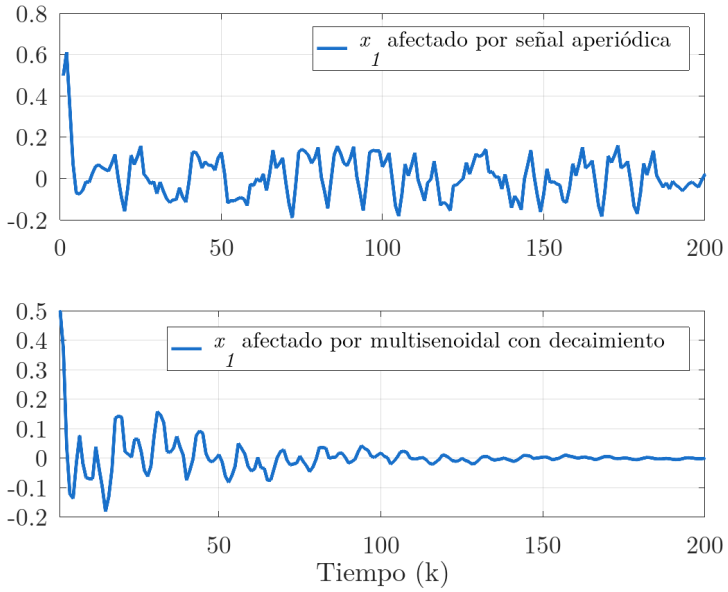

Figura 23. Estado x_1 del sistema afectado por la entrada de control con PE. De una señal con $\omega_i = 0.005\ rad/seg$ y $\omega_f = 345,000\ rad/seg$ con una baja amplitud controlada por la constante de amplificación $c_r = 0.09$ (gráfica superior) y señal multisenoidal con decaimiento exponencial $0.1e^{-0.022k}$ (gráfica inferior). Se usa un vector de ganancias de entrada de control $K = [1, -0.5]$, para una condición inicial de $x_1 = 0.5$.

A partir de la línea 20 del Algoritmo 3, la política es mejorada usando un segundo aproximador paramétrico, llamado red neuronal del actor, cuyo objetivo será el cálculo del control realimentado,

$$u_k = h(x_k) = U^T \sigma(x_k). \tag{113}$$

El uso de este aproximador se justifica debido a que la ecuación (86) resulta en un escalar que no es fácil de implementar, ya que la política de control h_{j+1} se encuentra de manera implícita y además, depende de x_{k+1} que es argumento de una función de activación como se menciona en Lewis et al. (2012a).

En la ecuación (113) se tiene a $\sigma : \mathbb{R}^n \to \mathbb{R}^n$ como un vector de n funciones de activación. Se hace uso de los estados visitados en el paso de la actualización de valor durante la sintonización del crítico, estados que harán el trabajo de funciones de activación para la red neuronal del actor, que en este punto cuentan con una excitación persistente.

Contamos con $U \in R^{n \times m}$ como una matriz de pesos desconocidos que se busca sintonizar. Se da solución a la red neuronal del actor desarrollando la ecuación (111), que corresponde al descenso del gradiente tal y como se indica en Lewis et al. (2012a) y Lewis & Vrabie (2009). En la parte final de la sintonización se tiene una matriz U_{j+1}^{i+1} cuyo número de parámetros y sus valores corresponden a la

actualización del vector de ganancias de control realimentado K_{j+1}:

$$U_{j+1}^T \sigma(x) = -K_{j+1} x, \tag{114}$$

donde i representa las iteraciones realizadas por el actor en la iteración j del descenso del gradiente de acuerdo a (111), es decir, se toma la iteración $i+1$ realizada para U_j como U_{j+1}^{i+1}. La función de activación se tomó como la identidad $\sigma(x) = x$. La sintonización del actor requiere en cada instante k de los datos de la trayectoria x_k y x_{k+1} como se aprecia en la Figura 26, es decir, el estado actual y estado siguiente. Se puede observar que la ecuación (113) reproduce una entrada de control u_k en función de x_k y que en esta formulación el estado siguiente x_{k+1} no es necesario, por lo que (113) corresponde a una legítima acción de control realimentado. Por otro lado en la ecuación (111), sí se necesita del estado siguiente x_{k+1}, por lo que el actor a pesar de no requerir de la matriz de dinámica A del sistema, aprende información acerca de ella y se refleja en la sintonización de los pesos de U_{j+1} al usar los datos x_k y x_{k+1}.

A manera de ilustración, la Figura 24 muestra las trayectorias de los estados x_1 y x_2 en cada episodio realizado durante la etapa de entrenamiento episódico con PE. Para una simulación con $n_{ep} = 3$ se observa que en los episodios 2 y 3 la PE se detiene en el índice de tiempo k indicado por la línea discontinua cuando se alcanza la condición del error de TD.

A continuación se describen cuáles son los parámetros iniciales requeridos por los algoritmos 3 y 4.

- *Parámetros generales:* longitud N de la trayectoria, número de episodios que se desea realizar n_{ep}, tamaño de lote de LS n_{ls} y factor de descuento γ.
- *Parámetros del crítico:* factor de olvido λ, que puede tomar valores en el intervalo $[0.97, 0.99]$ (Franklin et al., 1998), la condición de salida $e_{TD} \leq \delta$. El valor de δ se define tomando en consideración si el sistema es estable, inestable y de la longitud N. La Figura 25 da un representación de la sintonización del crítico con sus elementos variantes de entrada y salida para el Algoritmo 3 para un solo episodio.
- *Parámetros del actor:* parámetro de la sintonización del descenso del gradiente β conocido como tasa de aprendizaje. La Figura 26 ilustra la sintonización del actor con sus elementos variantes de entrada y salida.

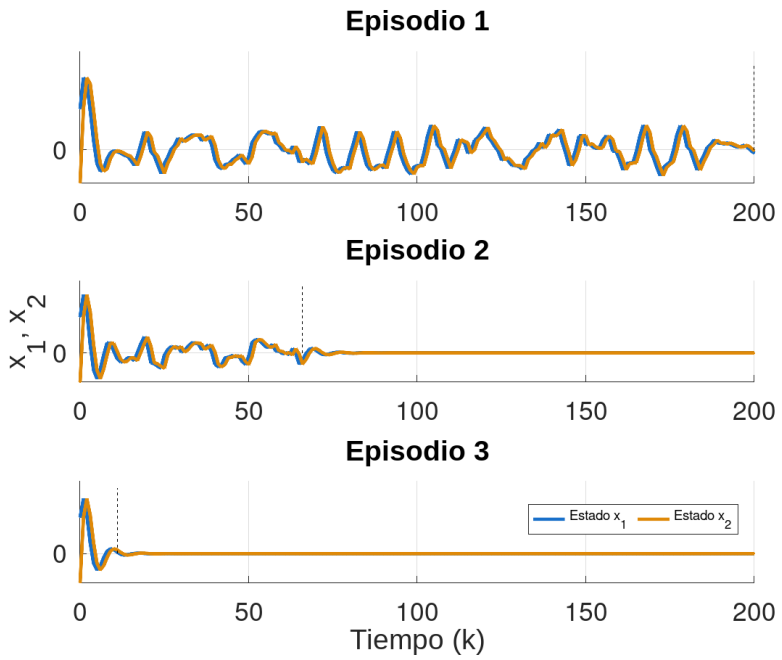

Figura 24. Trayectorias de los estados x_1 y x_2, del sistema estable en entrenamiento episódico, con un $n_{ep} = 3$ usando el Algoritmo 3.

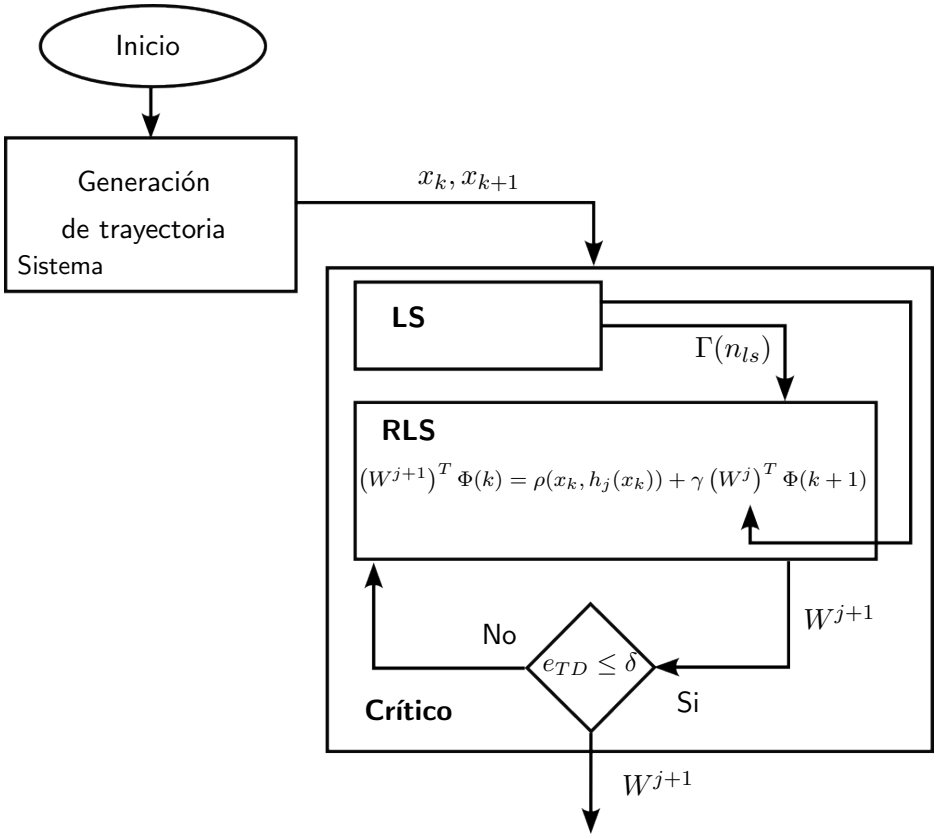

Figura 25. Ilustración del funcionamiento a bloques del crítico para un episodio usando el Algoritmo 3.

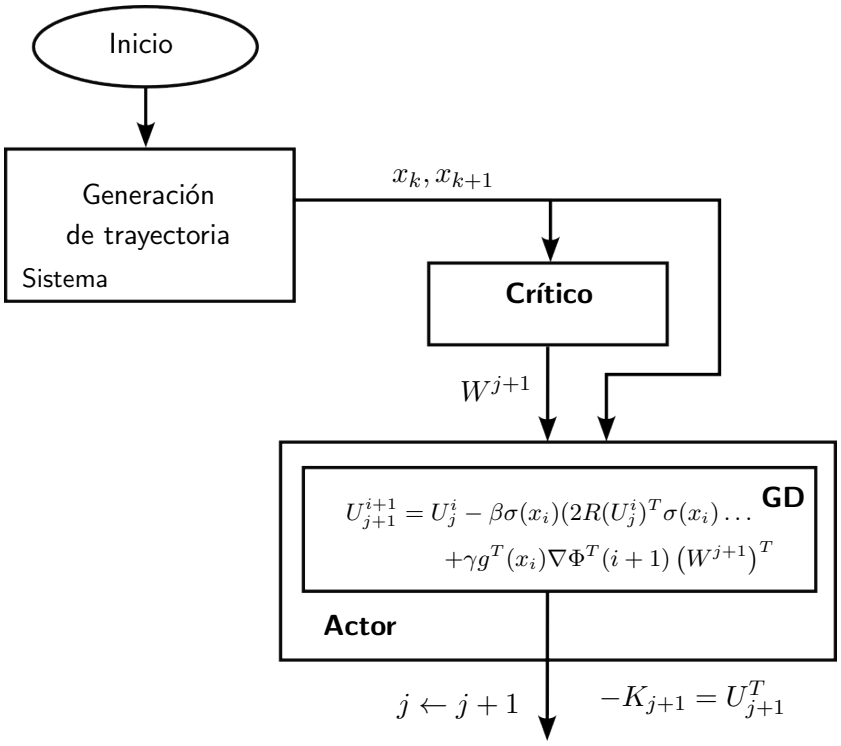

Figura 26. Ilustración del funcionamiento a bloques del actor para un episodio usando el Algoritmo 3.

3.2.3. Iteración de política usando aprendizaje por diferencias temporales en línea

En la implementación del algoritmo de iteración de política con TD en línea, se considera casi todo lo mencionado en el desarrollo del Algoritmo 3, aplicando algunos cambios necesarios para la iteración de política explicados en la sección 2.2.2. En esta sección solo se mencionan los cambios importantes para la correcta implementación del Algoritmo 4. Las referencias que se toman como base para la implementación del Algoritmo 4 de esta sección son: Lewis et al. (2012a) y Sanusi et al. (2020). Se realizan adaptaciones al crítico tomadas de lo implementado por Sanusi et al. (2020) en su algoritmo de iteración de política para seguimiento de trayectoria.

Sustituyendo la ecuación de Bellman (109) en el paso de evaluación de la política del algoritmo de iteración de política de la sección 2.4.2, se obtiene:

$$\left(W^{j+1}\right)^T \Phi(k) = \rho(x_k, h_j(x_k)) + \gamma \left(W^{j+1}\right)^T \Phi(k+1). \tag{115}$$

Se tiene la misma función como término en ambos miembros de la ecuación (115), y esta función solo

depende de los argumentos x_k y x_{k+1}, por lo tanto, se reescribe (115) como:

$$\left(W^{j+1}\right)^T \left(\Phi(k) - \gamma\Phi(k+1)\right) = \rho(x_k, h_j(x_k)). \tag{116}$$

Sea $\Phi'(k) \equiv (\Phi(k) - \gamma\Phi(k+1))$, se define el error de TD para el Algoritmo 4 como:

$$e_{TD} = -\left(W^{j+1}\right)^T \Phi'(k) + \rho(x_k, h_j(x_k)). \tag{117}$$

El uso de LS por lotes previo a iniciar la estimación con RLS en el crítico se justifica como alternativa a lo siguiente:

1. Para hacer uso de RLS es necesario resolver al menos l ecuaciones antes de considerar el paso de mejoramiento de la política. Basándose en la ecuación del error en TD (117), solo se necesita del vector de pesos W^1 para la solución con RLS. Al agregar el bloque de LS se evita dar valores de manera arbitraria al vector de pesos W^1 de RLS, obteniendo mejor convergencia y menos parámetros iniciales por sintonizar. Por la naturaleza de la iteración de política, no es necesario dar W^0 o W^1 al bloque de LS. Se considera un tamaño de lote inicial $n_{ls} \geq 2l$ de acuerdo a Franklin et al. (1998).
2. Considerar el punto 2 del Algoritmo 3, cuando se use LS+RLS en la sintonización del crítico.

En el paso de evaluación de política del algoritmo de iteración de política con TD se busca la evaluación de $h_j(x)$, para ello es necesario resolver la ecuación de Bellman, que cumple con ser una ecuación de Lyapunov lineal de acuerdo a lo mencionado por Vrabie et al. (2013) y Lewis et al. (2012a) (cf. sección 2.2.3.2). Se tiene una solución a esta ecuación para cada estado visitado dentro de la trayectoria de longitud N en el episodio j. Para este episodio j la política de control es fijada en $u_k = h_j(x_k)$, luego x_{k+1} es medido y un paso de RLS es realizado.

Este proceso se repite hasta la convergencia de los parámetros del vector de pesos W^{j+1}. Para que la solución con RLS converja, el vector de regresión $\Phi'(k)$ debe ser persistentemente excitado de la misma forma que en el Algoritmo 3. Después de la convergencia de los parámetros del vector de pesos W^{j+1}, la política de control se mejora con (119), el procedimiento se repite para $j+1$ hasta la convergencia a la solución de control óptimo.

Algoritmo 4: Iteración de política - usando la VFA con diferencias temporales para el problema del LQR

Requeridos: Condición inicial x_0, una política de control no necesariamente admisible $h_0(x)$, matrices de ponderación Q, R, $j = 0$, longitud de la trayectoria N, el tamaño del lote n_{ls} de LS, el número de episodios a realizar n_{ep}, valor de δ, factor de olvido λ, tasa de aprendizaje β, factor de descuento γ;
mientras $j < n_{ep}$ **hacer**
  % Evaluación de la política ;
  para $k = 0$ **a** N **hacer**
    Calcular la acción de control u_k con el estado x_k y una excitación persistente ε como $u_k = h_j(x_k) + \varepsilon$;
    $\ell = k + 1$;
    Aplicar u_k al sistema y medir x_{k+1};
    si $k = n_{ls}$ **entonces**
      Determinar la solución con LS para el cálculo de W^1 de RLS;

$$W^1 = \Gamma(n_{ls})\Phi^{'}(n_{ls})Y(n_{ls})$$

    fin
    si $k > n_{ls}$ **entonces**
      Determinar la solución con RLS para;

$$\left(W^{j+1}\right)^T \Phi^{'}(k) = x_k^T Q x_k + u_k^T R u_k \tag{118}$$

      si $e_{TD} \leq \delta$ **entonces**
        Ir al **Mejoramiento de la política**;
      fin
    fin
  fin
  % Mejoramiento de la política;
  para $i = 0$ **a** ℓ **hacer**
    Actualizar los parámetros de la política usando GD;

$$U_{j+1}^{i+1} = U_j^i - \beta\sigma(x_i)\left(2R(U_j^i)^T\sigma(x_i) + \gamma g^T(x_i)\nabla\Phi^T(i+1)W^{j+1}\right)^T \tag{119}$$

  fin
  $h_{j+1}(x_{\ell+1}) = \left(U_{j+1}^{\ell+1}\right)^T \sigma(x_{\ell+1})$;
  $j \leftarrow j + 1$;
fin

Consideraciones para el Algoritmo 3 y el Algoritmo 4: tomar en cuenta lo indicado por Sanusi et al. (2020), donde se menciona que el enfoque para dinámica parcialmente conocida es limitado a sistemas para los cuales las variaciones solamente ocurren en la matriz desconocida A. Además, solo se usó la

estimación de parámetros con LS+RLS para la sintonización del crítico y GD para la sintonización del actor en ambos algoritmos. Por lo tanto, se deben usar los mismos parámetros que se describen en la sección 3.2.2. Se implementa el entrenamiento continuo solo para la comparación con el entrenamiento episódico.

3.3. Control óptimo para un sistema con dinámica totalmente desconocida

Hasta esta sección de la tesis se han implementado controladores óptimos adaptativos usando RL cuando únicamente se tiene conocimiento de la matriz B de entrada de control del sistema, que corresponde a $g(x)$ en nuestra dinámica no lineal. El método de RL conocido como algoritmo de Q-learning resulta en un algoritmo de control adaptativo que converge en línea a la solución de control óptimo cuando la dinámica del sistema es totalmente desconocida, es decir, no se conocen las funciones $f(x)$ y $g(x)$, que en el caso del LQR corresponden a las matrices A y B.

Las referencias principales que se toman para esta sección son: Kiumarsi et al. (2014), Sanusi et al. (2020) y Görges (2019). Al igual que lo desarrollado en las referencias mencionadas, nuestra metodología se centra en una implementación en línea para el sistema estable descrito en la sección 3.1.1.1. Esta metodología puede ser usada en tiempo real mientras el sistema se encuentra en operación, utilizando un entrenamiento continuo como muestra la Figura 27. En esta sección y particularmente en esta tesis se aborda el algoritmo de Q-learning usando únicamente el algoritmo de iteración de política. Considere que sí se necesita del modelo del sistema para la generación de las trayectorias en la simulación.

3.3.1. Iteración de política para algoritmo de Q-learning en línea

La Q-función puede ser parametrizada de forma similar a la función de valor. Los algoritmos de iteración de valor e iteración de política pueden desarrollarse usando dicha parametrización:

$$Q^h(x,u) = W^T \phi(x,u). \tag{120}$$

El algoritmo Q-learning para control óptimo adaptativo se basa la ecuación (120). Recordando que la

Q-función para el problema del LQR en tiempo discreto es una forma cuadrática en términos de los estados $x_k \in \mathbb{R}^n$ y las acciones de control $u_k \in \mathbb{R}^m$ podemos definir la variable $z_k \in \mathbb{R}^{n+m}$:

$$z_k \equiv \begin{bmatrix} x_k^T & u_k^T \end{bmatrix}^T, \tag{121}$$

la Q-función se puede ver para el caso del LQR como:

$$Q^h(x_k, u_k) = \begin{bmatrix} x_k^T & u_k^T \end{bmatrix} H \begin{bmatrix} x_k^T & u_k^T \end{bmatrix}^T, \tag{122}$$

donde $H \in \mathbb{R}^{(n+m)\times(n+m)}$ está dada por (54) en términos de las matrices A y B del sistema. El número de elementos independientes de H es $s = (n+m)\times(n+m+1)/2$. Sin embargo, la matriz H puede ser estimada en línea sin conocer la dinámica usando técnicas de identificación de sistemas como lo indica Lewis et al. (2012a).

Ahora los pesos del vector $W \in \mathbb{R}^s$ consisten en los parámetros independientes de H y el vector de funciones base $\Phi(z) \in \mathbb{R}^s$ contiene también s elementos que consisten de términos cuadráticos de los elementos de z. Se define a $\Phi(z_k)$ como un vector de regresión que consiste en un conjunto base polinomial cuadrático en tiempo k para el sistema:

$$\Phi(z_k) = \begin{bmatrix} x_1^2 & x_1x_2 & x_1u & x_2^2 & x_2u & u^2 \end{bmatrix}^T, \tag{123}$$

y se define la estructura del aproximador paramétrico como:

$$W^T\Phi(z_k) = \begin{bmatrix} w_1 & 2w_2 & 2w_3 & w_4 & 2w_5 & w_6 \end{bmatrix} \begin{bmatrix} x_1^2 \\ x_1x_2 \\ x_1u \\ x_2^2 \\ x_2u \\ u^2 \end{bmatrix}. \tag{124}$$

En el caso del algoritmo de iteración de política, el vector de regresión $\phi(z_{k+1})$ se implementa de la misma forma que (123), usando el tiempo $k+1$. Usando la descripción de la Q-función (124) se define

la siguiente ecuación de Bellman:

$$W^T\Phi(k) = \rho(x_k, u_k) + \gamma W^T\Phi(k+1). \tag{125}$$

Sustituyendo la ecuación (125) en (88) del paso de evaluación de la política[1] del algoritmo Q-learning, se obtiene:

$$\left(W^{j+1}\right)^T (\Phi(k) - \gamma\Phi(k+1)) = \rho(x_k, h_j(x_k)). \tag{126}$$

Mientras que el paso de mejoramiento de la política (89) del algoritmo Q-learning, será:

$$h_{j+1}(x_k) = \arg\min_u \left(\left(W^{j+1}\right)^T \Phi(x_k, u) \right), \quad \text{para todo} \quad x_k \in \mathbb{R}^n. \tag{127}$$

Observe que el paso del mejoramiento (127) no necesita conocer la dinámica del sistema, ya que solo dependerá de la aproximación de la Q-función.

Para la implementación en línea se usan RLS o GD, ambos algoritmos se unen al bloque inicial de LS para dar solución a la ecuación (126). El objetivo es sintonizar los pesos del vector W^{j+1}, dando como vector de regresión a $\Phi'(k) \equiv (\Phi(k) - \gamma\Phi(k+1))$. En cada instante de tiempo k se mide x_{k+1}, y se forma z_k como se indica en la ecuación (121). Se calcula z_{k+1} usando $u_{k+1} = h_j(x_{k+1})$, siendo $h_j(x)$ la política actual.

El paso de mejoramiento (127) de la política se logra sin conocimiento de la dinámica del sistema debido a que la Q-función contiene u_k como argumento, por lo tanto, $\partial(\left(W^{j+1}\right)^T \Phi(z_k)/\partial u_k$ puede calcularse de manera explícita. Por lo anterior, la estructura de un aproximador paramétrico para el actor no es necesaria en el Algoritmo 5 como se ilustró en la Figura 13. Explícitamente (127) corresponde a la ecuación (129).

El uso de LS por lotes previo a iniciar la estimación con RLS o GD en el crítico se justifica como alternativa a lo siguiente:

1. Es necesario resolver al menos s ecuaciones antes de considerar el paso de mejoramiento de la política. Solo se necesita dar el vector de pesos W^1 para la solución con RLS o GD. Al agregar el bloque de LS evitamos dar valores de manera arbitraria al vector de pesos W^1 antes mencionado,

[1] El paso de evaluación es llamado también paso de *actualización* de la Q-función, por la naturaleza de algoritmo Q-learning y su relación con la iteración de valor (Sutton & Barto, 2018).

obteniendo mejor convergencia y menos parámetros iniciales por sintonizar. Por la naturaleza de la iteración de política, no es necesario dar W^0 o W^1 al bloque de LS. Se considera un tamaño de lote inicial $n_{ls} \geq s$.

2. Considerar para este algoritmo el punto 2 mencionado en el Algoritmo 3 cuando se use RLS.

Parámetros iniciales del Algoritmo 5

- *Parámetros generales:* longitud de la trayectoria N, tamaño de lote de LS n_{ls} y factor de descuento γ.
- *Parámetros del crítico:* condición del error δ de TD, que indica la calidad de la predicción de la Q-función, considerar que la condición depende de si el sistema es estable, inestable o de la longitud N.
  - RLS: factor de olvido λ que puede tomar valores en el intervalo $[0.97, 0.99]$.
  - GD: tasa de aprendizaje α.

3.4. Sumario

Se presentaron las metodologías de los algoritmos de RL implementados durante la tesis, para:

- dinámica parcialmente conocida:
  - iteración de valor con entrenamiento episódico,
  - iteración de política con entrenamiento episódico,
  - considere las *observaciones* descritas al final de la sección 3.2.3 y que el esquema usado es LS+RLS (crítico) con GD (actor).
- Dinámica totalmente desconocida: iteración de política con entrenamiento continuo para los esquemas:
  - LS+GD (crítico).
  - LS+RLS (crítico).

El Algoritmo 5 está considerando solamente el uso de LS+GD para la sintonización del crítico, el uso de LS+RLS es fácilmente implementado siguiendo las consideraciones descritas para el Algoritmo 4. El uso de las Q-funciones permite el diseño de controladores óptimos adaptativos sin que intervenga la dinámica del sistema, pero esto no quiere decir desconocer el sistema en su totalidad. Es fácil notar que al inicio de los algoritmos se requiere conocer algunas características como: orden del sistema y comportamiento del sistema al aplicar la política $h_0(x)$ en lazo cerrado. La Figura 28 muestra el diagrama de bloques del Algoritmo 5.

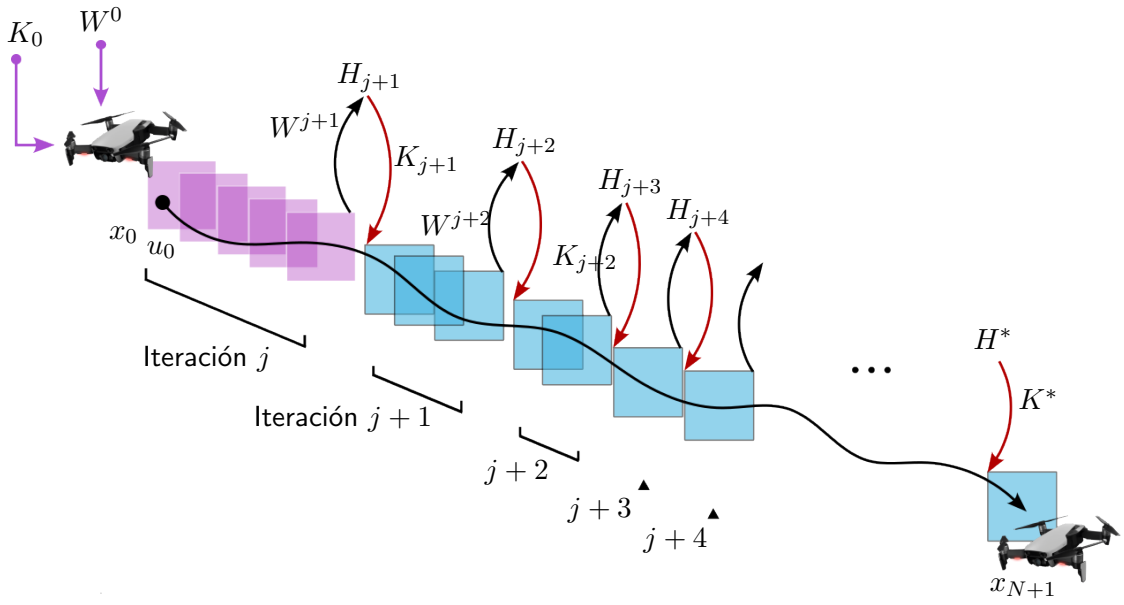

Figura 27. Ilustración conceptual del esquema para entrenamiento continuo usando el algoritmo Q-learning. El color magenta indica el bloque de muestras medidas de la trayectoria de estado y el control usados por LS para sintonizar los parámetros de W^{j+1}. El color azul indica la operación de RLS o GD usando las muestras medidas para la sintonización del crítico en línea.

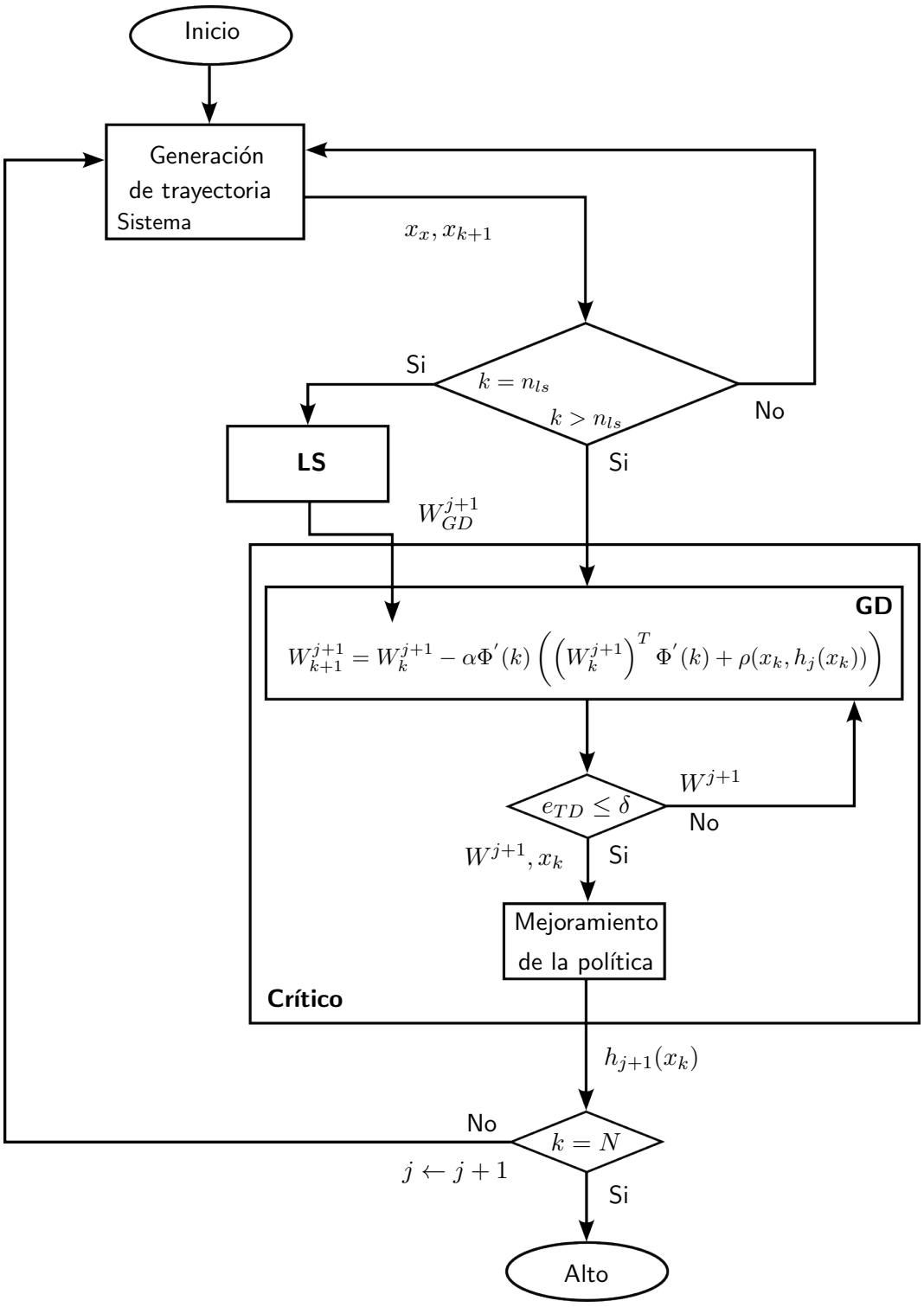

Figura 28. Descripción a bloques del Algoritmo 5, se toma como base la metodología descrita en Vrabie et al. (2013).

Algoritmo 5: Iteración de política - usando la VFA con Q-learning para el problema del LQR

Requeridos: Condición inicial x_0, política de control admisible $h_0(x)$, matrices de ponderación Q, R, se define $j = 0$, longitud de la trayectoria N, el tamaño del lote n_{ls} de LS, valor de δ, tasa de aprendizaje α y un factor de descuento γ ;

para $k = 0$ **a** N **hacer**

  % Evaluación de la política (actualización de la Q-función);

  Calcular la acción de control u_k con el estado x_k y una excitación persistente ε como $u_k = h_j(x_k) + \varepsilon$;

  Aplicar u_k al sistema y medir x_{k+1};

  Calcular $u_{k+1} = h_j(x_{k+1})$;

  si $k = n_{ls}$ **entonces**

    Determinar la solución con LS para el cálculo de $W^1_{n_{ls}}$ de GD;

$$W^1_{n_{ls}} = \Gamma(n_{ls})\Phi^{'}(n_{ls})Y(n_{ls})$$

  fin

  si $k > n_{ls}$ **entonces**

    Determinar la solución con GD;

$$W^{j+1}_{k+1} = W^{j+1}_k - \alpha\Phi^{'}(k)\left(\left(W^{j+1}_k\right)^T\Phi^{'}(k) - x^T_k Q x_k + u^T_k R u_k\right) \tag{128}$$

    si $e_{TD} \leq \delta$ **entonces**

      % Mejoramiento de la política;

$$\begin{aligned} H_{j+1} &\leftarrow W^{j+1}_{k+1} \\ u_k &= -(H_{uu})^{-1}H_{ux}\, x_k \\ &= h_{j+1}(x_k) \end{aligned} \tag{129}$$

      $j \leftarrow j + 1$;

    fin

  fin

fin

Capítulo 4. Simulaciones numéricas y resultados

Este capítulo presenta el funcionamiento de los algoritmos de aprendizaje por refuerzo en tiempo discreto mencionados en el Capítulo 3, a través de simulaciones numéricas. Los resultados se basan en los siguientes escenarios:

- **Dinámica parcialmente conocida:**
  - iteración de valor (Algoritmo 3):
    - sistema estable, sección 4.2.1,
    - sistema inestable, sección 4.2.2.
  - Iteración de política (Algoritmo 4):
    - sistema estable, sección 4.2.3,
    - sistema inestable, sección 4.2.4.
- **Dinámica totalmente desconocida (sistema estable):**
  - sin variaciones: secciones 4.3.1 y 4.3.2,
  - con variaciones: sección 4.4.

4.1. Especificaciones de hardware y software

En este estudio, se utilizó el lenguaje de programación GNU Octave en su versión 5.2.0, ejecutado en un sistema operativo Ubuntu 20.04.5 LTS. Las simulaciones se llevaron acabo en un computador ACER, modelo Nitro AN515-54 V1.53. La comparación de los resultados se hace respecto a los resultados obtenidos usando la función `DLQR` del paquete de control de GNU Octave.

4.2. Dinámica parcialmente conocida

En este escenario, el modelo del sistema es usado en la simulación numérica para la generación de las trayectorias y en el diseño de la ganancia de control admisible inicial. Además, todos los ejemplos con dinámica parcialmente conocida usan las condiciones iniciales $x_0 = [0.5 \quad -0.4]^T$.

4.2.1. Ejemplo 1: Sistema estable usando el Algoritmo 3

Se considera el sistema de la sección 3.1.1.1, donde los vectores de pesos W^0 de LS y W^0 de RLS están dados por $W^0_{LS} = W^1_{RLS} = [p_{11} \quad p_{12} \quad p_{22}]^T$. Aunque el Algoritmo 3 requiere una política de control $h_0(x)$, no es necesario que esta política sea admisible. Por ello, se utilizaron los valores de los parámetros de P_0 indicados en la Tabla 3 para el cálculo de K_0 mediante la ecuación (49), dando como resultado $K_0 = [0.63267 \quad -0.30160]$. El vector K_0 tiene un efecto estabilizante sobre el sistema. Para la PE se usa una señal aperiódica con una constante c_r para diminuir la amplitud. Para más detalles de la implementación, ver la sección A.2 del Anexo A.

Tabla 3. Valores de los parámetros requeridos por el Algoritmo 3, para el actor $\beta = 0.0102$.

Generales		**Crítico**	
Parámetro	**Valor**	**Parámetro**	**Valor**
N	450	n_{ls}	7
n_{ep}	12	λ	0.99
q_i	1	δ	10^{-6}
r_i	2	c_r	0.102
γ	0.999		
P_0	$\mathbb{I}_{2\times 2}$		

Las figuras 29 y 32 muestran los valores de los parámetros del vector de pesos W durante los diferentes episodios del entrenamiento, obtenidos en la sintonización del crítico para el sistema estable e inestable respectivamente. Las figuras 30 y 33 muestran los valores de los parámetros del vector de ganancias K, obtenidos en la sintonización del actor para el sistema estable e inestable respectivamente. Con el objetivo conocer el episodio en el cual detener el entrenamiento, las figuras 31 y 34 muestra los cambios medidos entre las estimaciones de la matriz P y K de cada episodio para el sistema estable e inestable respectivamente. Se usa la norma euclidiana de la diferencia entre la estimación futura y la estimación actual. Los valores de los parámetros estimados después de un entrenamiento de 12 episodios se muestran en en el renglón 3 de la Tabla 5.

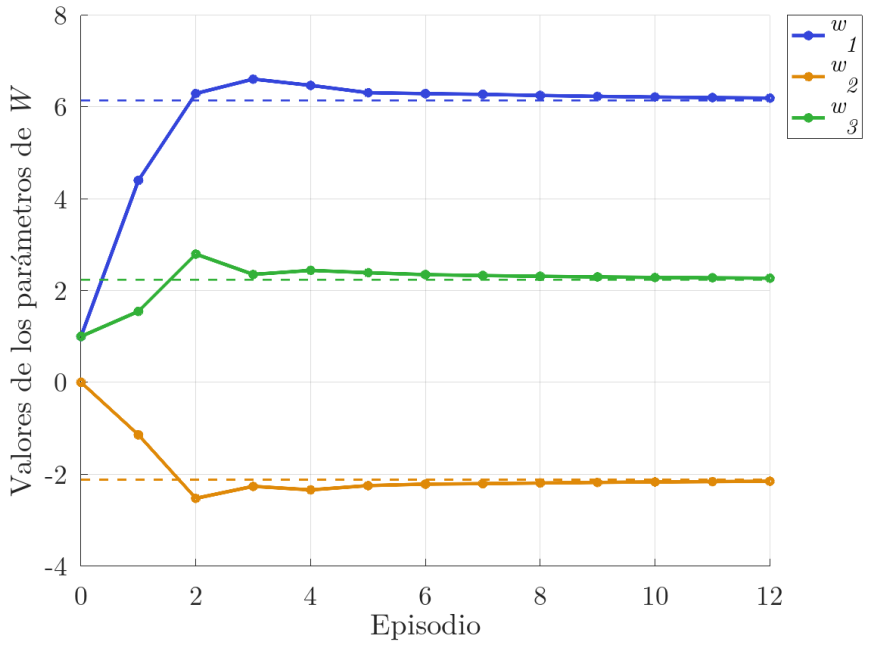

Figura 29. Convergencia de los parámetros del vector W hacia valores óptimos usando el Algoritmo 3.

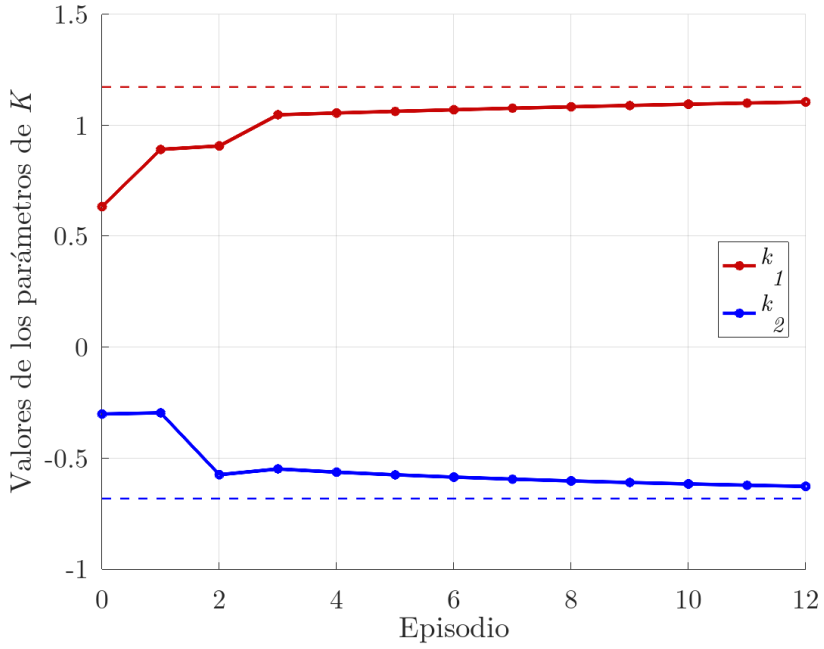

Figura 30. Convergencia de los parámetros del vector K hacia valores óptimos usando el Algoritmo 3.

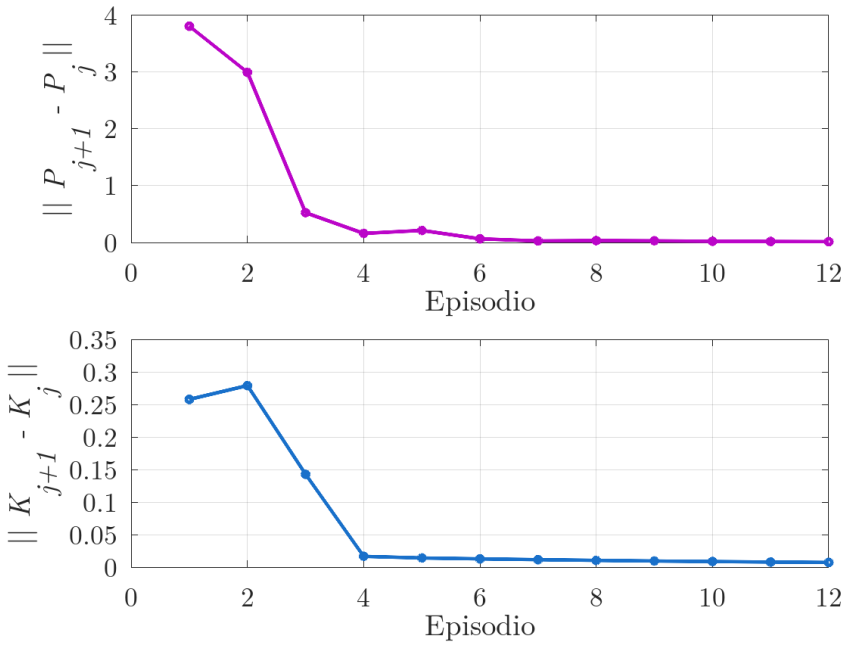

Figura 31. Norma euclidiana de la diferencia medida entre P_{j+1} y P_j (gráfica superior) y de la diferencia medida entre K_{j+1} y K_j (gráfica inferior) usando el Algoritmo 3.

4.2.2. Ejemplo 2: Sistema inestable usando el Algoritmo 3

Se considera el sistema de la sección 3.1.1.2, donde la matriz $P_0 = \text{diag}(10, 0.01)$ y los vectores de pesos $W_{LS}^0 = W_{RLS}^1 = [p_{11} \quad p_{12} \quad p_{22}]^T$. Se usan los valores de P_0 para el cálculo de K_0 mediante la ecuación (49), dando como resultado $K_0 = [-0.47506 \quad 0.95245]$. Donde K_0 tiene un efecto estabilizante sobre el sistema, con polos en lazo cerrado $Z_1 = -0.47937$ y $Z_2 = 0.60558$. Se utilizaron los mismos parámetros indicados en la Tabla 3, exceptuando $\beta = 0.00202$, $\delta = 10^{-4}$ y $N = 500$, estos cambios fueron debido a que por ser un sistema inestable requirió mayor tiempo de entrenamiento. Para la PE se usa una señal aperiódica. Para más detalles de la implementación, ver la sección A.3 del Anexo A.

Los valores de los parámetros estimados por el Algoritmo 3 para el sistema inestable, se muestran en el renglón 2 de la Tabla 4.

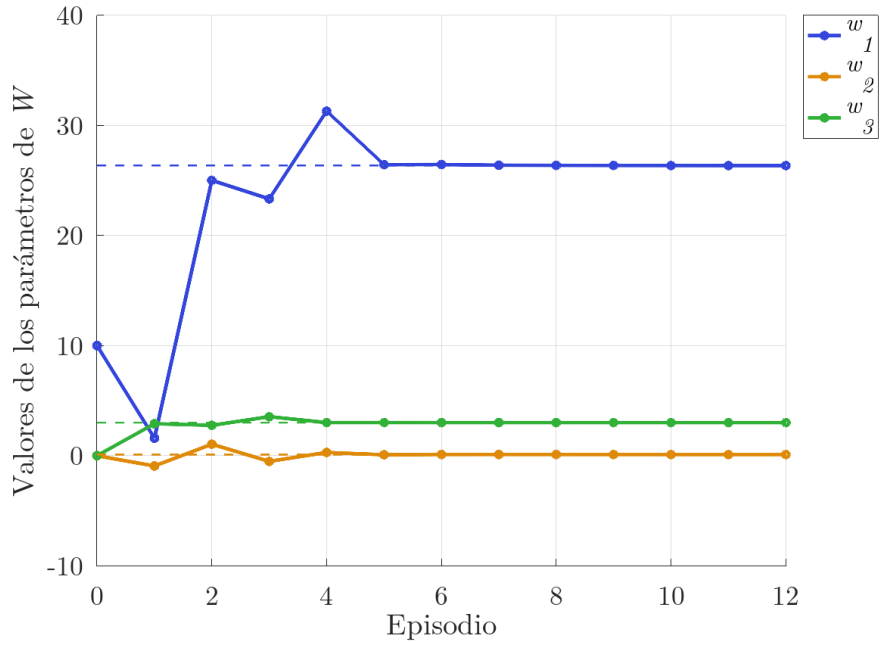

Figura 32. Convergencia de los parámetros del vector W hacia valores óptimos usando el Algoritmo 3 - sistema inestable.

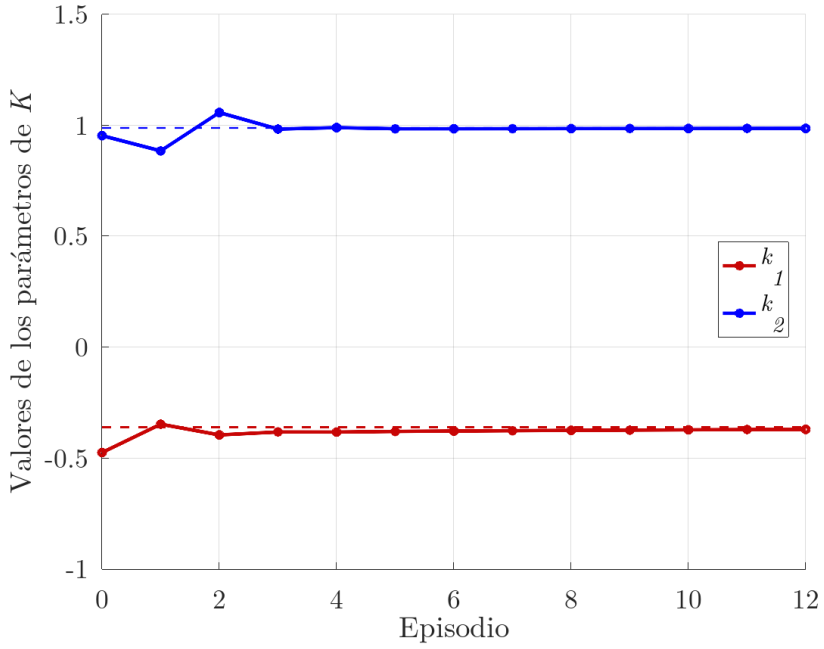

Figura 33. Convergencia de los parámetros del vector K hacia valores óptimos usando el Algoritmo 3 - sistema inestable.

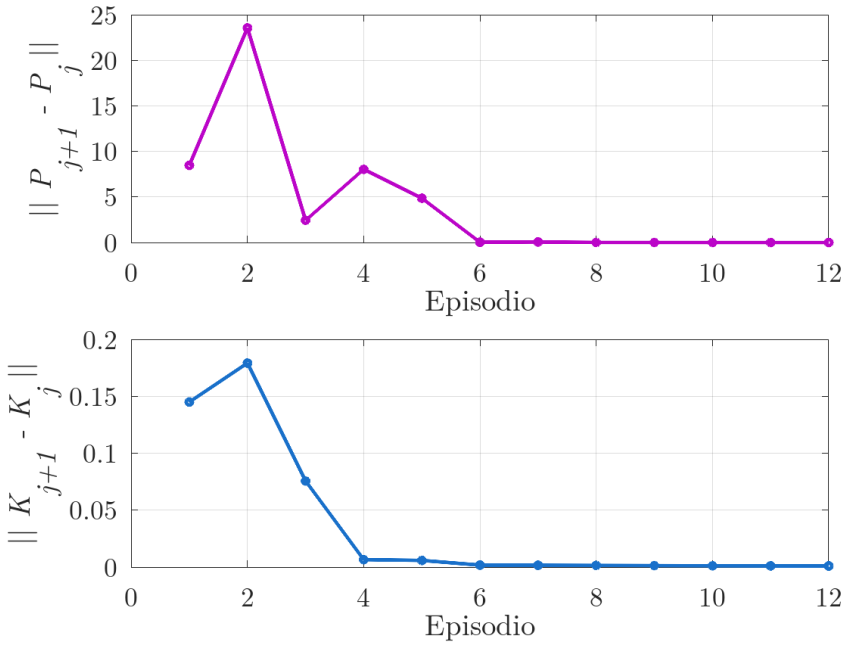

Figura 34. Norma euclidiana de la diferencia medida entre P_{j+1} y P_j (gráfica superior) y de la diferencia medida entre K_{j+1} y K_j (gráfica inferior) usando el Algoritmo 3 - sistema inestable.

4.2.3. Ejemplo 3: Sistema estable usando el Algoritmo 4

Se considera el sistema de la sección 3.1.1.1. Una condición del Algoritmo 4 es que la política de control $h_0(x)$ sea admisible, es decir, debe estabilizar el sistema. Por ello, se utilizaron los mismos parámetros de P_0 indicados en la Tabla 3 para el cálculo de K_0 mediante la ecuación (49), dando como resultado $K_0 = [0.63267 \quad -0.30160]$. El vector K_0 tiene un efecto estabilizante sobre el sistema, con polos en lazo cerrado $Z_{1,2} = 0.63267 \pm 0.45048i$.

Se usan los mismos parámetros mostrados en la Tabla 3 para el entrenamiento, solo cambió la longitud de la trayectoria $N = 250$ debido a que el algoritmo de iteración de política requirió menor cantidad de datos medidos para la convergencia. Para la PE se usa una señal aperiódica. Para más detalles de la implementación, ver la sección A.4 del Anexo A.

Las figuras 35 y 38 muestran los valores de los parámetros del vector de pesos W durante los diferentes episodios del entrenamiento, obtenidos en la sintonización del crítico para el sistema estable e inestable respectivamente. Las figuras 36 y 39 muestran los valores de los parámetros del vector de ganancias K obtenidos en el actor para el sistema estable e inestable respectivamente.

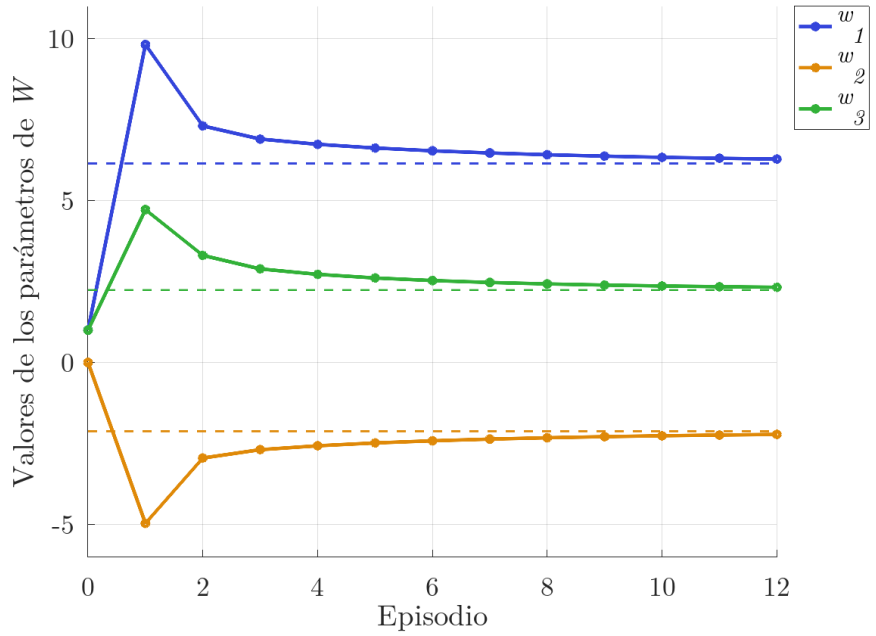

Figura 35. Convergencia de los parámetros del vector W hacia valores óptimos usando el Algoritmo 4.

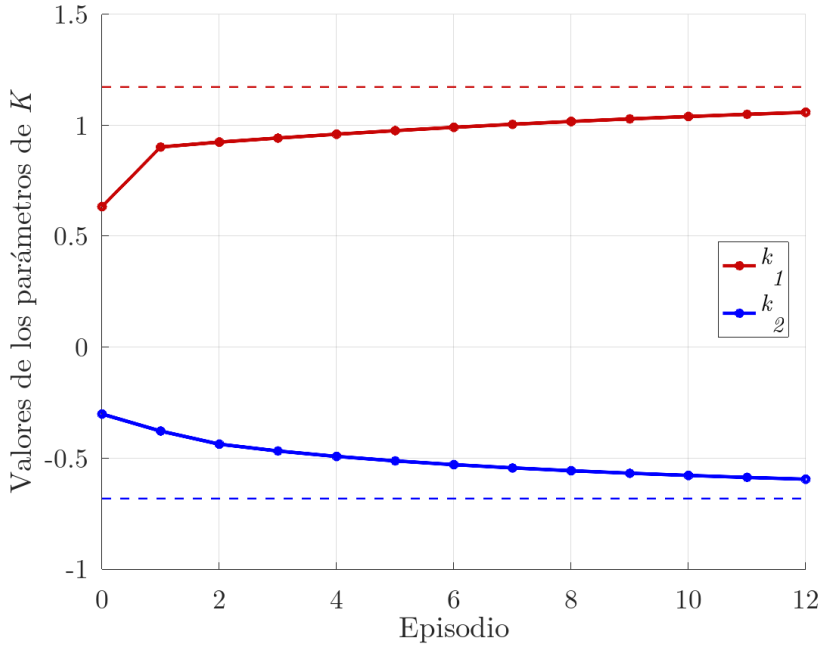

Figura 36. Convergencia de los parámetros del vector K hacia valores óptimos usando el Algoritmo 4.

Las figuras 37 y 40 muestran los cambios medidos entre las estimaciones de la matriz P y K de cada episodio para el sistema estable e inestable respectivamente. Los valores de los parámetros estimados después de un entrenamiento de 12 episodios se muestran en en el renglón 4 de la Tabla 5.

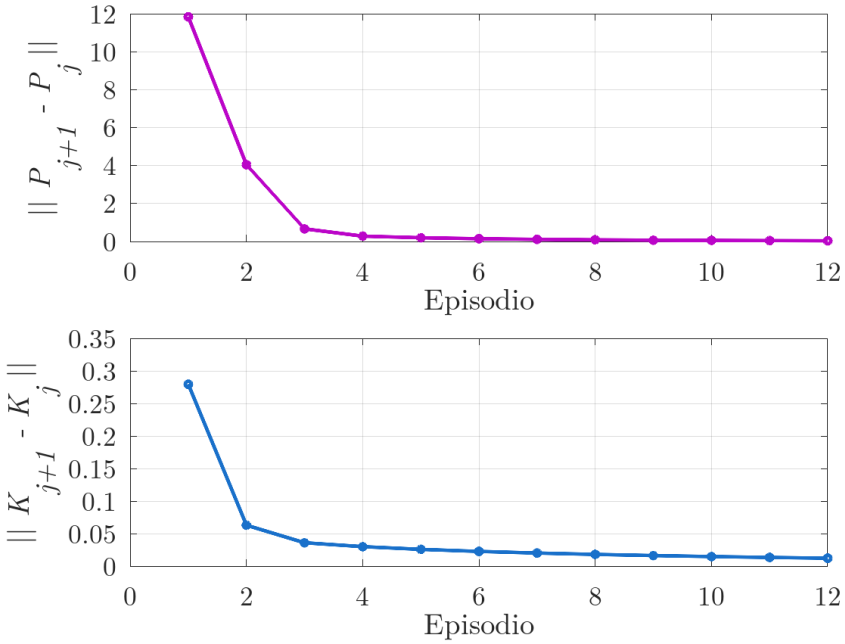

Figura 37. Norma euclidiana de la diferencia medida entre P_{j+1} y P_j (gráfica superior) y la diferencia medida entre K_{j+1} y K_j (gráfica inferior) usando el Algoritmo 4.

4.2.4. Ejemplo 4: Sistema inestable usando el Algoritmo 4

Se considera el sistema de la sección 3.1.1.2 y la matriz de $P_0 = \text{diag}(6, 0.01)$. Se asegura tener un política de control admisible usando los valores de P_0 en la ecuación (49) para el cálculo de K_0, dando como resultado $K_0 = [-0.45973 \quad 0.92321]$. El vector K_0 tiene un efecto estabilizante sobre el sistema, con polos en lazo cerrado $Z_1 = -0.61719$ y $Z_2 = 0.75952$. Además, se utilizaron los mismos parámetros indicados en la Tabla 3, exceptuando $\beta = 0.00202$, $\delta = 10^{-4}$ y $N = 1050$, estos cambios fueron debido a que por ser un sistema inestable requirió mayor tiempo de entrenamiento, una tasa de aprendizaje para el actor menor y una condición del error mayor. Para la PE se usa una señal aperiódica. Para más detalles de la implementación, ver la sección A.5 del Anexo A.

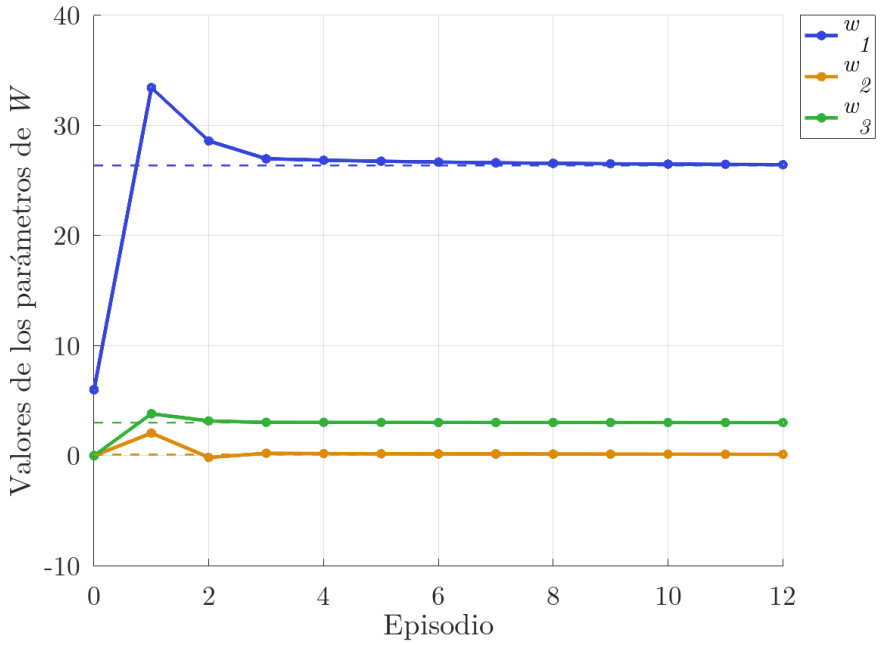

Figura 38. Convergencia de los parámetros del vector W hacia valores óptimos usando el Algoritmo 4 - sistema inestable.

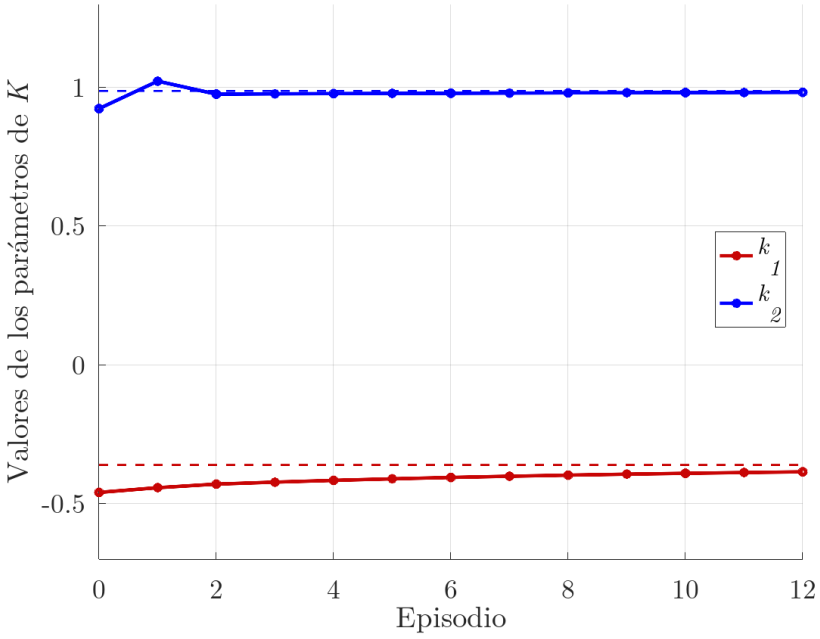

Figura 39. Convergencia de los parámetros del vector K hacia valores óptimos usando el Algoritmo 4 - sistema inestable.

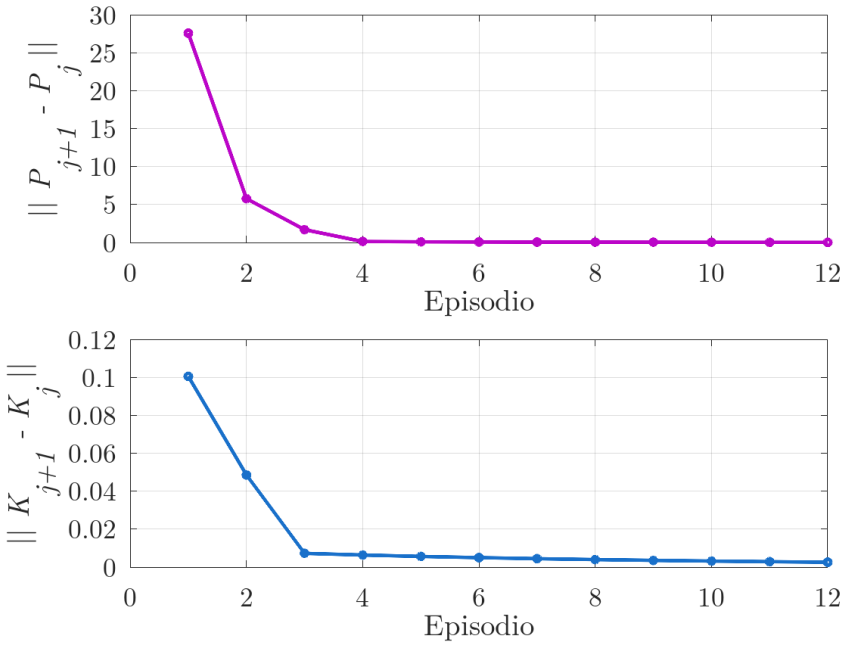

Figura 40. Norma euclidiana de la diferencia medida entre P_{j+1} y P_j (gráfica superior) y de la diferencia medida entre K_{j+1} y K_j (gráfica inferior) usando el Algoritmo 4 - sistema inestable.

La Tabla 4 muestra los valores óptimos al usar la función `DLQR`, también muestra en el renglón 3 los estimados por el Algoritmo 4 para el sistema inestable después de un entrenamiento de 12 episodios.

Tabla 4. Valores óptimos y estimados de la matriz P y el vector K - sistema inestable.

Parámetro	p_{11}	p_{12}	p_{22}	k_1	k_2
Óptimo	26.35369	0.11143	3.01079	−0.36083	0.98709
Ejemplo 2	26.34298	0.11413	3.01142	−0.37072	0.98487
Ejemplo 4	26.42532	0.13060	3.01472	−0.38558	0.98192

4.2.5. Comparación de resultados con entrenamiento continuo

Con el objetivo de comparar los parámetros estimados en los ejemplos 1 (sección 4.2.1) y 3 (sección 4.2.3), se implementó el algoritmo de iteración de política con aprendizaje TD en entrenamiento continuo para el sistema estable descrito en la sección 3.1.1.1. La referencia principal para la implementación de este algoritmo es Lewis et al. (2012a). Se usan dos esquemas en un sistema estable cuya dinámica es parcialmente conocida, usando iteración de política con aprendizaje de TD en entrenamiento continuo. En ambos se usa GD para el actor y para el crítico LS+RLS en un caso y LS+GD para el otro.

Debido a que el entrenamiento continuo para dinámica parcialmente conocida no fue objeto de estudio en la metodología de esta tesis, el pseudocódigo utilizando LS+RLS para el crítico se muestra en la sección C.1 del Anexo C. Se usan los valores de $P_0 = \text{diag}(15, 1)$ en ambos escenarios.

La Figura 41 muestra los valores de los parámetros del vector de pesos W en cada iteración del entrenamiento continuo, obtenidos en la sintonización del crítico a través del esquema LS+RLS. La Figura 42 muestra los valores de los parámetros del vector de ganancias K en cada iteración, obtenidos en la sintonización del actor a través del GD. La Figura 43 muestra los estados y la salida del sistema afectados por la señal multisenoidal con decaimiento exponencial como PE.

La Figura 44 muestra los valores de los parámetros del vector de pesos W en cada iteración del entrenamiento continuo, obtenidos en la sintonización del crítico a través del esquema LS+GD. La Figura 45 muestra los valores de los parámetros del vector de ganancias K en cada iteración del entrenamiento, obtenidos en la sintonización del actor a través del GD. La Figura 46 muestra los estados y la salida del sistema afectados por la señal multisenoidal con decaimiento exponencial como PE.

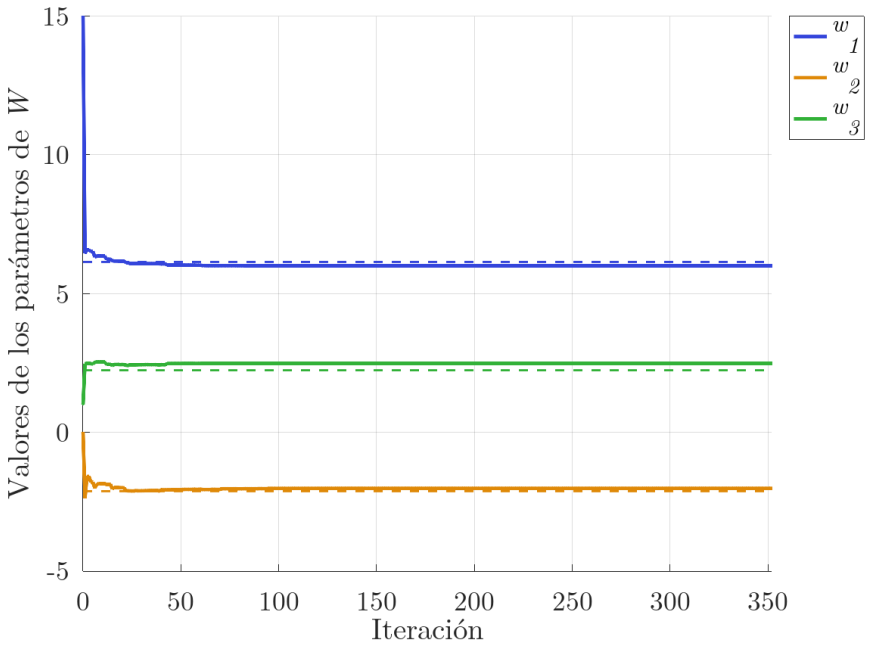

Figura 41. Convergencia de los parámetros del vector W hacia valores óptimos usando el Algoritmo 6. Sintonización del crítico usando LS+RLS con los parámetros iniciales: $n_{ls} = 3$, $\lambda = 0.99$ y $\delta = 10^{-2}$.

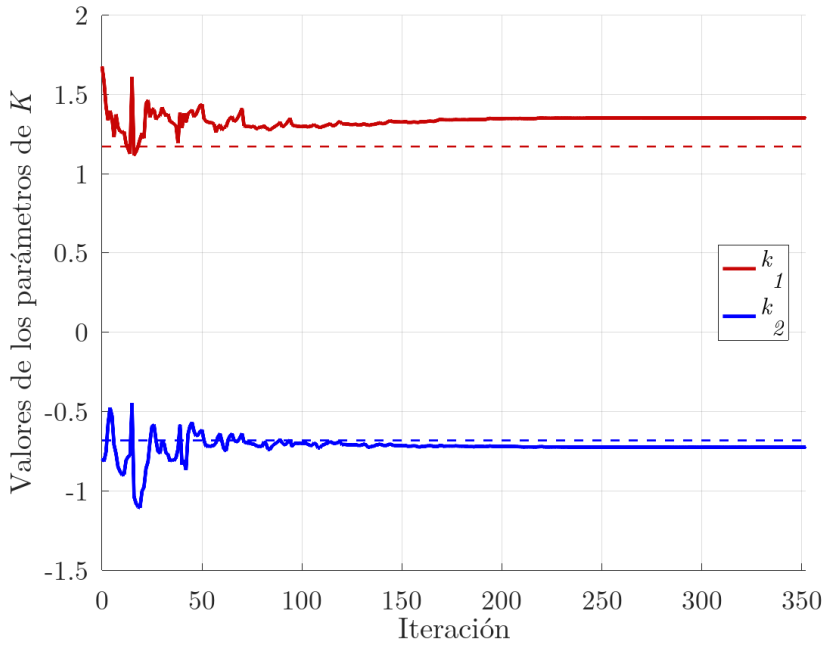

Figura 42. Convergencia de los parámetros del vector K hacia valores óptimos usando el Algoritmo 6. Sintonización del actor usando GD con $\beta = 0.3939$.

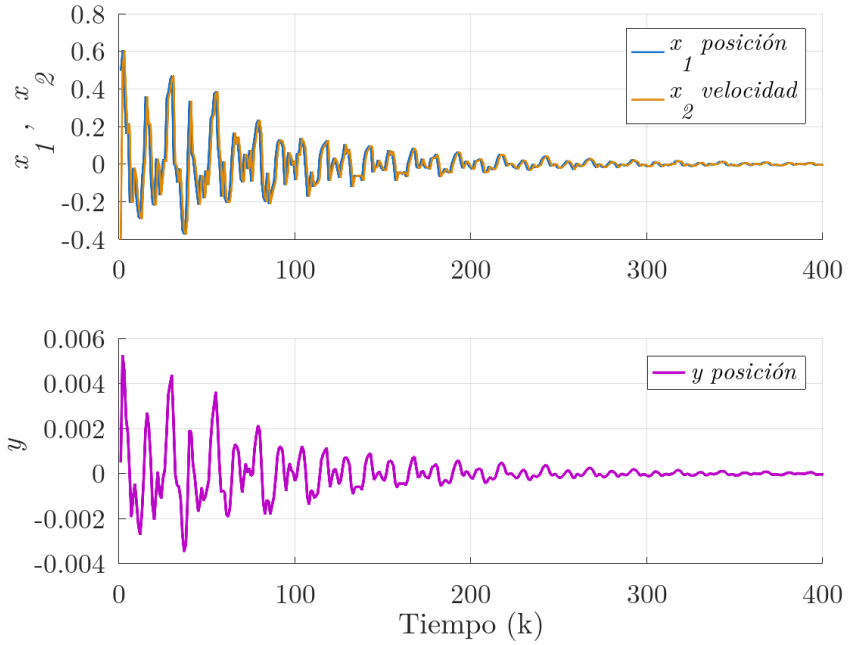

Figura 43. Comportamiento de los estados y la salida durante el entrenamiento continuo con condiciones iniciales $x_0 = [0.5 \quad -4]^T$. Los parámetros generales para el entrenamiento continuo usando LS+RLS y GD en el Algoritmo 6 son: $N = 400$, $\gamma = 0.999$, $Q = \mathbb{I}_{2\times 2}$ y $R = 2$. Tiempo de ejecución ≈ 3.07284 segundos.

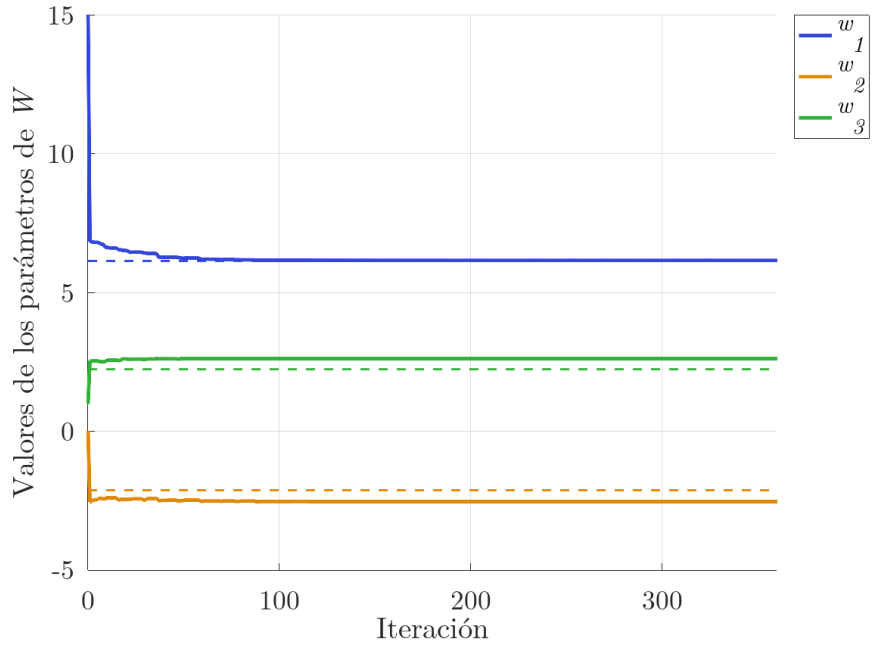

Figura 44. Convergencia de los parámetros del vector W hacia valores óptimos usando el Algoritmo 6. Sintonización del crítico usando LS+GD con los parámetros iniciales: $n_{ls} = 4$, $\alpha = 2.01988$ y $\delta = 10^{-2}$.

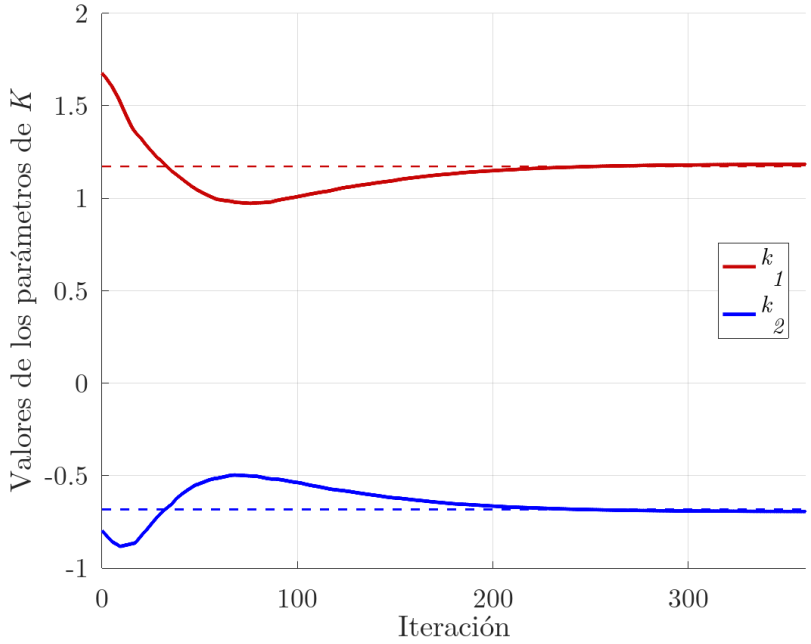

Figura 45. Convergencia de los parámetros del vector K hacia valores óptimos usando el Algoritmo 6 (LS+GD). Sintonización del actor usando GD con $\beta = 0.007339$.

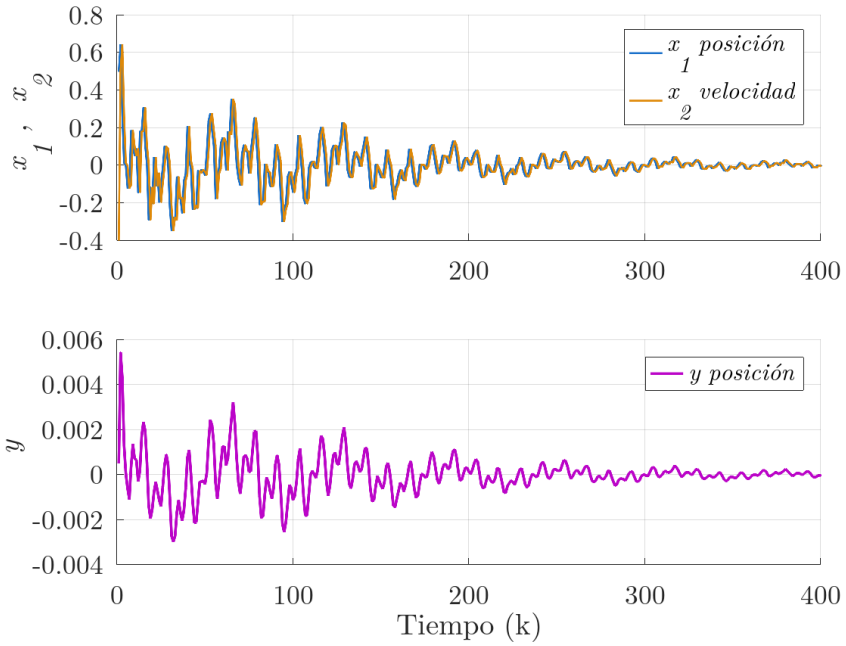

Figura 46. Comportamiento de los estados y la salida durante el entrenamiento continuo con condiciones iniciales $x_0 = [0.5 \quad -4]^T$. Los parámetros generales para el entrenamiento continuo usando LS+GD y GD en el Algoritmo 6 son: $N = 400$, $\gamma = 0.999$, $Q = \mathbb{I}_{2\times 2}$ y $R = 2$. Tiempo de ejecución ≈ 3.05006 segundos.

Para comparar la calidad de los parámetros obtenidos por cada esquema contra los parámetros óptimos, se realizaron simulaciones usando ambos juegos de parámetros y se calculó el error cuadrático medio (MSE, por sus siglas en inglés) entre ambas respuestas. La Tabla 5 muestra los valores obtenidos en los ejemplos desarrollados con entrenamiento episódico y entrenamiento continuo para el sistema estable, cuando se tiene conocimiento parcial de la dinámica del sistema.

Tabla 5. Comparación de valores óptimos y estimados del vector W y el vector K para el sistema estable.

Parámetro	p_{11}	p_{12}	p_{22}	k_1	k_2	**MSE**
Óptimo	6.1414	−2.1197	2.2351	1.17138	−0.68253	0
Arbitrario	1	0	1	0.63267	−0.30160	1422.8
Ejemplo 1	6.1922	−2.1559	2.2691	1.10414	−0.62737	17.032
Ejemplo 3	6.2776	−2.2193	2.3224	1.05755	−0.59515	46.471
LS+RLS, GD	6.0008	−2.0222	2.4847	1.35174	−0.72508	61.070
LS+GD, GD	6.1571	−2.5329	2.6183	1.18243	−0.69387	0.55785

Los valores arbitrarios corresponden a valores que generalmente se eligen en estimación de parámetros. Se consiguen buenos resultados en la convergencia de los parámetros al usar las siguientes frecuencias angulares en las señales multisenoidales, obtenidas por tanteo, y descritas por:

$$Ce^{-\xi k}f(k) = 0.20099e^{-0.010899k}(\text{sen}^3(k)\cos(k) + \text{sen}^2(2k)\cos(0.26k) + \text{sen}^2(-1.4k)\cos(0.5k) + \ldots$$
$$+ \text{sen}^5(k) + \text{sen}^5(0.5k) + 0.4\,\text{sen}^2(1.99k)\cos(2k) + 0.4\,\text{sen}^5(10k)), \quad (130)$$

$$Ce^{-\xi k}f(k) = 0.198e^{-0.0082k}(\text{sen}^2(k)\cos(k) + \text{sen}^2(2k)\cos(0.1k) + \ldots$$
$$+ \text{sen}^2(-1.4k)\cos(0.5k) + \text{sen}^5(k)). \quad (131)$$

Se usa (130) en el esquema LS+RLS y (131) en el esquema LS+GD.

4.3. Dinámica totalmente desconocida

Para los ejemplos con dinámica totalmente desconocida se considera el sistema estable descrito en la sección 3.1.1.1, con condiciones iniciales $x_0 = [5 \quad -4]^T$. Debido a que el Algoritmo 5 requiere de una política de control $h_0(x)$ admisible, se propone una matriz H_0 como:

$$H_0 = \begin{bmatrix} 14 & -2 & 2 \\ -8 & 3 & -1 \\ 8 & -5 & 4 \end{bmatrix}, \quad (132)$$

y mediante la ecuación (55) se calcula K_0, dando como resultado $K_0 = [2 \quad -1.25]$. El vector K_0 tiene un efecto estabilizante sobre el sistema, con polos en lazo cerrado $Z_1 = -0.64075$ y $Z_2 = 0.53875$.

4.3.1. Ejemplo 5: Sistema estable usando el Algoritmo 5 (LS+GD)

La Tabla 6 muestra los valores de los parámetros requeridos por el Algoritmo 5 para el entrenamiento continuo usando el esquema LS+GD.

Tabla 6. Valores de los parámetros requeridos por el Algoritmo 5 usando LS+GD para la sintonización del crítico.

Generales		Crítico	
Parámetro	**Valor**	**Parámetro**	**Valor**
N	4000	n_{ls}	7
q_i	1	α	120.02
r_i	2	δ	10^{-2}
γ	0.999	c_r	0.202

La Figura 47 muestra los valores de los parámetros del vector de pesos W en cada iteración del entrenamiento, obtenidos en la sintonización del crítico. La Figura 48 muestra los valores de los parámetros del vector de ganancias K en cada iteración del entrenamiento continuo y la Figura 49 muestra los estados y la salida del sistema afectados por la señal aperiódica como PE. Para más detalles de la implementación, ver la sección B.1 del Anexo B.

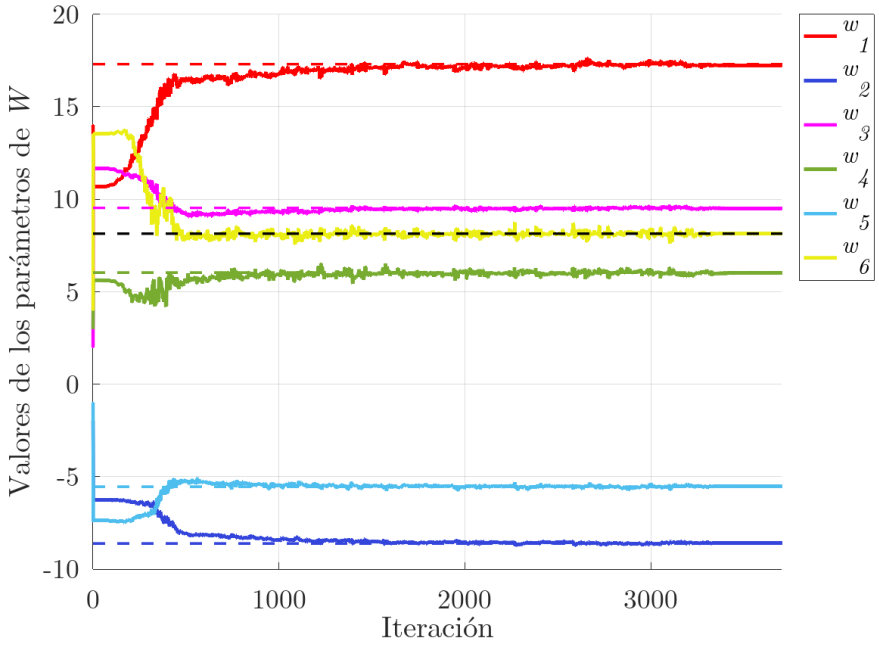

Figura 47. Convergencia de los parámetros del vector W hacia valores óptimos usando el Algoritmo 5 con LS+GD.

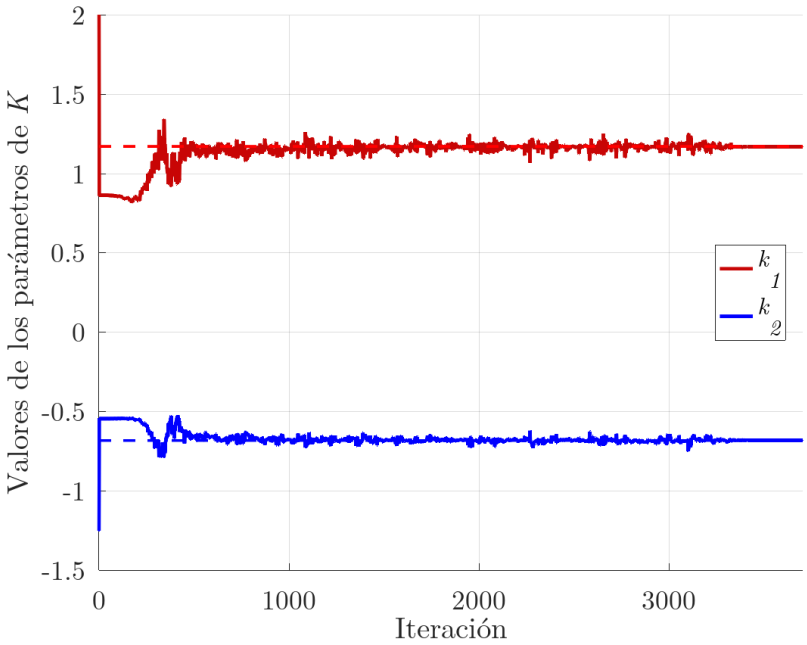

Figura 48. Convergencia de los parámetros del vector K hacia valores óptimos usando el Algoritmo 5.

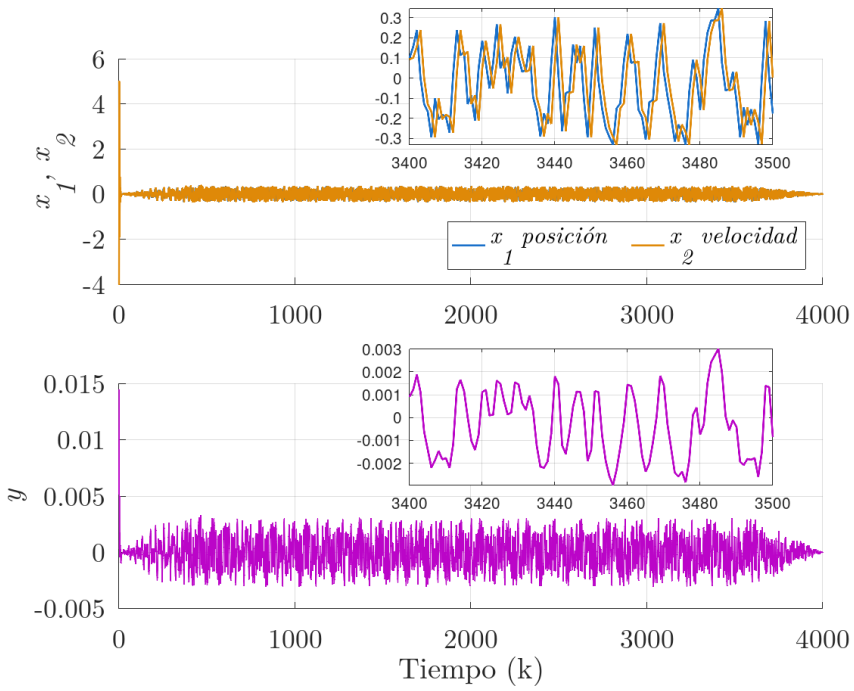

Figura 49. Comportamiento de los estados y la salida durante el entrenamiento continuo usando Algoritmo 5 con LS+GD. Tiempo de ejecución ≈ 3.83227 segundos.

4.3.2. Ejemplo 6: Sistema estable usando el Algoritmo 5 (LS+RLS)

Se usan los mismos valores de los parámetros generales indicados en la Tabla 6, exceptuando el factor de olvido $\lambda = 0.99$ debido a que este ejemplo usa el esquema LS+RLS. La Figura 50 muestra los valores de los parámetros del vector de pesos W en cada iteración del entrenamiento continuo, obtenidos en la sintonización del crítico. La matriz $\Gamma(n_{ls})$ diseñada en por LS fue multiplicada por un escalar igual a 1000 para mejorar los tiempos de convergencia. Para más detalles de la implementación, ver la sección B.2 del Anexo B.

La Figura 51 muestra los valores de los parámetros del vector de ganancias K en cada iteración del entrenamiento continuo y la Figura 52 muestra los estados y la salida del sistema afectados por la señal multisenoidal con decaimiento exponencial descrita por:

$$Ce^{-\xi k}f(k) = 0.20099e^{-0.000269k}(\text{sen}^3(k)\cos(k) + \text{sen}^2(2k)\cos(0.46k) + \text{sen}^2(-1.4k)\cos(0.5k) + \ldots$$
$$+ \text{sen}^5(k) + \text{sen}^5(0.5k) + 0.4\,\text{sen}^2(1.99k)\cos(2k) + 0.4\,\text{sen}^5(18.0k)). \quad (133)$$

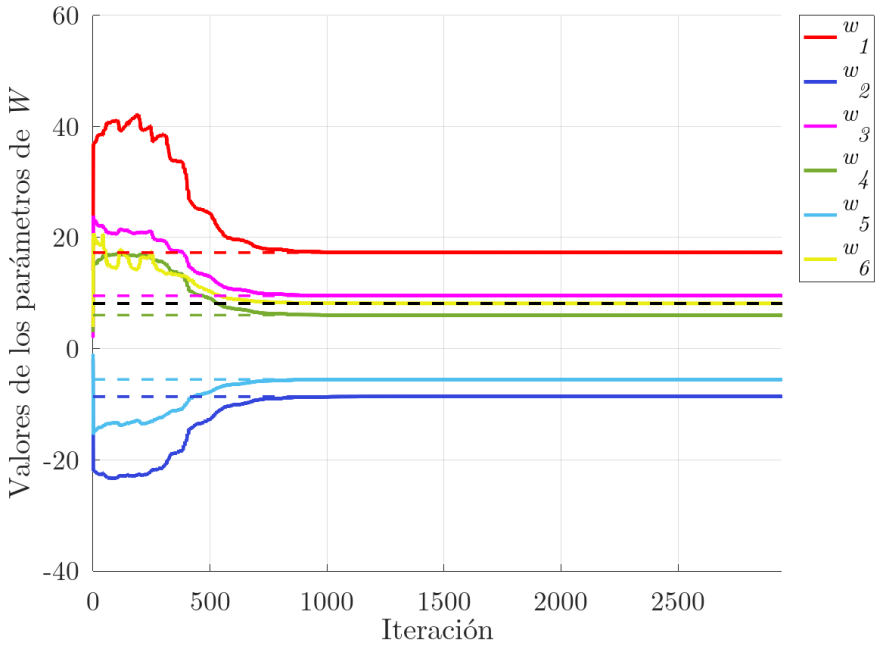

Figura 50. Convergencia de los parámetros del vector W hacia valores óptimos usando el Algoritmo 5 con LS+RLS. La matriz $\Gamma(n_{ls})$ es escalada por 1000.

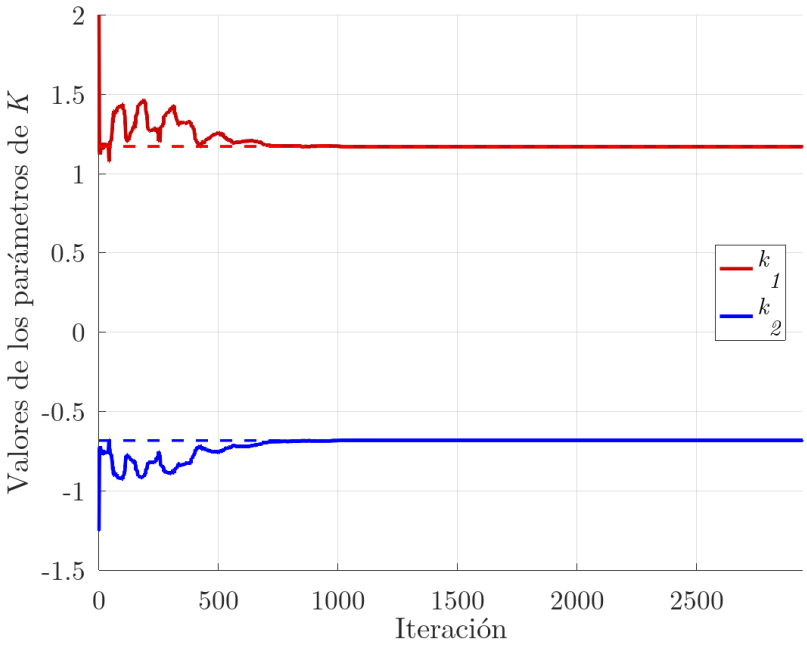

Figura 51. Convergencia de los parámetros del vector K hacia valores óptimos usando el Algoritmo 5.

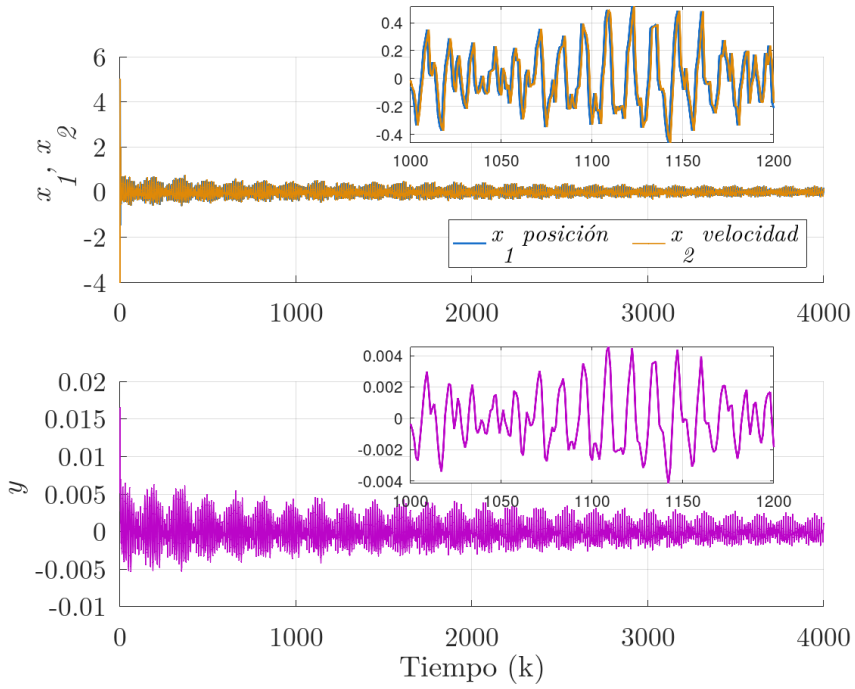

Figura 52. Comportamiento de los estados y la salida durante el entrenamiento continuo usando Algoritmo 5 con LS+RLS. Tiempo de ejecución ≈ 1.1352 segundos.

Con el objetivo de comparar los parámetros sintonizados en los ejemplos 5 (sección 4.3.1) y 6 (sección 4.3.2), se implementa el Algoritmo 5 usando solo GD y RLS de manera independiente en la sintonización del crítico. Se consideran las mismas condiciones iniciales, la matriz H_0 descrita por (132) y las matrices $Q = \mathbb{I}_{2\times 2}$ y $R = 2$.

La Tabla 7 muestra los valores obtenidos y la medición del MSE entre las simulaciones al aplicar directamente al sistema cada vector de ganancias K estimado y el vector de ganancias K óptimo.

Los valores arbitrarios se escogieron de tal forma que generan un vector K estabilizante con valores lejanos de los óptimos. La figuras 53 y 54 muestran los valores de los parámetros del vector de pesos W en cada iteración del entrenamiento continuo usando GD y RLS respectivamente.

Tabla 7. Comparación de valores óptimos y estimados del vector W y el vector K para el sistema estable.

Parámetro	h_{11}	h_{12}	h_{13}	h_{22}	h_{23}	h_{33}	k_1	k_2	**MSE**
Óptimo	17.2962	−8.6202	9.5272	6.0227	−5.5512	8.1353	1.17138	−0.68253	0
Arbitrario	16	−7	7	7	−4	9	0.77778	−0.44444	550.57
Ejemplo 5	17.2338	−8.5853	9.5021	5.9986	−5.5346	8.1243	1.16958	−0.68124	0.01087
Ejemplo 6	17.2868	−8.6151	9.5216	6.0200	−5.5481	8.1319	1.17090	−0.68227	0.0006349
GD	14.8340	−7.4545	8.5215	5.2772	−5.0017	7.7319	1.10212	−0.64689	12.850
RLS	17.2868	−8.6151	9.5216	6.0200	−5.5481	8.1319	1.17090	−0.68227	0.0006349

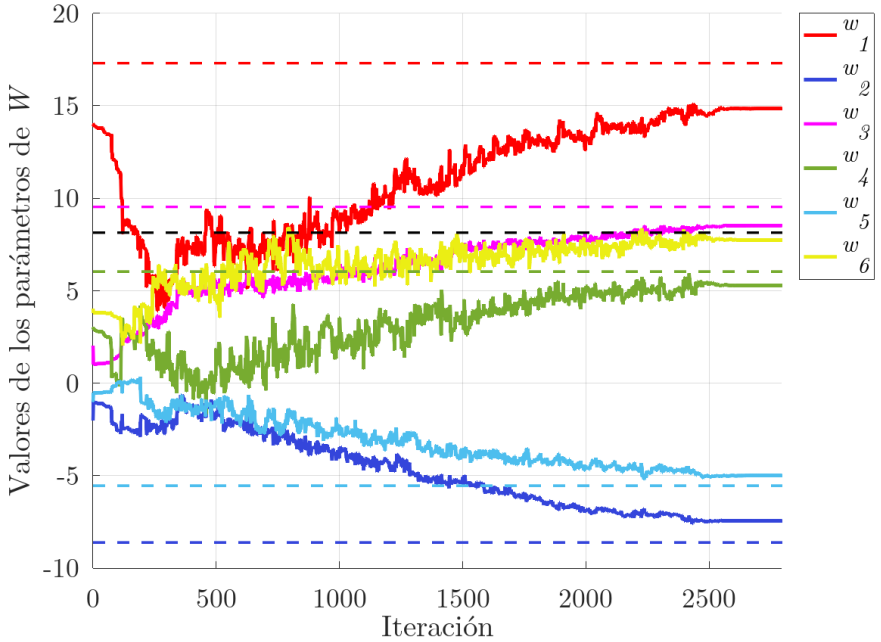

Figura 53. Convergencia de los parámetros del vector W hacia valores óptimos usando el Algoritmo 5 con GD para la sintonización del crítico. Con los parámetros $N = 4000$, $\gamma = 0.999$, $\alpha = 120.02$, $\delta = 10^{-2}$ y $c_r = 0.202$. Tiempo de ejecución ≈ 0.90386 segundos.

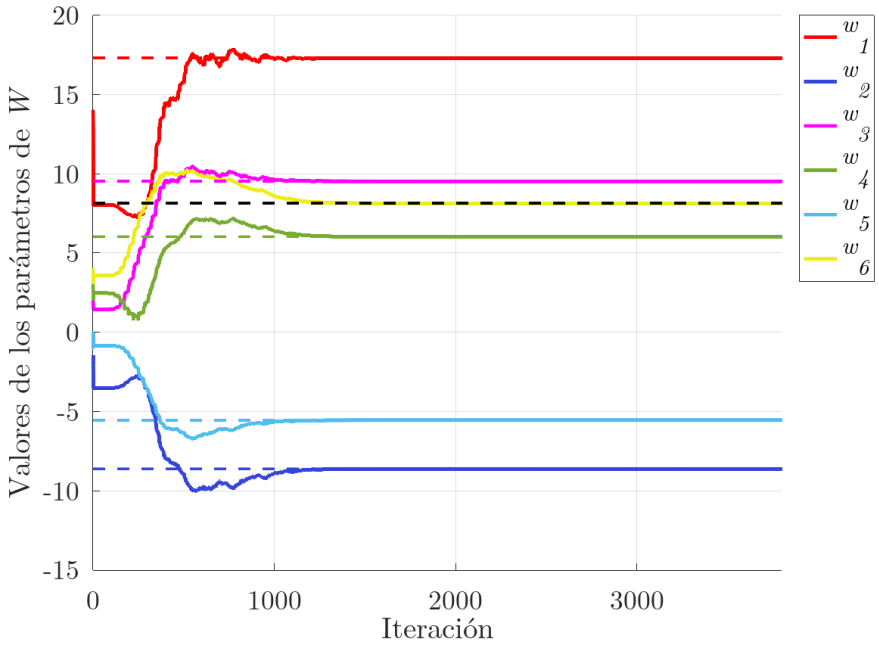

Figura 54. Convergencia de los parámetros del vector W hacia valores óptimos usando el Algoritmo 5 con RLS para la sintonización del crítico. Con los parámetros $N = 4000$, $\gamma = 0.999$, $\lambda = 0.99$, $\delta = 10^{-2}$ y $c_r = 0.202$. La matriz $\Gamma(k) = \mathbb{I}_{6\times 6} \times 1000$. Tiempo de ejecución ≈ 1.2035 segundos.

4.4. Prueba de adaptabilidad a variaciones con dinámica totalmente desconocida

Para la prueba de adaptabilidad de los algoritmos de RL vistos en la sección 4.3, se consideran variaciones en los parámetros del sistema estable únicamente en la matriz de dinámica A. Las variaciones corresponden a cambios físicos que ocurren en los elementos de la planta.

Se considera el sistema B como la peor variación que puede ocurrir en el sistema A (sistema nominal). Los parámetros de la peor variación se muestran en la Tabla 8, junto a una variación intermedia indicada por los parámetros del sistema A2.

La prueba de adaptabilidad consiste de los siguientes pasos:

1. Realizar un entrenamiento usando el sistema B hasta la convergencia.

2. Finalizado el paso 1, realizar el entrenamiento principal sobre el sistema A considerando el tiempo de entrenamiento requerido para la convergencia del sistema B.

3. Mientras se realiza el punto 2 se puede provocar una variación sobre el sistema A que sea igual o menor a la peor variación.

Los sistemas A2 y B suceden aproximadamente al $45\,\%$ y $77\,\%$ respectivamente del tiempo del entrenamiento principal.

Tabla 8. Parámetros del sistema masa-resorte-amortiguador nominal, variación intermedia y peor variación.

Símbolo	**Descripción**	**Valores**		
		Sistema A	Sistema A2	Sistema B
m	Masa	$1\ kg$	$1.5\ kg$	$2\ kg$
b	Coeficiente de amortiguamiento	$1\ N \cdot s/m$	$0.75\ N \cdot s/m$	$0.5\ N \cdot s/m$
k	Coeficiente de elasticidad	$0.72\ N/m$	$0.54\ N/m$	$0.36\ N/m$

Se realiza la discretización de los sistemas con un tiempo de muestreo $T = 0.1\ seg$, como resultado el sistema A2 se define:

$$x_{k+1} = \begin{bmatrix} 1.9480 & -0.9512 \\ 1 & 0 \end{bmatrix} x_k, \qquad y_k = \begin{bmatrix} 0.0049 & 0.0048 \end{bmatrix},$$

cuyos polos $Z_{1,2} = 0.97400 \pm 0.05024i$ están dentro del círculo unitario por lo que es estable. El sistema B se define como:

$$x_{k+1} = \begin{bmatrix} 1.9740 & -0.9753 \\ 1 & 0 \end{bmatrix} x_k, \qquad y_k = \begin{bmatrix} 0.005 & 0.0049 \end{bmatrix},$$

cuyos polos $Z_{1,2} = 0.98700 \pm 0.03363i$ se encuentran en la región de estabilidad.

Para los siguientes ejemplos se usan las mismas condiciones iniciales y la matriz H_0 de la sección 4.3.

4.4.1. Ejemplo 7: Sistema estable con dos variaciones (LS+GD)

Para facilitar la observación de la convergencia, las figuras 55 y 56 solo muestran el valor del parámetro w_1 en cada iteración del entrenamiento continuo.

Nótese que una vez que inicia la variación intermedia indicada como sistema A2 en la Figura 55 (alrededor de la iteración 4900) al algoritmo le toma aproximadamente 2600 iteraciones para alcanzar la convergencia a valores óptimos. Respecto al sistema B (peor variación), se necesitan aproximadamente 1500 iteraciones para alcanzar la convergencia a valores óptimos. Considerando que el tiempo de muestro es de 0.1 seg, le tomaría al controlador 2.5 min adaptarse a la peor variación después de sucedida la variación intermedia.

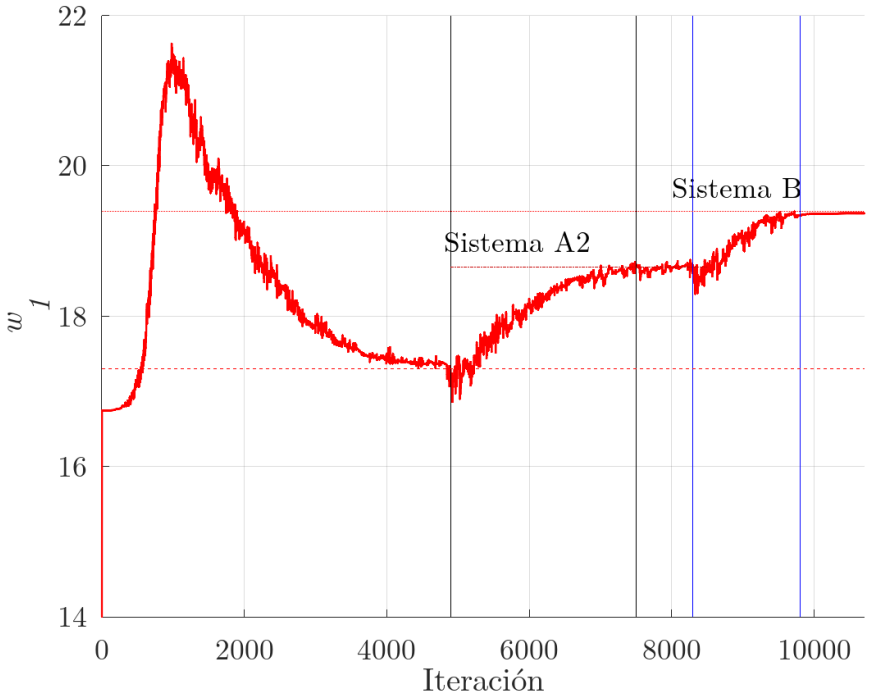

Figura 55. Convergencia del parámetro w_1 con dos variaciones usando el Algoritmo 5, con LS+GD para la sintonización del crítico. Con los parámetros $N = 11000$, $\gamma = 0.999$, $n_{ls} = 21$, $\alpha = 70.02$, $\delta = 10^{-2}$ y $c_r = 0.202$. Tiempo de ejecución ≈ 11.2009 segundos.

4.4.2. Ejemplo 8: Sistema estable con dos variaciones (LS+RLS)

El número de iteraciones se reduce considerablemente usando el esquema LS+RLS en comparación al esquema LS+GD usado en el Ejemplo 7. Se necesitan aproximadamente 500 iteraciones para alcanzar la convergencia a valores óptimos usando el sistema A2 (variación intermedia). Considerando un tiempo de muestreo de 0.1 seg, le tomaría al controlador 50 seg adaptarse a la variación intermedia. Respecto a la peor variación, se requieren aproximadamente 600 iteraciones para alcanzar la convergencia a valores óptimos, lo que le tomaría al controlador 60 seg adaptarse a la peor variación después de sucedida la

variación intermedia. La señal de PE usada corresponde a (133), la cual fue usada en el Ejemplo 6 (sección 4.3.2). Por lo anterior, existe la posibilidad de reducir la longitud N de la trayectoria sin afectar demasiado la medición del MSE. Sin embargo, se justifica el uso de $N = 11000$ en el Ejemplo 8 como una comparación directa con el Ejemplo 7.

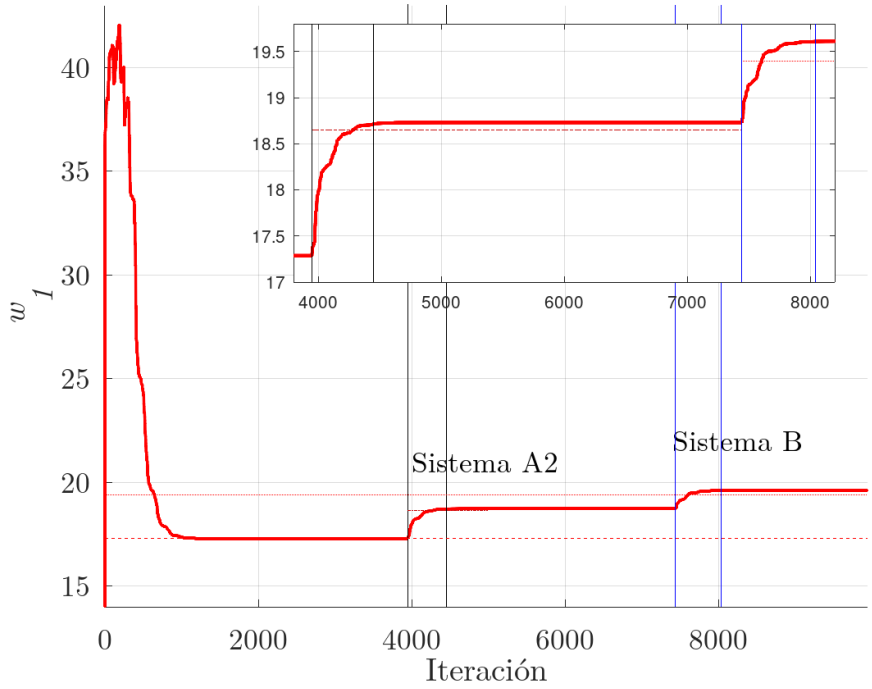

Figura 56. Convergencia del parámetro w_1 con dos variaciones usando el Algoritmo 5, con LS+RLS para la sintonización del crítico. Con los parámetros $N = 11000$, $\gamma = 0.999$, $n_{ls} = 7$, $\lambda = 0.99$, $\delta = 10^{-2}$ y $\Gamma(n_{ls})$ es escalada por 1000. Tiempo de ejecución ≈ 4.4748 segundos.

La Tabla 9 muestra los valores de los parámetros óptimos al usar la función `DLQR` y los estimados respecto a la peor variación. Considere que los resultados fueron obtenidos después de sucedida la variación intermedia.

Tabla 9. Comparación de valores óptimos y estimados del vector W y el vector K para la peor variación.

Parámetro	h_{11}	h_{12}	h_{13}	h_{22}	h_{23}	h_{33}	k_1	k_2	**MSE**
Óptimo	19.3944	−10.2127	10.4713	7.2004	−6.3574	8.5184	1.22955	−0.74649	0
Ejemplo 7	19.3665	−10.2679	10.4790	7.2707	−6.3904	8.5680	1.22304	−0.74584	0.06631
Ejemplo 8	19.6123	−10.3856	10.5772	7.3489	−6.4435	8.6067	1.22896	−0.74866	0.00217

Capítulo 5. Discusión

Dinámica parcialmente conocida: se implementaron los algoritmos de iteración de valor (Algoritmo 3) e iteración de política (Algoritmo 4) usando el método de aprendizaje por diferencias temporales en sistemas con comportamiento estable (masa-resorte-amortiguador) e inestable. Se realiza un entrenamiento episódico para un esquema que usa LS+RLS en la sintonización del crítico y GD en la sintonización del actor.

En el entrenamiento episódico, se usa la señal aperiódica como PE. Este esquema muestra características adecuadas para sistemas que pueden ser sometidos a entrenamientos repetitivos con una longitud corta de trayectoria, debido a que se puede interrumpir la señal de PE en al menos $2l$ muestras, obteniendo buenos resultados y evitando comprometer los elementos físicos del sistema, incluso en los sistemas inestables.

Se implementó el algoritmo de iteración de política usando el método de aprendizaje por diferencias temporales en entrenamiento continuo para el sistema estable, usando los esquemas LS+RLS y LS+GD en la sintonización del crítico, y con GD en la sintonización del actor (sección 4.2.5), como una comparación a los resultados obtenidos en el entrenamiento episódico. Aunque se logró la convergencia cercana a los valores óptimos, solo se logra usando una señal multisenoidal con decaimiento exponencial como PE. Además, la selección de parámetros iniciales para la sintonización del crítico y el actor es empíricamente más compleja que hacerlo con la metodología de entrenamiento episódico. No se logra la convergencia para el sistema inestable en el entrenamiento continuo.

La Tabla 5 muestra los resultados obtenidos usando el sistema estable. El mejor resultado se obtuvo con el método de LS+GD en la sintonización del crítico y GD en la sintonización del actor en entrenamiento en continuo, con un MSE de 0.55785 respecto a los valores óptimos. Los esquemas que usan entrenamiento episódico tienen un MSE de 17.032 usando iteración de valor y 46.471 usando iteración de política. Se puede mejorar el resultado de la convergencia incrementando el número de episodios de entrenamiento, que no necesariamente implica someter el sistema a una señal de PE prolongada. Además, solo se considera en la sintonización del actor el GD con un paso fijo como tasa de aprendizaje, existe la posibilidad de mejorar la convergencia si se agrega un paso variable como tasa de aprendizaje.

Dinámica totalmente desconocida: se implementó el algoritmo de iteración de política (Algoritmo 5) con Q-learning para el sistema con comportamiento estable en entrenamiento continuo, usando los esquemas LS+RLS y LS+GD en la sintonización del crítico.

Se implementó el algoritmo de iteración de política con Q-learning en entrenamiento continuo usando solo GD y RLS en la sintonización del crítico, para una comparación de los resultados.

La Tabla 7 muestra los resultados obtenidos con los escenarios descritos anteriormente. Los mejores resultados se obtienen en los métodos que implican el uso de los esquemas de LS+RLS y RLS, con un MSE de 0.0006349 respecto a los valores óptimos. En el Ejemplo 6 (sección 4.3.2) tenemos una mejora de aproximadamente 200 iteraciones en comparación al esquema que usa solo RLS en la sintonización del crítico (Figura 54). Lo que se traduce en menor tiempo para alcanzar la convergencia, es decir, aproximadamente $20\ seg$ más rápido, si se considera que el periodo de muestreo es de $0.1\ seg$. Además, el esquema del Ejemplo 5 (sección 4.3.1) es mucho mejor que el esquema que usa solo GD (Figura 53) en la sintonización del crítico, con un MSE de 0.01087 respecto a los valores óptimos.

La prueba de adaptabilidad a las variaciones se desarrolló sobre el sistema con comportamiento estable considerando dos variaciones en la etapa de entrenamiento (sección 4.4). El esquema usado en el Ejemplo 8 (sección 4.4.2) logra la convergencia para la peor variación con un MSE de 0.00217 respecto a los valores óptimos, y le toma aproximadamente 600 iteraciones ($60\ seg$) adaptarse a la peor variación. Mientras que el esquema usado en el Ejemplo 7 (sección 4.4.1) con un MSE de 0.06631, necesita de aproximadamente 1500 iteraciones ($2.5\ min$) para adaptarse a la peor variación, en ambos casos después de sucedida la variación intermedia.

El uso de la señal multisenoidal con decaimiento exponencial presenta mejores resultados en la convergencia de los parámetros hacia valores óptimos, sin embargo, se ve limitada cuando las variaciones en la matriz de dinámica A suceden cerca del final de la etapa de entrenamiento, como se observa en las últimas iteraciones de la Figura 56, debido a que la mayor dinámica de la señal de PE es generada al inicio.

Aún existen aspectos limitados en autonomía que requieren la intervención de un experto:

- conocimiento de al menos el orden del sistema y la estabilidad al aplicar la política de control $h_0(x)$ en lazo cerrado,
- diseño de la política de control $h_0(x)$ estabilizante.

Capítulo 6. Conclusiones

Este capítulo presenta las conclusiones del proyecto de tesis, resaltando las mejoras en el uso del aprendizaje por refuerzo en el diseño de controladores para sistemas lineales. Además, se exponen las limitaciones del trabajo y las aportaciones a la metodología propuesta por Lewis et al. (2012a). Para finalizar el capítulo, se proponen algunas perspectivas de investigación, tomando como punto de partida la experiencia adquirida en esta tesis.

6.1. Conclusiones generales

Los algoritmos de aprendizaje por refuerzo son una alternativa para el diseño de controladores cuando se tenga conocimiento parcial o desconocimiento total del modelo que describe la dinámica del sistema. Particularmente, el algoritmo Q-learning es adecuado para un esquema que busca tener poca dependencia del modelo del sistema. Además, la forma de actualizar este algoritmo usando iteración de política, permite tener menor número de parámetros a sintonizar de manera arbitraria, necesarios al inicio del algoritmo. El uso de aproximadores paramétricos son una forma conveniente de representar las Q-funciones o funciones de valor V debido a que la estimación de sus parámetros se combinan de forma adecuada con técnicas de identificación de sistemas. Esto permite requerir menor intervención de un experto en el diseño de un controlador que implemente leyes de control óptimas con capacidad adaptativa a las variaciones. Para esta tesis se consideraron solo variaciones que puedan ocurrir en la matriz de dinámica A durante la etapa de entrenamiento. Sin embargo, el desconocimiento del modelo del sistema no es completo, debido a que se requieren conocer importantes características del sistema, como lo son; el orden del sistema y el tipo de comportamiento en lazo cerrado al aplicar una política de control $h_0(x)$. Es necesario que la política de control aplicada no inestabilice al sistema durante el entrenamiento, para evitar daños en la planta o en el sistema completo, además, el orden del sistema es necesario en el diseño de la estructura del aproximador paramétrico a usar.

La mayor dificultad en la implementación de los algoritmos se observa en la etapa de estimación de parámetros para el vector de pesos W, debido a que las técnicas de identificación usadas requieren mantener excitado el vector de regresión durante el entrenamiento. Sin embargo, el uso de LS+RLS y señales multisenoidales con decaimiento exponencial aseguraron mejores resultados en la convergencia a valores óptimos, reduciendo el número de iteraciones considerablemente, incluso cuando hay variaciones. Lo anterior es ideal para tareas en tiempo real donde se requiere menor tiempo para la convergencia y

rápida adaptación a variaciones.

6.2. Aportaciones realizadas

Sobre la implementación de los algoritmos propuestos en Lewis et al. (2012a) para un sistema lineal estable:

- Se utilizó una señal aperiódica como señal de PE, que resulta fácil de sintonizar debido a que requiere menor número de argumentos en la función, en comparación con señales multisenoidales. Sin embargo, no presentó convergencia para los ejemplos con dinámica parcialmente conocida en entrenamiento continuo.
- Implementación en entrenamiento episódico para dinámica parcialmente conocida.
- Estimación del vector de pesos para el aproximador paramétrico del crítico usando métodos de estimación de parámetros que eviten la pérdida de rango en la matriz de covarianza:
  - LS+RLS.
  - LS+GD.

El alcance para los sistemas inestables es similar a los puntos anteriores, pero depende forzosamente del correcto diseño de la política de control $h_0(x)$ estabilizante en lazo cerrado.

Este proyecto de tesis es un primer esfuerzo por parte del departamento de electrónica y telecomunicaciones del CICESE por usar los conceptos de RL en el diseño de controladores automáticos, haciendo un trabajo exploratorio sobre el alcance de algoritmos propuestos por otros autores.

6.3. Trabajo futuro

- **Implementación en una planta física:** la implementación de los algoritmos vistos en esta tesis en una planta física puede tener retos en:
  - la resolución de las ecuaciones en tiempo real,

  - el diseño de señales de PE adecuadas al tipo de sistema.

- **Movimientos colectivos de sistemas multiagente:** el diseño de controladores para sistemas multiagente basado en algoritmos de RL que tengan movimientos colectivos con seguimiento de trayectoria o con evasión de obstáculos.

- **Realimentación de la salida:** explorar el diseño de los controladores basados en RL usando la realimentación de la salida para la regulación o el seguimiento de trayectoria, debido a que en los sistemas reales no siempre se puede tener acceso a todos los estados, pero es posible considerar la realimentación de la salida para el diseño de una ley de control.

- **Excitación persistente no necesaria:** estimar los parámetros desde un problema de regresión sin la necesidad de una señal de PE como en Korotina et al. (2021). Esto simplificaría el uso de los algoritmos de RL para tareas de control adaptativo con mayor autonomía.

- **Funciones base:** explorar mejoras al usar funciones base Fourier o funciones no lineales como funciones de activación.

Literatura citada

Al-Tamimi, A., Lewis, F., & Abu-Khalaf, M. (2008). Discrete-Time Nonlinear HJB Solution Using Approximate Dynamic Programming: Convergence Proof. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 38:943 – 949. https://doi.org/10.1109/TSMCB.2008.926614.

Al-Tamimi, A. A. (2007). *Discrete Time Control Algorithms and Adaptive Intelligent Systems Designs*. Tesis de doctorado, The University of Texas, USA.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.

Bertsekas, D. (2019). *Reinforcement Learning and Optimal Control*. Athena Scientific Optimization and Computation Series. Athena Scientific.

Bradtke, S., Ydstie, B., & Barto, A. (1994). Adaptive linear quadratic control using policy iteration. In *Proceedings of 1994 American Control Conference - ACC '94*, 3475–3479, Baltimore, MD, USA. IEEE. https://doi.org/10.1109/ACC.1994.735224.

Brunton, S. L. & Kutz, J. N. (2019). *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press.

Busoniu, L., Babuska, R., De Schutter, B., & Ernst, D. (2017). *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press.

Cheng, Y., Feng, H., & Wang, X. (2013). Efficient data use in incremental actor–critic algorithms. *Neurocomputing*, 116:346–354. https://doi.org/10.1016/j.neucom.2011.11.034.

Díaz, H., Armesto, L., & Sala, A. (2019). Metodología de programación dinámica aproximada para control óptimo basada en datos. *Revista Iberoamericana de Automática e Informática industrial*, 16(3):273. https://doi.org/10.4995/riai.2019.10379.

Fernandez-Gauna, B., Osa, J. L., & Graña, M. (2018). Experiments of conditioned reinforcement learning in continuous space control tasks. *Neurocomputing*, 271:38–47. https://doi.org/https://doi.org/10.1016/j.neucom.2016.08.155.

Franklin, G. F., Powell, D. J., & Workman, M. L. (1998). *Digital Control of Dynamic Systems*, (3a ed.). Ellis-Kagle Press.

Grzedzinski, K. & Trodden, P. (2018). Learning MPC: System stability and convergent identification under bounded modelling error. In *2018 Australian and New Zealand Control Conference*, 125–130, Melbourne, VIC. IEEE.

Gym, O. (2022). Openai. https://www.gymlibrary.dev/.

Görges, D. (2019). Distributed Adaptive Linear Quadratic Control using Distributed Reinforcement Learning. *IFAC-PapersOnLine*, 52(11):218–223. https://doi.org/https://doi.org/10.1016/j.ifacol.2019.09.144.

Hao, J., Yang, T., Tang, H., Bai, C., Liu, J., Meng, Z., Liu, P., & Wang, Z. (2022). Exploration in Deep Reinforcement Learning: From Single-Agent to Multiagent Domain. *IEEE Transactions on Neural Networks and Learning Systems*, 1–21. https://doi.org/10.1109/TNNLS.2023.3236361.

Khalil, H. K. (2015). *Nonlinear control*. Pearson.

Kiumarsi, B., Lewis, F. L., Modares, H., Karimpour, A., & Naghibi-Sistani, M.-B. (2014). Reinforcement Q-learning for optimal tracking control of linear discrete-time systems with unknown dynamics. *Automatica*, (4):1167–1175. https://doi.org/https://doi.org/10.1016/j.automatica.2014.02.015.

Kiumarsi, B., Vamvoudakis, K. G., Modares, H., & Lewis, F. L. (2018). Optimal and Autonomous Control Using Reinforcement Learning: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, 29(6):2042–2062. https://doi.org/10.1109/TNNLS.2017.2773458.

Kiumarsi-Khomartash, B., Lewis, F. L., Naghibi-Sistani, M.-B., & Karimpour, A. (2013). Optimal Tracking Control for Linear Discrete-time Systems Using Reinforcement Learning. In *52nd IEEE Conference on Decision and Control*, 3845–3850. IEEE.

Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274. https://doi.org/10.1177/0278364913495721.

Kober, J. & Peters, J. (2012). *Reinforcement Learning - State-of-the-Art*, (pp. 579–610). Adaptation, Learning, and Optimization. Springer.

Koenig, S. & Liu, Y. (2002). The interaction of representations and planning objectives for decision-theoretic planning tasks. *Journal of Experimental & Theoretical Artificial Intelligence*, 14(4):303–326. https://doi.org/10.1080/09528130210151831.

Korotina, M., Romero, J. G., Aranovskiy, S., Bobtsov, A., & Ortega, R. (2021). Persistent Excitation is Unnecessary for On-line Exponential Parameter Estimation: A New Algorithm that Overcomes this Obstacle. arXiv.

Landelius, T. & Knutsson, H. (1996). Greedy Adaptive Critics for LQR Problems: Convergence Proofs. *Department of Electrical Engineering, Computer Vision Laboratory*.

Lewis, F. L. & Vamvoudakis, K. G. (2011). Reinforcement Learning for Partially Observable Dynamic Processes: Adaptive Dynamic Programming Using Measured Output Data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(1):14–25. https://doi.org/10.1109/TSMCB.2010.2043839.

Lewis, F. L. & Vrabie, D. (2009). Reinforcement Learning and Adaptive Dynamic Programming for Feedback Control. *IEEE Circuits and Systems Magazine*, (3):32–50. https://doi.org/10.1109/MCAS.2009.933854.

Lewis, F. L., Vrabie, D., & Vamboudakis, K. (2012a). Reinforcement Learning and Feedback Control: Using Natural Decision Methods to Design Optimal Adaptive Controllers. *IEEE Control Systems*, (6):76–105. https://doi.org/10.1109/MCS.2012.2214134.

Lewis, F. L., Vrabie, D. L., & Syrmos, V. L. (2012b). *Optimal Control*, (3a ed.). John Wiley & Sons.

Ljung, L. (1999). *System identification: Theory for the user*, (2a ed.). Prentice Hall PTR.

Marafioti, G., Bitmead, R. R., & Hovd, M. (2010). Persistently Exciting Model Predictive Control Using FIR Models. Norwegian University Of Science and Technology.

Miguel, J. M. (2023). Códigos. https://github.com/alxmz2/labcontrol/tree/master.

Miranda, C. (2012). *Sistemas de control continuos y discretos*. Paraninfo.

Miroslav, K. & Zhong-Hua, L. (1997). Optimal Design of Adaptative Tracking Controllers for Non-linear Systems. *Automatica*, 33(8):1459–1473. https://doi.org/https://doi.org/10.1016/S0005-1098(97)00072-1.

Moerland, T. M., Broekens, J., Plaat, A., & Jonker, C. M. (2022). Model-based Reinforcement Learning: A Survey. arXiv.

Montero, P. E. (2014). *Aprendizaje por refuerzo en espacios continuos*. Tesis de doctorado, Universidad de Valencia, España.

Muller, A. C. & Guido, S. (2017). *Introduction to Machine Learning with Python*. O'Reilly Media, Incorporated.

Murillo, R. (1997). El teorema de aproximación de Weierstrass. https://miscelaneamatematica.org/download/tbl_articulos.pdf2.9da3c5dad16cc1fe.6d7572696c6c6f2e706466.pdf.

Neto, J. V. d. F., Ferreira, E. F., & Rego, P. H. M. (2013). Online Optimal DLQR-DFIG Control System Design via Recursive Least-Square Approach and State Heuristic Dynamic Programming for Approximate Solution of the HJB Equation. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, 3174–3179, Manchester. IEEE. https://doi.org/10.1109/SMC.2013.541.

Ogata, K. (2010). *Ingeniería de control moderna*, (5ta ed.). Pearson Education.

Perrusquia, A., Yu, W., & Soria, A. (2019). Large space dimension Reinforcement Learning for Robot Position/Force Discrete Control. In *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, 91–96, Paris, France. IEEE. https://doi.org/10.1109/CoDIT.2019.8820575.

Perrusquía, A., Yu, W., & Li, X. (2021). Multi-agent reinforcement learning for redundant robot control in task-space. *International Journal of Machine Learning and Cybernetics*, 12(1):231–241. https://doi.org/10.1007/s13042-020-01167-7.

Puriel Gil, G., Yu, W., & Sossa, H. (2019). Reinforcement Learning Compensation based PD Control for a Double Inverted Pendulum. *IEEE Latin America Transactions*, 17(02):323–329. https://doi.org/10.1109/TLA.2019.8863179.

Pygame (2023). Módulo pygame. https://www.pygame.org/news.

Raschka, S. (2016). *Python Machine Learning: Unlock deeper insights into machine learning with this vital guide to cutting-edge predictive analytics*. Community experience distilled. Packt Publishing open source.

Recht, B. (2018). Optimization Perspectives on Learning to Control. https://people.eecs.berkeley.edu/~brecht/l2c-icml2018/.

Recht, B. (2019). A Tour of Reinforcement Learning: The View from Continuous Control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2:253–279. https://doi.org/10.1146/annurev-control-053018-023825.

Rubio, F. & Sánchez, M. (1996). *Control adaptativo y robusto*. Colección Ingeniería. Universidad de Sevilla.

Sanusi, I., Mills, A., Dodd, T., & Konstantopoulos, G. (2020). Online optimal and adaptive integral tracking control for varying discrete-time systems using reinforcement learning. *International Journal of Adaptive Control and Signal Processing*, 34(8):971–991. https://doi.org/10.1002/acs.3115.

Siraskar, R. (2021). Reinforcement learning for control of valves. *Machine Learning with Applications*, 4. https://doi.org/10.1016/j.mlwa.2021.100030.

Skiena, S. S. (2008). *The Algorithm Design Manual*. Springer London.

Strang, G. (2019). *Linear Algebra and learning from data*, (2a ed.). Wellesley-Cambridge Press.

Sutton, R. S. (1988). Learning to Predict by the Methods of Temporal Differences. *Machine Learning*, 3:9–44.

Sutton, R. S. & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*, (2a ed.). Adaptive Computation and Machine Learning Series. The MIT Press.

Tangirala, A. K. (2018). *Principles of System Identification: Theory and Practice*. CRC Press.

Tedrake, R. (2009). Underactuated Robotics - Analytical Optimal Control with the Hamilton-Jacobi-Bellman Sufficiency Theorem. https://ocw.mit.edu/courses/6-832-underactuated-robotics-spring-2009/resources/mit6_832s09_read_ch10/.

Vamvoudakis, K. G., Vrabie, D., & Lewis, F. L. (2014). Online adaptive algorithm for optimal control with integral reinforcement learning. *International Journal of Robust and Nonlinear Control*, 24(17):2686–2710. https://doi.org/10.1002/rnc.3018.

Vrabie, D. L., Vamvoudakis, K. G., & Lewis, F. L. (2013). *Optimal Adaptive Control and Differential Games by Reinforcement Learning Principles*. Number 81 in IET Control Engineering Series. The Institution of Engineering and Technology.

Wang, F.-Y., Zhang, H., & Liu, D. (2009). Adaptive Dynamic Programming: An Introduction. *IEEE Computational Intelligence Magazine*, 4(2):39–47. https://doi.org/10.1109/MCI.2009.932261.

Wang, Z. & Hong, T. (2020). Reinforcement learning for building controls: The opportunities and challenges. *Applied Energy*, 269:115036. https://doi.org/10.1016/j.apenergy.2020.115036.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. Tesis de doctorado, King's College, Cambridge, UK.

Watkins, C. J. C. H. & Dayan, P. (1992). Q-Learning. *Machine Learning*, 8(3):279–292. https://doi.org/10.1007/BF00992698.

Werbos, P. J. (1987). Building and Understanding Adaptive Systems: A Statistical/Numerical Approach to Factory Automation and Brain Research. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(1):7–20. https://doi.org/10.1109/TSMC.1987.289329.

Werbos, P. J., Miller, W. T., & Sutton, R. S. (1991). *A Menu of Designs for Reinforcement Learning Over Time*, 67–95. The MIT Press.

Yu, W. & Perrusquía, A. (2021). *Human-Robot Interaction Control Using Reinforcement Learning*. IEEE.

Yun, W. J., Park, S., Kim, J., Shin, M., Jung, S., Mohaisen, D. A., & Kim, J.-H. (2022). Cooperative Multi-Agent Deep Reinforcement Learning for Reliable Surveillance via Autonomous Multi-UAV Control. *IEEE Transactions on Industrial Informatics*, 18(10):7086–7096. https://doi.org/10.1109/TII.2022.3143175.

Zheng Chen & Jagannathan, S. (2008). Generalized Hamilton–Jacobi–Bellman Formulation Based Neural Network Control of Affine Nonlinear Discrete-Time Systems. *IEEE Transactions on Neural Networks*, 19(1):90–106. https://doi.org/10.1109/TNN.2007.900227.

Anexos

Anexo A

Contiene los códigos diseñados para los algoritmos de RL vistos el Capítulo 4, sección 4.2, correspondientes a dinámica parcialmente conocida.

A.1. Código para generar señal chirp

Código diseñado por el Dr. Luis Alejandro Márquez Martínez (lmarquez@cicese.edu.mx), fue usado para la generación de señales aperiódicas.

Archivo: michirp.m

```
function [u,wk]=michirp(wini,wfin,N,T)
  for i=0:(N-1),
      wk(i+1)=wini+(i-1)/N*(wfin-wini);
      w(i+1)=sat(10*i/N)*sat((N-i)/(0.1*N));
      u(i+1,1)=i*T;
      u(i+1,2)=w(i+1)*sin(wk(i+1)*i*T);
  end
end
```

Archivo: sat.m

```
function xs=sat(x)
  if abs(x)<1
          xs=x;
      else
          xs=sign(x);
  end
end
```

A.2. Iteración de valor - usando la VFA con diferencias temporales para el problema del LQR

Este es el único código que incluye las líneas correspondientes a las gráficas, por simplicidad, se evitan las líneas en los demás códigos del Anexo A. Se pueden consultar los códigos completos en Miguel (2023).

```
close all;
clear all;
clc;
set(0, "defaulttextfontname", "Arial");
set(0, "defaultlinelinewidth", 1.5);

%Algoritmo de iteración de valor usando TD en entrenamiento episódico para
%dinámica parcialmente conocida de un sistema estable sin variaciones.

%Autor: Jesús Miguel Martínez, email: <jmmiguel@cicese.edu.mx>
%Asesores: Carlos Alberto Brizuela Rodríguez, Luis Alejandro Márquez
   Martínez.

%Ecuaciones del sistema en espacio de
   estados-----------------------------------

A = [1.8980 -0.9048;1 0];
B = [1;0];
C = [0.0048 0.0047];

%Matrices de ponderación del índice de
   desempeño---------------------------------

Q = diag ( [ 1 1 ] );
R = 2;
polos_la = eig(A)                          %Polos de la matriz A en lazo
   abierto

%Diseño de política
   inicial-----------------------------------------------------

P_e = [1 0;0 1];
Pp = P_e;
K = -inv(R + B'*P_e*B)*B'*P_e*A              %Vector control realimentado
   inicial
K_1 = -K(1, 1)
K_2 = -K(1, 2)
polos_lc = eig(A + B*K)

%Parámetros
   iniciales---------------------------------------------------------------

N = 450;                                   %Tiempo total N
gamma = 0.999;                             %Factor de descuento
p = 3;                                     %Términos independientes de P_e
num_ep = 12;                               %Número de episodios
n_ls = 7;                                  %Tamaño de lote para LS
W = [P_e(1,1);P_e(1,2);P_e(2,2)];          %Pesos de W_0 (estimación inicial)
W_ant = [P_e(1,1);P_e(1,2);P_e(2,2)];      %Pesos de W_1 (predicción inicial)
l = 0;
con = 0;
ep = 0;
datos = 0;

%Parámetros del
   crítico-----------------------------------------------------------

f_olvido = 0.99;
a = 1 - f_olvido;
Iden = eye(p);
condicion_e_k = 10e-6;                     %Condición del error de TD
convergencia = 1;                          %Convergencia = 0
                                           %solo cuando se cumple la condición
                                           %del error
b = 1;

```

```
%Parámetros del actor----------------------------------------------------------

beta = 0.0102;                                  %Tasa de aprendizaje del DG

%Señal de excitación-----------------------------------------------------------

[inp,wk] = michirp(0.005,345000,N+1,0.1); %Frecuencias angulares inicial y
                                          %final para la función Chirp

%Condiciones iniciales---------------------------------------------------------

x(:,1) = [0.5;-0.4];

%Inicio del algoritmo----------------------------------------------------------

while ep < num_ep
  datos = 0;
  ep = ep + 1;
  convergencia = 1;
  dd = 1;
  P_ant = P_e;

%Crítico-----------------------------------------------------------------------

  for i = 1:N
    if (convergencia == 1)
      datos = datos + 1;
      final_exc = datos;
      if ep >= 1
        u = K*x(:,i) + b*inp(i,2)*0.102;
      end
      x(:,i+1) = A*x(:,i) + B*u;
      y(i,:) =  C*x(:,i);
      r(i,:) = x(:,i)'*Q*x(:,i) + u'*R*u;
      phi(:,i) = [x(1,i)^2; 2*x(1,i)*x(2,i) ; x(2,i)^2];
      phi1(:,i) = [x(1,i+1)^2; 2*x(1,i+1)*x(2,i+1) ; x(2,i+1)^2];
      Y(i,:) = r(i,:) + gamma*W_ant'*phi1(:,i);

      if i == n_ls
        for k = 1:n_ls
          phi_(:,k) = phi(:,k);
          Y_(k,:) = Y(k,:);
        endfor
        %LS
        C_ls = phi_*phi_';
        W_ant = inv(C_ls)*phi_*Y_;
        theta_wls(:,n_ls) = W_ant;
        P_n = phi_*eye(n_ls)*phi_';
      end

      if i > n_ls
        con = con + 1;
        k = i;
        phi_k = phi(:,k);
        r_k = r(k,:);
        phi1_k = phi1(:,k);
        %RLS
        e_rls(:,k) = -W'*phi_k + r_k + gamma*W_ant'*phi1_k;
        e_k = e_rls(:,k);
        if (abs(e_k) < condicion_e_k)
          convergencia = 1;
          vec_datos(1,dd) = datos;
          b = 0;
```

```
122             dd = dd + 1;
123             final_exc = min(vec_datos);
124           end
125           L1(:,k) = 1/f_olvido*P_n*phi_k*inv(1/a + phi_k'*1/f_olvido*P_n*phi_k);
126           L = L1(:, k);
127           theta_wls(:, k) = W_ant + L*e_k;
128           P_n = 1/f_olvido*(Iden - L*phi_k')*P_n;
129           W = theta_wls(:, k);
130         end
131       end
132     endfor
133
134     final_exc
135     W_ant = W;
136     P_e = [theta_wls(1,k) theta_wls(2,k); theta_wls(2,k) theta_wls(3,k)];
137
138     error_P(:,ep) = norm(P_e - P_ant);
139     length(error_P);
140
141   %Actor-------------------------------------------------------------------------
142
143     U_j = K';
144     for i = 1:datos
145       sigma_k = x(:, i);
146       dphi(:,:) = [2*x(1,i+1) 0; 2*x(2,i+1) 2*x(1,i+1); 0 2*x(2,i+1)];
147       diferencia = beta*sigma_k*(2*R*U_j'*sigma_k + gamma*B'*dphi'*W)';
148       U_j1  = U_j - diferencia;
149       K_es = U_j1;
150       U_j = K_es; %K_es es realimentada
151     endfor
152   %-------------------------------------------------------------------------------
153     error_K(:,ep) = norm(K' - K_es);
154     K_e = -K_es';
155     K = -K_e;
156
157     %Acomodar los valores de K para para gráficar
158     for c = 1:2
159       K1(1,c + l) = K_e(1,c);
160     end
161
162     %Acomodar los valores de P para para gráficar
163     for f = 1:2
164       for c = 1:2
165       P1(f,c + l) = P_e(f,c);
166       end
167     end
168
169     l = l + 2;
170
171   endwhile
172
173   K_e
174   P_e
175   [ K0 , P0 ] = dlqr( A , B , Q , R )
176
177   % Las siguientes líneas son para el armado de los vectores K_e y
        P_e------------
178   %
        K_e---------------------------------------------------------------------------
179
180   g1 = 1;
181   k11 = zeros(1,ep + 1);
182   k11(:, 1) = K_1;
183   for j = 2:ep + 1
184     k11(1, j) = K1(1,g1);
185     g1 = g1 + 2;
186   end
187
```

```
g2 = 2;
k12 = zeros(1,ep + 1);
k12(:, 1) = K_2;
for j = 2: ep + 1;
  k12(1, j) = K1(1,g2);
  g2 = g2 + 2;
end

%   P_e--------------------------------------------------------------------------

l1 = 1;
p11 = zeros(1,ep + 1);
p11(:,1) = Pp(1,1); %P_11 ini
for j = 2:ep + 1;
  p11(1, j) = P1(1,l1);
  l1 = l1 + 2;
end

l2 =2;
p12 = zeros(1,ep + 1);
p12(:,1) = Pp(1,2);
for j = 2: ep + 1;
  p12(1, j) = P1(1,l2);
  l2 = l2 + 2;
end

l3 = 1;
p21 = zeros(1,ep + 1);
p21(:,1) = Pp(2,1);
for j = 2:ep + 1;
  p21(1, j) = P1(2,l3);
  l3 = l3 + 2;
end

l4 = 2;
p22 = zeros(1,ep + 1);
p22(:,1) = Pp(2,2);
for j = 2: ep + 1;
  p22(1, j) = P1(2,l4);
  l4 = l4 + 2;
end

%Colores de
   gráficas-----------------------------------------------------------------

str = '#3445DB'; %Azul
color1 = sscanf(str(2:end),'%2x%2x%2x',[1 3])/255;
str = '#DF8908'; %Naranja
color2 = sscanf(str(2:end),'%2x%2x%2x',[1 3])/255;
str = '#32B138'; %Verde
color3 = sscanf(str(2:end),'%2x%2x%2x',[1 3])/255;
str = '#C80505'; %Rojo
color4 = sscanf(str(2:end),'%2x%2x%2x',[1 3])/255;
str = '#BB05C8'; %Purpura
color5 = sscanf(str(2:end),'%2x%2x%2x',[1 3])/255;
str = '#1970CA'; %Azul
color6 = sscanf(str(2:end),'%2x%2x%2x',[1 3])/255;

%Gráficas---------------------------------------------------------------------

figure(2)
hold on;

plot( (0:ep), p11,"marker", "o", "markerEdgeColor", color1, ...
     "markersize", 4, "linewidth", 2.5, "color", color1);

```

```
plot( (0:ep), p12, "marker", "o", "markerEdgeColor", color2, ...
"markersize", 4, "linewidth", 2.5, "color", color2);

plot( (0:ep), p22, "marker", "o", "markerEdgeColor", color3, ...
"markersize", 4, "linewidth", 2.5, "color", color3);

xlabel("Episodio");
ylabel("Parámetros de W");
set(gca, 'FontSize', 17);
legend({"W_{1}","W_{2}", "W_{3}"}, "location", "bestoutside");

line([0 num_ep], [ P0(1,1)  P0(1,1)], "linestyle", "--", "color", color1);
line([0 num_ep], [P0(1,2) P0(1,2)], "linestyle", "--", "color", color2);
line([0 num_ep], [P0(2,2) P0(2,2)], "linestyle", "--", "color", color3);
xlim ([0,num_ep]);
grid on;

figure(3)
hold on;

plot( (0:ep), k11,"marker", "o", "markerEdgeColor", color4, ...
     "markersize", 4, "linewidth", 2.5, "color", color4);

plot( (0:ep), k12, "marker", "o", "markerEdgeColor", "b", ...
"markersize", 4, "linewidth", 2.5, "color","b");

xlabel("Episodio");
ylabel("Parámetros de K");
set(gca, 'FontSize', 17);
legend({"K_{1}","K_{2}"},  "location", "east");

line([0 num_ep], [ K0(1,1) K0(1,1) ], "linestyle", "--", "color", color4);
line([0 num_ep], [K0(1,2) K0(1,2) ], "linestyle", "--", "color", "b");
xlim ([0,num_ep]);
grid on;

%Normas del
   error----------------------------------------------------------------

figure(4)
hold on;

subplot(2,1,1)
plot( (1:ep), error_P,"marker", "o", "markerEdgeColor", color5, ...
     "markersize", 4, "linewidth", 2.5, "color", color5);

xlabel("Episodio");
ylabel("||P_{j+1} - P_{j}||");
set(gca, 'FontSize', 17);
grid on;

subplot(2,1,2)
plot( (1:ep), error_K,"marker", "o", "markerEdgeColor", color6, ...
     "markersize", 4, "linewidth", 2.5, "color", color6);

xlabel("Episodio");
ylabel("||K_{j+1} - K_{j}||");
set(gca, 'FontSize', 17);
grid on;
```

A.3. Iteración de valor - usando la VFA con diferencias temporales para el problema del LQR (sistema inestable)

```
close all;
clear all;
clc;
set(0, "defaulttextfontname", "Arial");
set(0, "defaultlinelinewidth", 1.5);

%Ecuaciones del sistema en espacio de estados-----------------------------------
%Inestable
A = [-1 2;2.2 1.7];
B = [2;1.6];
C = [1 2];

%Matrices de ponderación del índice de desempeño--------------------------------

Q = diag ( [ 1 1 ] );
R = 2;
polos_la = eig(A)                          %Polos de la matriz A en lazo abierto

%Diseño de política inicial-----------------------------------------------------

P_e = [10, 0; 0, 0]
Pp = P_e;
K = -inv(R + B'*P_e*B)*B'*P_e*A             %Vector control realimentado inicial
K_1 = -K(1, 1)
K_2 = -K(1, 2)
polos_lc = eig(A + B*K)

%Parámetros iniciales-----------------------------------------------------------

N = 500;                                   %Tiempo total N
gamma = 0.999;                             %Factor de descuento
p = 3;                                     %Términos independientes de P_e
num_ep = 12;                               %Número de episodios
n_ls = 7;                                  %Tamaño de lote para LS
W = [P_e(1,1);P_e(1,2);P_e(2,2)];          %Pesos de W_0 (estimación inicial)
W_ant = [P_e(1,1);P_e(1,2);P_e(2,2)];      %Pesos de W_1 (predicción inicial)
l = 0;
con = 0;
ep = 0;
datos = 0;

%Parámetros del crítico---------------------------------------------------------

f_olvido = 0.99;
a = 1 - f_olvido;
Iden = eye(p);
condicion_e_k = 10e-4;                     %Condición del error de TD
convergencia = 1;                          %Convergencia = 0
                                           %solo cuando se cumple la condición
                                           %del error
b = 1;

%Parámetros del actor-----------------------------------------------------------
```

```
54
55  beta = 0.00202;                              %Tasa de aprendizaje del DG
56
57  %Señal de
       excitación-------------------------------------------------------------
58
59  [inp,wk] = michirp(0.005,3450000,N+1,0.1);
60
61  %Condiciones
       iniciales--------------------------------------------------------------
62
63  x(:,1) = [0.5;-0.4];
64
65  %Inicio del
       algoritmo--------------------------------------------------------------
66
67  while ep < num_ep
68    datos = 0;
69    ep = ep + 1
70    convergencia = 1;
71    dd = 1;
72    P_ant = P_e;
73
74  %Crítico-----------------------------------------------------------------------
75
76    for i = 1:N
77      if (convergencia == 1)
78        datos = datos + 1;
79        final_exc = datos;
80        %La señal de PE solo se considera para el primer episodio
81        if ep == 1
82          u = K*x(:,i) + b*inp(i,2)*0.102;
83        end
84        if ep > 1
85          u = K*x(:,i);
86        end
87        x(:,i+1) = A*x(:,i) + B*u;
88        y(i,:) =  C*x(:,i);
89        r(i,:) = x(:,i)'*Q*x(:,i) + u'*R*u;
90        phi(:,i) = [x(1,i)^2; 2*x(1,i)*x(2,i) ; x(2,i)^2];
91        phi1(:,i) = [x(1,i+1)^2; 2*x(1,i+1)*x(2,i+1) ; x(2,i+1)^2];
92        Y(i,:) = r(i,:) + gamma*W_ant'*phi1(:,i);
93
94        if i == n_ls
95          for k = 1:n_ls
96            phi_(:,k) = phi(:,k);
97            Y_(k,:) = Y(k,:);
98          endfor
99          %LS
100         C_ls = phi_*phi_';
101         W_ant = inv(C_ls)*phi_*Y_;
102         theta_wls(:,n_ls) = W_ant;
103         P_n = phi_*eye(n_ls)*phi_';
104       end
105
106       if i > n_ls
107         con = con + 1;
108         k = i;
109         phi_k = phi(:,k);
110         r_k = r(k,:);
111         phi1_k = phi1(:,k);
112         %RLS
113         e_rls(:,k) = -W'*phi_k + r_k + gamma*W_ant'*phi1_k;
114         e_k = e_rls(:,k);
115         if (abs(e_k) < condicion_e_k)
116           convergencia = 1;
117           vec_datos(1,dd) = datos;
```

```
            b = 0;
            dd = dd + 1;
            final_exc = min(vec_datos);
          end
          L1(:,k) = 1/f_olvido*P_n*phi_k*inv(1/a + phi_k'*1/f_olvido*P_n*phi_k);
          L = L1(:, k);
          theta_wls(:, k) = W_ant + L*e_k;
          P_n = 1/f_olvido*(Iden - L*phi_k')*P_n;
          W = theta_wls(:, k);
        end
      end
    endfor

    final_exc
    W_ant = W;
    P_e = [theta_wls(1,k) theta_wls(2,k); theta_wls(2,k) theta_wls(3,k)];

    error_P(:,ep) = norm(P_e - P_ant);
    length(error_P);

  %Actor-------------------------------------------------------------------------

    U_j = K';
    for i = 1:datos
      sigma_k = x(:, i);
      dphi(:,:) = [2*x(1,i+1) 0; 2*x(2,i+1) 2*x(1,i+1); 0 2*x(2,i+1)];
      diferencia = beta*sigma_k*(2*R*U_j'*sigma_k + gamma*B'*dphi'*W)';
      U_j1  = U_j - diferencia;
      K_es = U_j1;
      U_j = K_es; %K_es ya es realimentada
    endfor
  %-------------------------------------------------------------------------------
    error_K(:,ep) = norm(K' - K_es);
    K_e = -K_es';
    K = -K_e;

    %Acomodar los valores de K para las gráficas
    for c = 1:2
      K1(1,c + l) = K_e(1,c);
    end

    %Acomodar los valores de P para las gráficas
    for f = 1:2
      for c = 1:2
      P1(f,c + l) = P_e(f,c);
      end
    end

    l = l + 2;

  endwhile

  K_e
  P_e
  [ K0 , P0 ] = dlqr( A , B , Q , R )
```

A.4. Iteración de política - usando la VFA con diferencias temporales para el problema del LQR

```
close all;
clear all;
clc;
set(0, "defaulttextfontname", "Arial");
set(0, "defaultlinelinewidth", 1.5);

%Algoritmo de iteración de política usando TD en entrenamiento episódico para
%dinámica parcialmente conocida de un sistema estable sin variaciones.

%Ecuaciones del sistema en espacio de estados-----------------------------------

A = [1.8980 -0.9048;1 0];
B = [1;0];
C = [0.0048 0.0047];

%Matrices de ponderación del índice de desempeño--------------------------------

Q = diag ( [ 1 1 ] );
R = 2;
polos_la = eig(A)                          %Polos de la matriz A en lazo abierto

%Diseño de política inicial-----------------------------------------------------

P_e = [1 0; 0 1]
Pp = P_e;
K = -inv(R + B'*P_e*B)*B'*P_e*A
K_1 = -K(1, 1)
K_2 = -K(1, 2)
polos_lc = eig(A + B*K)

%Parámetros iniciales-----------------------------------------------------------

N = 250;                                   %Tiempo total N
gamma = 0.999;                             %Factor de descuento
p = 3;                                     %Términos independientes de P_e
num_ep = 12;                               %Número de episodios
n_ls = 7;                                  %Tamaño de lote para LS
l = 0;
con = 0;
ep = 0;
datos = 0;

%Parámetros del crítico---------------------------------------------------------

f_olvido = 0.99;
a = 1 - f_olvido;
Iden = eye(p);
condicion_e_k = 10e-6;                     %Condición del error de TD
convergencia = 1;                          %Convergencia = 0
                                           %solo cuando se cumple la condición
                                           %del error
b = 1;

%Parámetros del actor-----------------------------------------------------------
```

```
55
56  beta = 0.0102;                                  %Tasa de aprendizaje del DG
57
58  %Señal de
       excitación-------------------------------------------------------------
59
60  [inp,wk] = michirp(0.005,345000,N+1,0.1);
61
62  %Condiciones
       iniciales--------------------------------------------------------------
63
64  x(:,1) = [0.5;-0.4];
65
66  %Inicio del
       algoritmo--------------------------------------------------------------
67
68  while ep < num_ep
69    datos = 0;
70    ep = ep + 1;
71    convergencia = 1;
72    dd = 1;
73    P_ant = P_e;
74
75  %Crítico---------------------------------------------------------------------------
76
77    for i = 1:N
78      if (convergencia == 1)
79        datos = datos + 1;
80        final_exc = datos;
81        if ep >= 1
82          u = K*x(:,i) + b*inp(i,2)*0.102;
83        end
84        x(:,i+1) = A*x(:,i) + B*u;
85        y(i,:) =  C*x(:,i);
86        r(i,:) = x(:,i)'*Q*x(:,i) + u'*R*u;
87        phi(:,i) = [x(1,i)^2; 2*x(1,i)*x(2,i) ; x(2,i)^2];
88        phi1(:,i) = [x(1,i+1)^2; 2*x(1,i+1)*x(2,i+1) ; x(2,i+1)^2];
89        Y(i,:) = r(i,:);
90
91        if i == n_ls
92          for k = 1:n_ls
93            phi_(:,k) = phi(:,k);
94            phi1_k1_(:,k) = phi1(:,k);
95            Y_(k,:) = Y(k,:);
96          endfor
97          %LS
98          vec_act_ls_ = (phi_ - gamma*phi1_k1_);
99          C_ls = vec_act_ls_*vec_act_ls_';
100         W = inv(C_ls)*vec_act_ls_*Y_;
101         theta_wls(:,n_ls) = W;
102         P_n = vec_act_ls_*eye(n_ls)*vec_act_ls_';
103       end
104
105       if i > n_ls
106         con = con + 1;
107         k = i;
108         phi_k = phi(:,k);
109         r_k = r(k,:);
110         phi1_k1 = phi1(:,k);
111         %RLS
112         vec_act = (phi_k - gamma*phi1_k1);
113         e_rls(:,k) = r_k - W'*(phi_k - gamma*phi1_k1);
114         e_k = e_rls(:,k);
115
116         if (abs(e_k) < condicion_e_k)
117           convergencia = 1;
118           vec_datos(1,dd) = datos;
```

```
119             b = 0;
120             dd = dd + 1;
121             final_exc = min(vec_datos);
122           end
123           L1(:,k) = 1/f_olvido*P_n*vec_act*inv(1/a +
                 vec_act'*1/f_olvido*P_n*vec_act);
124           L = L1(:,k);
125           theta_wls(:, k) = W + L*e_k;
126           P_n = 1/f_olvido*(Iden - L*vec_act')*P_n;
127           W = theta_wls(:, k);
128         end
129       end
130     endfor
131
132     final_exc
133     P_e = [theta_wls(1,k) theta_wls(2,k); theta_wls(2,k) theta_wls(3,k)];
134
135     error_P(:,ep) = norm(P_e - P_ant);
136     length(error_P);
137
138   %Actor-------------------------------------------------------------------------
139
140     U_j = K';
141     for i=1:datos
142       sigma_k = x(:, i);
143       dphi(:,:) = [2*x(1,i+1) 0; 2*x(2,i+1) 2*x(1,i+1); 0 2*x(2,i+1)];
144       diferencia = beta*sigma_k*(2*R*U_j'*sigma_k + gamma*B'*dphi'*W)';
145       U_j1  = U_j - diferencia;
146       K_es = U_j1;
147       U_j = K_es; %K_es ya está retroalimentada
148     endfor
149   %-----------------------------------------------------------------------------
150     error_K(:,ep) = norm(K' - K_es);
151     K_e = -K_es';
152     K = -K_e;
153
154     %Acomodar los valores de K para las gráficas
155     for c = 1:2
156       K1(1,c+l) = K_e(1,c);
157     end
158
159     %Acomodar los valores de P para las gráficas
160     for f = 1:2
161       for c = 1:2
162       P1(f,c+l) = P_e(f,c);
163       end
164     end
165
166     l = l + 2;
167
168   endwhile
169
170   K_e
171   P_e
172   [ K0 , P0 ] = dlqr( A , B , Q , R )
```

A.5. Iteración de política - usando la VFA con diferencias temporales para el problema del LQR (sistema inestable)

```
close all;
clear all;
clc;
set(0, "defaulttextfontname", "Arial");
set(0, "defaultlinelinewidth", 1.5);

%Ecuaciones del sistema en espacio de estados-----------------------------------
%Inestable
A = [-1 2;2.2 1.7];
B = [2;1.6];
C = [1 2];

%Matrices de ponderación del índice de desempeño--------------------------------

Q = diag ( [ 1 1 ] );
R = 2;
polos_la = eig(A)                          %Polos de la matriz A en lazo abierto

%Diseño de la ganancia inicial--------------------------------------------------

P_e = [6 1; 1 0]
Pp = P_e;
K = -inv(R + B'*P_e*B)*B'*P_e*A
K_1 = -K(1, 1)
K_2 = -K(1, 2)
polos_lc = eig(A + B*K)

%Parámetros iniciales-----------------------------------------------------------

N = 1050;                                  %Tiempo total N
gamma = 0.999;                             %Factor de descuento
p = 3;                                     %Términos independientes de P_e
num_ep = 12;                               %Número de episodios
n_ls = 7;                                  %Tamaño de lote para LS
l = 0;
con = 0;
ep = 0;
datos = 0;

%Parámetros del crítico---------------------------------------------------------

f_olvido = 0.99;
a = 1 - f_olvido;
Iden = eye(p);
condicion_e_k = 10e-4;                     %Condición del error de TD
convergencia = 1;                          %Convergencia = 0
                                           %solo cuando se cumple la condición
                                           %del error
b = 1;

%Parámetros del actor-----------------------------------------------------------

beta = 0.00202;                            %Tasa de aprendizaje del DG

```

```
%Señal de excitación-----------------------------------------------------------

[inp,wk] = michirp(0.005,3450000,N+1,0.1);

%Condiciones iniciales---------------------------------------------------------

x(:,1) = [0.5;-0.4];

%Inicio del algoritmo----------------------------------------------------------

while ep < num_ep
  datos = 0;
  ep = ep + 1;
  convergencia = 1;
  dd = 1;
  P_ant = P_e;

%Crítico-----------------------------------------------------------------------

  for i = 1:N
    if (convergencia == 1)
      datos = datos + 1;
      final_exc = datos;
      %Solo se consiera PE en el primer episodio
      if ep == 1
        u = K*x(:,i) + b*inp(i,2)*0.102;
      end
      if ep > 1
        u = K*x(:,i);
      end
      x(:,i+1) = A*x(:,i) + B*u;
      y(i,:) =  C*x(:,i);
      r(i,:) = x(:,i)'*Q*x(:,i) + u'*R*u;
      phi(:,i) = [x(1,i)^2; 2*x(1,i)*x(2,i) ; x(2,i)^2];
      phi1(:,i) = [x(1,i+1)^2; 2*x(1,i+1)*x(2,i+1) ; x(2,i+1)^2];
      Y(i,:) = r(i,:);
      if i == n_ls
        for k=1:n_ls
          phi_(:,k) = phi(:,k);
          phi1_k1_(:,k) = phi1(:,k);
          Y_(k,:) = Y(k,:);
        endfor
        %LS
        vec_act_ls_ = (phi_ - gamma*phi1_k1_);
        C_ls = vec_act_ls_*vec_act_ls_';
        W = inv(C_ls)*vec_act_ls_*Y_;
        theta_wls(:,n_ls) = W;
        P_n = vec_act_ls_*eye(n_ls)*vec_act_ls_';
      end

      if i > n_ls
        con = con + 1;
        k = i;
        phi_k = phi(:,k);
        r_k = r(k,:);
        phi1_k1 = phi1(:,k);
        %RLS
        vec_act = (phi_k - gamma*phi1_k1);
        e_rls(:,k) = r_k - W'*(phi_k - gamma*phi1_k1);
        e_k = e_rls(:,k);
        if (abs(e_k) < condicion_e_k)
          convergencia = 1;
          vec_datos(1,dd) = datos;
          b = 0;
```

```
            dd = dd + 1;
            final_exc = min(vec_datos);
          end
          L1(:,k) = 1/f_olvido*P_n*vec_act*inv(1/a +
             vec_act'*1/f_olvido*P_n*vec_act);
          L = L1(:,k);
          theta_wls(:, k) = W + L*e_k;
          P_n = 1/f_olvido*(Iden - L*vec_act')*P_n;
          W = theta_wls(:, k);
        end
      end
    endfor

    final_exc
    P_e = [theta_wls(1,k) theta_wls(2,k); theta_wls(2,k) theta_wls(3,k)];

    error_P(:,ep) = norm(P_e - P_ant);
    length(error_P);

%Actor-------------------------------------------------------------------------

    U_j = K';
    for i = 1:datos
      sigma_k = x(:, i);
      dphi(:,:) = [2*x(1,i+1) 0; 2*x(2,i+1) 2*x(1,i+1); 0 2*x(2,i+1)];
      diferencia = beta*sigma_k*(2*R*U_j'*sigma_k + gamma*B'*dphi'*W)';
      U_j1  = U_j - diferencia;
      K_es = U_j1;
      U_j = K_es; %K_es ya está retroalimentada
    endfor
%-------------------------------------------------------------------------------
    error_K(:,ep) = norm(K' - K_es);
    K_e = -K_es';
    K = -K_e;

    %Acomodar los valores de K para las gráficas
    for c = 1:2
      K1(1,c+l) = K_e(1,c);
    end

    %Acomodar los valores de P para las gráficas
    for f = 1:2
      for c = 1:2
      P1(f,c+l) = P_e(f,c);
      end
    end
    l = l + 2;
  endwhile
  K_e
  P_e
  [ K0 , P0 ] = dlqr( A , B , Q , R )
```

Anexo B

Contiene los códigos diseñados para los algoritmos de RL vistos el Capítulo 4, sección 4.3, correspondientes a dinámica totalmente desconocida. Se pueden consultar los códigos completos en Miguel (2023).

B.1. Iteración de política - usando la VFA con Q-learning para el problema del LQR (LS+GD)

```
close all;
clc;
clear all;

set(0, "defaulttextfontname", "Arial")
set(0, "defaultlinelinewidth", 1.5)

%Ecuaciones del sistema en espacio de estados-----------------------------------

A = [1.8980 -0.9048;1 0];
B = [1;0];
C = [0.0048 0.0047];

eig(A)

%Matrices de ponderación del índice de desempeño--------------------------------

Q = diag ( [ 1 1 ] );
R = 2;
polos_la = eig(A)

%Diseño de política inicial-----------------------------------------------------

H = [14 -2 2; -8 3 -1; 8 -5 4]; %(lc =  -0.64075   0.53875)
eig(H);
H_ini = [H(1,1), H(1,2), H(1,3), H(2,2), H(2,3), H(3,3)]'
Huu = H(3,3);
Hux = H(3,1:2);
K = -inv(Huu)*Hux
polos_lc = eig(A+B*K)
K_1 = K(1, 1);
K_2 = K(1, 2);

%Parámetros iniciales-----------------------------------------------------------

N = 4000;                                   %Tiempo total N
gamma = 0.999;                              %Factor de descuento
p = 3;
p_indep = 6;                                %Términos independientes de H
n_ls = 7;                                   %Tamaño de lote para LS

phi_ls = zeros(p_indep, n_ls);
phi1_ls = zeros(p_indep, n_ls);
```

```
r_ls = zeros(n_ls,1);
l = 0;
ll = 0;
iteracion = 0;
conteo_1 = 0;
datos = 0;
dd = 1;

%Parámetros del crítico----------------------------------------------------------

con_error_k = 10e-2;                    %Condición del error de TD
alpha = 120.02;                         %Tasa de aprendizaje del GD
b = 1;

%Señal de excitación-------------------------------------------------------------

[inp,wk] = michirp(0.005,3450000,N+1,0.1);

%Condiciones iniciales-----------------------------------------------------------
x = [5; -4];

%Inicio del algoritmo------------------------------------------------------------
tic
for i = 1:N
  datos = datos + 1;
  u = K*x(:,i) + b*inp(i,2)*0.202;
  x(:,i+1) = A*x(:,i) + B*u;
  y(i,:) = C*x(:,i);
  u2 = K*x(:,i+1);
  H_ant = H;
  K_ant = K;

  r_ls = circshift(r_ls,-1);
  phi_ls = circshift(phi_ls,-1, 2);
  phi1_ls = circshift(phi1_ls,-1, 2);

  r_ls(n_ls,1) = x(:,i)'*Q*x(:,i) + u'*R*u;
  phi_ls(:,n_ls) = [x(1,i)^2; x(1,i)*x(2,i); x(1,i)*u; x(2,i)^2; x(2,i)*u; u^2];
  phi1_ls(:,n_ls) = [x(1,i+1)^2; x(1,i+1)*x(2,i+1); x(1,i+1)*u2; x(2,i+1)^2; x(2,i+1)*u2; u2^2];

  if i == n_ls
    %LS
    vec_act_ls = (phi_ls - phi1_ls);
    C_ = vec_act_ls*vec_act_ls';
    rank(C);
    W_H_ant = inv(C_)*vec_act_ls*r_ls;
  end

  if i > n_ls
    r = x(:,i)'*Q*x(:,i) + u'*R*u;
    phi = [x(1,i)^2; x(1,i)*x(2,i); x(1,i)*u; x(2,i)^2; x(2,i)*u; u^2];
    phi1 = [x(1,i+1)^2; x(1,i+1)*x(2,i+1); x(1,i+1)*u2; x(2,i+1)^2; x(2,i+1)*u2; u2^2];
    conteo_1 = conteo_1 + 1;
    %GD
    vec_act = (phi - gamma*phi1);
    W_H = W_H_ant - alpha*vec_act*(W_H_ant'*vec_act - r);
    e_d_t = -W_H'*vec_act + r;
    if norm(e_d_t) < con_error_k
      iteracion = iteracion + 1;
      H = [ W_H(1,1), W_H(2,1)/2, W_H(3,1)/2;
```

```
104             W_H(2,1)/2, W_H(4,1), W_H(5,1)/2;
105             W_H(3,1)/2, W_H(5,1)/2, W_H(6,1)];
106       Huu = H(3,3);
107       Hux = H(3,1:2);
108       K = -inv(Huu)*Hux;
109       K_e = -K;
110       W_H_ant = W_H;
111
112       %Acomodar los valores de K para las gráficas
113       for c = 1:2
114         K1(1,c+l) = K_e(1,c);
115       end
116
117       l = l + 2;
118
119       %Acomodar los valores de P para las gráficas
120       for f = 1:p
121         for c = 1:p
122         H1(f,c+ll) = H(f,c);
123         end
124       end
125       H1;
126       ll = ll + 3;
127       error_H(:, iteracion) = norm(H - (H_ant));
128       error_K(:, iteracion) = norm(K_e - (-K_ant));
129   end
130 end
131 end
132
133 toc
134 H
135 K_e
136
137 [K_opt P_opt] = dlqr(A, B, Q, R)
138
139 H_opt = [Q + gamma*A'*P_opt*A,  gamma*A'*P_opt*B;
140          gamma*B'*P_opt*A, R+gamma*B'*P_opt*B]
```

B.2. Iteración de política - usando la VFA con Q-learning para el problema del LQR (LS+RLS)

```
1  close all;
2  clc;
3  clear all;
4
5  set(0, "defaulttextfontname", "Arial")
6  set(0, "defaultlinelinewidth", 1.5)
7
8  %Ecuaciones del sistema en espacio de
     estados-----------------------------------
9
10 A = [1.8980 -0.9048;1 0];
11 B = [1;0];
12 C = [0.0048 0.0047];
13
14 eig(A)
15
16 %Matrices de ponderación del índice de
     desempeño---------------------------------
17
```

```
Q = diag ( [ 1 1 ] );
R=2;

%Diseño de política inicial-----------------------------------------------------

H=[14 -2 2; -8 3 -1; 8 -5 4];
eig(H);
H_ini = [H(1,1), H(1,2), H(1,3), H(2,2), H(2,3), H(3,3)]'
Huu=H(3,3);
Hux=H(3,1:2);
K=-inv(Huu)*Hux
va_propi_lazo_cerrado = eig(A+B*K)
K_1 = -K(1, 1)
K_2 = -K(1, 2)

%Parámetros iniciales-----------------------------------------------------------

N = 4000;                                   %Tiempo total N
gamma = 0.999;                              %Factor de descuento
p = 3;
p_indep = 6;                                %Términos independientes de H
n_ls = 7;                                   %Tamaño de lote para LS


phi_ls= zeros(p_indep, n_ls);
phi1_ls= zeros(p_indep, n_ls);
r_ls = zeros(n_ls, 1);
l = 0;
ll = 0;
iteracion = 0;
c_r = 1;
datos = 0;

%Parámetros del crítico---------------------------------------------------------

con_error_k = 10e-2;                        %Condición del error de TD
f_olvido = 0.99;                            %Factor de olvido
a = 1 - f_olvido;
Iden = eye(6);

%Condiciones iniciales----------------------------------------------------------
x=[5; -4];
tic
for i=1:N
  datos = datos + 1;
  final_ruido = datos;
  nn(i,1) = 0.20099*exp(-0.000269*i)*(sin(i)^(3)*cos(i) +
     sin(2*i)^(2)*cos(0.46*i) + sin(-1.4*i)^(2)*cos(0.5*i) + sin(i)^(5) +
     sin(0.5*i)^(5) + 0.4*sin(1.99*i)^(2)*cos(2*i)+0.4*sin(18.0*i)^(5));
  u = K*x(:,i) + nn(i,1);
  x(:,i+1) = A*x(:,i)+B*u;
  y(i,:) = C*x(:,i);
  u2 = K*x(:,i+1);
  H_ant = H;
  K_ant = K;

  if i <= n_ls
    r_ls = circshift(r_ls,-1);
    phi_ls = circshift(phi_ls,-1, 2);
    phi1_ls = circshift(phi1_ls,-1, 2);

    r_ls(n_ls,1) = x(:,i)'*Q*x(:,i) + u'*R*u;
```

```
78      phi_ls(:,n_ls) = [x(1,i)^2; x(1,i)*x(2,i); x(1,i)*u; x(2,i)^2; x(2,i)*u;
           u^2];
79      phi1_ls(:,n_ls) = [x(1,i+1)^2; x(1,i+1)*x(2,i+1); x(1,i+1)*u2; x(2,i+1)^2;
           x(2,i+1)*u2; u2^2];
80
81      if mod(i,n_ls) == 0
82        %LS
83        vec_act_ls = (phi_ls - phi1_ls);
84        C_ = vec_act_ls*vec_act_ls';
85        rank(C);
86        W_H = inv(C_)*vec_act_ls*r_ls;
87        P_n = vec_act_ls*vec_act_ls'*1000;
88      end
89    end
90
91    if i>n_ls
92      r = x(:,i)'*Q*x(:,i) + u'*R*u;
93      phi = [x(1,i)^2; x(1,i)*x(2,i); x(1,i)*u; x(2,i)^2; x(2,i)*u; u^2];
94      phi1 = [x(1,i+1)^2; x(1,i+1)*x(2,i+1); x(1,i+1)*u2; x(2,i+1)^2;
           x(2,i+1)*u2; u2^2];
95      %RLS
96      vec_act = (phi - gamma*phi1);
97      e_d_t = r - W_H'*(phi - gamma*phi1);
98      L = 1/f_olvido*P_n*vec_act*inv(1/a + vec_act'*1/f_olvido*P_n*vec_act);
99      W_H = W_H + L*e_d_t;
100     P_n = 1/f_olvido*(Iden - L*vec_act')*P_n;
101     if norm(e_d_t)<con_error_k
102       iteracion = iteracion + 1;
103       H = [W_H(1,1), W_H(2,1)/2, W_H(3,1)/2;
104            W_H(2,1)/2, W_H(4,1), W_H(5,1)/2;
105            W_H(3,1)/2, W_H(5,1)/2, W_H(6,1)];
106       Huu = H(3,3);
107       Hux = H(3,1:2);
108       K = -inv(Huu)*Hux;
109       K_e = -K;
110
111       %Acomodar los valores de K para las gráficas
112       for c = 1:2
113         K1(1,c+l) = K_e(1,c);
114       end
115
116       l=l+2;
117
118       %Acomodar los valores de P para las gráficas
119       for f = 1:p
120         for c = 1:p
121         H1(f,c+ll) = H(f,c);
122         end
123       end
124       H1;
125       ll=ll+3;
126       error_H(:, iteracion) = norm(H - (H_ant));
127       error_K(:, iteracion) = norm(K_e - (-K_ant));
128     end
129   end
130 end
131
132 toc
133 H
134 K_e
135
136 [K_opt P_opt] = dlqr(A, B, Q, R)
137
138 H_opt = [Q + gamma*A'*P_opt*A,  gamma*A'*P_opt*B;
139          gamma*B'*P_opt*A, R+gamma*B'*P_opt*B]
```

Anexo C

C.1. Iteración de política - usando la VFA con diferencias temporales para el problema del LQR en entrenamiento continuo LS+RLS

Algoritmo 6: Iteración de política - usando VFA con diferencias temporales para el problema del LQR en entrenamiento continuo

Requeridos: Condición inicial x_0, política de control admisible $h_0(x)$, matrices de ponderación Q, R, se define $j = 0$, longitud N de la trayectoria, el tamaño del lote n_{ls} de LS, valor de δ, factor de olvido λ, tasa de aprendizaje β y un factor de descuento γ;

para $k = 0$ **a** N **hacer**

  % Evaluación de la política;

  Calcular en x_k la acción de control u_k con una excitación persistente ε como $u_k = h_j(x_k) + \varepsilon$;

  Aplicar u_k al sistema y medir x_{k+1};

  si $k = n_{ls}$ **entonces**

    Determinar la solución con LS para el cálculo de W^1 de RLS;

$$W^1 = \Gamma(n_{ls})\Phi^{'}(n_{ls})Y(n_{ls})$$

  fin

  si $k > n_{ls}$ **entonces**

    Determinar la solución con RLS para;

$$\left(W^{j+1}\right)^T \Phi^{'}(k) = x_k^T Q x_k + u_k^T R u_k \tag{134}$$

    si $e_{TD} \leq \delta$ **entonces**

      % Mejoramiento de la política;

      para $i = 0$ **a** ℓ **hacer**

        Actualizar los parámetros de la política usando GD;

$$U_{j+1}^{i+1} = U_j^i - \beta\sigma(x_i)\left(2R(U_j^i)^T\sigma(x_i) + \gamma g(x_i)^T \nabla\Phi^T(i+1)W^{j+1}\right)^T \tag{135}$$

      fin

      $h_{j+1}(x_\ell) = \left(U_{j+1}^{\ell+1}\right)^T \sigma(x_\ell)$;

      $j \leftarrow j + 1$;

    fin

  fin

fin