Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería Mecatrónica

Aprendizaje en tiempo real y control de manipuladores robóticos

Informe de Proyecto de Graduación para optar por el título de Ingeniero en Mecatrónica con el grado académico de Licenciatura

Alejandro José Ordóñez Conejo

Cartago, julio de 2020

Technische Universität München | Arcisstraße 21 | 80333 München

Tecnológico de Costa Rica
Eng. Ana Lucía Morera Barquero
Graduation Projects' Coordinator
Mechatronics Engineering
Costa Rica

München, 8. Juli 2020

Confirmation Letter

Dear Eng. Morera Barquero,

I herewith confirm that the student Alejandro José Ordóñez Conejo, ID 207650024, student ID 2015069998 has fulfilled the requirements and tasks regarding the thesis titled "Real-time Learning and Control of Robotic Manipulators" from 13.01.2020 until 28.06.2020.

At our Chair of Information-oriented Control we are satisfied with the work regarding his Bachelor thesis done by Mr. Ordóñez.

Prof. Dr.-Ing. Sandra Hirche

Lehrstuhl für
Informationstechnische Regelung
Ordinaria: Univ.-Prof. Dr.-Ing. Sandra Hirche
Technische Universität München
80290 München
Tel.: 089 / 289 25722
Fax: 089 / 289 25724
Email: itr@ei.tum.de

Technische Universität München
Fakultät für Elektro- und Informationstechnik
Lehrstuhl für Informationstechnische Regelung

Prof.
Sandra Hirche
Barer Str. 21
80333 München

Tel. +49 89 289 23403
Fax +49 89 289 28340

hirche@tum.de
www.itr.ei.tum.de
www.tum.de

Bayerische Landesbank
IBAN-Nr.:
DE10700500000000024866
BIC: BYLADEMM
Steuer-Nr.: 143/241/80037
USt-IdNr.: DE811193231

INSTITUTO TECNOLÓGICO DE COSTA RICA

CARRERA DE INGENIERÍA MECATRÓNICA

PROYECTO DE GRADUACIÓN

ACTA DE APROBACIÓN

El Profesor Asesor, da fe de que el presente Proyecto de Graduación ha sido aprobado y cumple con las normas establecidas por la Carrera de Ingeniería Mecatrónica como requisito para optar por el título de Ingeniero en Mecatrónica, con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Estudiante: Alejandro José Ordóñez Conejo

Nombre del Proyecto: Aprendizaje en tiempo real y control de manipuladores robóticos

Ing. Juan Carlos Brenes Torres

Profesor Asesor

Cartago, viernes 14 de agosto del 2020.

Resumen

En las últimas décadas, el uso de manipuladores robóticos en la industria ha incrementado ampliamente. Esto ha resultado en producción eficiente y evolución de los puestos laborales. Sin embargo, en muchos casos los manipuladores pueden verse afectados por entornos cambiantes. Asimismo, no hay suficientes garantías de seguridad para que un manipulador robótico trabaje en conjunto con personas. El propósito de este proyecto es desarrollar un novedoso método de control automático que permita al manipulador aprender del entorno y adaptarse en tiempo real. Se utiliza aprendizaje máquina (procesos gaussianos locales) en conjunto con la linealización torque calculado en un manipulador SCARA de dos grados de libertad. Los resultados demuestran que el control aprende a adaptarse eficientemente y tiene valores de error considerablemente menor que sin el aprendizaje.

Palabras clave: procesos gaussianos, torque calculado, aprendizaje máquina, tiempo real, manipulador robótico

Abstract

In the last decades, the use of robotic manipulators in the industry has widely increased. This has resulted in efficient production and evolution of jobs. However, in many cases the robotic manipulators are affected by changing environments. Moreover, there are not enough safety warranties for a robotic manipulator to work together with people. The purpose of this work is to develop a new automatic control method that allows the manipulator to learn from the environment and adapt in real time. Machine learning is used (local gaussian processes) together with computed torque linearization in a SCARA manipulator with two degrees of freedom. The results show that the control manages to learn to adapt efficiently and have values of error considerably smaller than without learning.

Keywords: gaussian processes, computed torque, machine learning, real time, robotic manipulator

A mis padres

Índice general

Índice de figuras

Índice de tablas

Símbolos y notación

Símbolo	Significado
$\sim$	Distribución probabilística
$a\|b, p(a\|b)$	Distribución condicional y probabilidad de a dado b
$\boldsymbol{A}\backslash\boldsymbol{B}$	Solución **z** de $\boldsymbol{Az} = \boldsymbol{B}$
$\boldsymbol{\mathcal{D}}$	Conjunto de datos
$\det$	Determinante
ε_i	Error por ruido de y_i
$f(\boldsymbol{x_i})$	Escalar i-ésimo de salida con ruido
$\boldsymbol{I}$	Matriz identidad
k	Dimensionalidad de $\boldsymbol{x_i}$
k_p	Control proporcional
$\boldsymbol{k_v}$	Control derivativo
$K(\boldsymbol{X}, \boldsymbol{X})$	Matrix cuadrada de covarianza
l	Escala de longitud
$\log$	Logaritmo natural
$L(\theta, \dot{\theta})$	Lagrangiano
n	Cantidad de datos aprendidos
$\boldsymbol{\mathcal{N}}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Distribución normal de media $\boldsymbol{\mu}$ y covarianza $\boldsymbol{\Sigma}$
$\mathcal{O}$	Notación *Big O*
σ_n	Desviación estándar del ruido
σ_f	Desviación estándar de la señal de entrada
$tr(\boldsymbol{A})$	Traza de $\boldsymbol{A}$
$\boldsymbol{u_\tau}$	Señal de entrada
$\boldsymbol{x}^*$	Vector i-ésimo de entrada de prueba, $\in \mathbb{R}^k$
$\boldsymbol{x_i}$	Vector i-ésimo de entrada aprendido, $\in \mathbb{R}^k$
$\boldsymbol{X}$	Matriz con datos aprendidos
y_i	Escalar i-ésimo de salida aprendido
$\boldsymbol{y}$	Vector de salidas

Capítulo 1. Introducción

1.1. Entorno

La robótica y la inteligencia artificial tienen un importante rol en la economía mundial. Las estadísticas publicadas anualmente por la *International Federation of Robotics* (IFR) y descritas en [1] explican que desde el año 2010 hasta el 2016 hubo un incremento de aproximadamente 150 % en los envíos mundiales de robots para industria. En [2] se describe el incremento en el volumen de producción de robots en Corea del Sur: el valor registrado en 2009 fue de USD 842 millones, mientras que en 2014 fue de USD 2 600 millones.

La interacción entre humanos y robots representa un nuevo paso en los procesos productivos. Para asegurar un funcionamiento seguro y correcto, se debe garantizar que las máquinas puedan adaptarse a funcionar en cercanía de las personas sin disminuir la eficiencia. La interacción humano-máquina, posible mediante inteligencia artificial, permite mejorar la eficiencia y evolucionar los roles que tienen las personas en la producción [3].

La Cátedra de Control Orientado a la Información pertenece al Departamento de Ingeniería Eléctrica e Informática de la Universidad Técnica de Múnich. La investigación de esta cátedra se basa en robótica cooperativa. El área de investigación del control inducido por datos se fundamenta en utilizar paradigmas de aprendizaje máquina para modelar sistemas matemáticos complejos. Utilizando los algoritmos de aprendizaje máquina, se puede desarrollar novedosos controladores con ventajas sobre los métodos clásicos.

1.2. Justificación

Desarrollar el modelo matemático de un sistema es el primer paso para poder controlarlo. En el caso de muchos manipuladores robóticos el modelo matemático no es lineal: los elementos giran respecto a las uniones. Desarrollar un controlador basado en linealización puede simplificar el proceso de control, sin embargo, esta tarea puede ser

compleja e inexacta. La flexibilidad de los elementos, el desgaste del dispositivo, la variabilidad ambiental, y las discontinuidades son algunas de las razones por las cuales obtener un modelo matemático sin error es imposible. Utilizar un controlador que pueda adaptar el modelo del sistema con datos del funcionamiento, puede hacer que el error sea anulado [4].

Los métodos convencionales de control son utilizados con los manipuladores robóticos. Utilizando estas técnicas, se puede disminuir considerablemente el error de desempeño de los manipuladores cuando se utiliza altas ganancias. Sin embargo, tener altas ganancias en un lazo de control representa, físicamente, un consumo energético mayor, movimientos repentinos del sistema y vibraciones. Utilizar ganancias menores pueden mitigar estos problemas, sin embargo, la exactitud, es menor.

Los métodos de aprendizaje máquina han sido utilizados en las últimas décadas para adaptar controladores. El objetivo de estos avances es desarrollar un controlador que logre tener los beneficios de altas y bajas ganancias sin tener las limitaciones respectivas de estos métodos. Se ha demostrado en [5] que se puede obtener un desempeño con error bajo si se utiliza aprendizaje máquina para adaptar el error al sistema real. En [5] se observa que se puede lograr un desempeño similar al controlador de alta ganancias utilizando bajas ganancias y aprendizaje máquina.

Un importante factor por tomar en consideración en el uso de paradigmas de inteligencia artificial es el método de aprendizaje. La información que se debe utilizar para que el sistema aprenda debe ser obtenida del manipulador durante un óptimo funcionamiento de este. Para esto, se puede desarrollar pruebas y exportar datos y posteriormente entrenar el modelo, o bien, realizar aprendizaje durante la prueba. A estas técnicas se les llama aprendizaje fuera de línea y aprendizaje en línea respectivamente. Aunque diseñar un sistema que aprenda fuera de línea es más simple, la ejecución de un sistema que aprende en línea es más eficiente. Asimismo, si se presentan cambios en el entorno, el aprendizaje en línea podrá adaptarse a esto conforme suceden. En [6] se presentan diversas técnicas de aprendizaje en línea han sido desarrolladas con éxito.

1.3. Problema

El método de aprendizaje máquina desarrollado en la cátedra no permite aprendizaje en línea que funcione eficientemente para que el control de los manipuladores robóticos pueda adaptarse en tiempo real al entorno.

1.4. Objetivos

Objetivo general

Desarrollar un sistema de control automático que incluya procesos gaussianos locales, y que pueda aprender en tiempo real, para controlar el torque de un robot SCARA

Objetivos específicos

- Implementar un algoritmo de aprendizaje máquina capaz de aprender en tiempo real mediante procesos gaussianos locales.
- Diseñar un paradigma de control que permita combinar el aprendizaje máquina con otros métodos de control.
- Evaluar el efecto en el desempeño y en los resultados que tienen los métodos de distribución de los datos en el algoritmo de aprendizaje máquina.
- Comprobar la eficacia del algoritmo de aprendizaje y control elaborados mediante implementaciones en mecanismos robóticos.

1.5. Análisis financiero

La inversión necesaria para implementar el control descrito en este proyecto es expuesta en esta sección. La **Tabla 1.1** muestra los costos necesarios para implementar el sistema.

Tabla 1.1. Descripción aproximada de la inversión necesaria para desarrollar el proyecto.

Descripción	Costo aproximado (₡)
Manipulador SCARA*	3 000 000 – 12 000 000
Computador*	500 000 – 2 000 000
Licencia MATLAB	1 400 000
*Statistics and Machine Learning Toolbox***	265 250
Total	5 165 250 – 15 665 250

* El valor depende de la aplicación que se pretende para el manipulador, así como la precisión y el proveedor.

** *Toolbox* de Matlab, puede omitirse según los requerimientos y la implementación del aprendizaje.

Los datos descritos en esta sección corresponden a aproximaciones basadas en diversos proveedores. El costo final de implementación dependerá de la aplicación y la cantidad de componentes necesarios.

1.6. Trabajo relacionado

- En [5] se utiliza procesos gaussianos en el control de un manipulador robótico y el ala de un vehículo aéreo no tripulado. Los resultados son asintóticamente estables.
- En [7] se describe la comparación de procesos gaussianos, redes neuronales y modelos autorregresivos para predecir el comportamiento de las olas y poder controlar los convertidores de energía undimotriz. Usar aprendizaje máquina puede maximizar la eficiencia energética.
- En [8] se utiliza procesos gaussianos dispersos en la implementación del control de una minicargadora. La variabilidad del terreno, las condiciones ambientales y la sensibilidad de estos vehículos para girar hace que el control sea complejo sin aprendizaje.
- En [9] se utiliza procesos gaussianos para identificar y estabilizar sistemas no lineales y parcialmente desconocidos.
- En [10] se analiza la dinámica del gas de escape en motores de combustión mediante procesos gaussianos. Con los datos aprendidos se implementa control que resulta para aliviar el efecto de dinámica y reacciones complejas de modelar.
- En [11] se utiliza procesos gaussianos locales desarrollar control evolutivo de un exoesqueleto humano con base en los movimientos de la persona. Se demuestra que la fuerza de interacción, el amortiguamiento y el esfuerzo del humano es reducido.

1.7. Estructura

El restante de este informe está distribuido de la siguiente manera. En el Capítulo 2 se describe la metodología del proyecto. En el Capítulo 3 se exponen los conceptos teóricos de control y aprendizaje utilizados en este documento. En el Capítulo 4 se explica la implementación del sistema para las evaluaciones simuladas y físicas. En el Capítulo 5 se evalúa el desempeño del método desarrollado y se compara con otros métodos. Las conclusiones y recomendaciones principales están compiladas en el 0.

Capítulo 2. Metodología

En este capítulo se describe las etapas desarrolladas durante la ejecución del proyecto. Se plantea los métodos y fuentes de investigación, así como el análisis de los requerimientos del proyecto. Se analiza la distribución de tiempo y las prioridades para el desarrollo del proyecto.

2.1. Análisis previo

La implementación de control inteligente es un área de conocimiento relativamente nueva. Se pretende analizar el antecedente a esta técnica, el control adaptativo, cuales limitaciones y potenciales mejoras impulsaron la implementación de inteligencia artificial en control automático.

Los controladores que implementan aprendizaje máquina no son desarrollados exclusivamente con un paradigma de aprendizaje, los diversos métodos se distinguen por factores como exactitud, implementación, precisión y rapidez de ejecución. Es, por lo tanto, necesario analizar el método que se desarrolla en este proyecto y compararlo con las diversas posibilidades que han sido investigadas en la academia.

El análisis previo debe culminar con la investigación de la implementación de paradigmas similares al que se desarrolla en este proyecto. Se busca analizar los antecedentes expuestos por los distintos autores, así como los entornos en donde el controlador podría representar una novedosa y conveniente solución. Es importante analizar las limitaciones y especificaciones del equipo disponible, así como las descritas por los otros autores para que el controlador tenga la mayor utilidad y versatilidad posible.

2.2. Especificaciones

El proyecto debe ser desarrollado con diversas consideraciones relacionadas con los recursos que se dispone. Se debe definir la interfaz en la que desarrolla el control y que se utiliza para controlar el manipulador en el cual se hará la demostración de funcionamiento. Se debe definir las especificaciones necesarias para que el proyecto sea funcional y que represente una novedad en el ámbito investigativo.

Las especificaciones que se analizan inicialmente son la frecuencia de actualización del sistema, así como a las variables físicas de torque, posición, velocidad y aceleración que el manipulador tiene. Se define un error de posición que sea competitivo con otros métodos publicados.

2.3. Interfaz de prueba

El controlador debe ser probado en simulaciones antes de la implementación en el manipulador. Se debe desarrollar un modelo matemático aproximado del manipulador en la interfaz de prueba. Los parámetros del modelo deben ser fácilmente variados. Las pruebas iniciales deben confirmar el comportamiento correcto de la simulación. Para esto, se desarrolla una animación del movimiento. Se desarrolla pruebas con los diversos controladores y se comprueba la estabilidad del sistema con el control.

Cuando se tiene certeza de que el modelo matemático es aproximadamente correcto, se agrega variaciones que hacen al sistema. Se inicia alterando los valores de la linealización, de manera que estos no coincidan con el manipulador. Las simulaciones son ejecutadas nuevamente y se busca estabilizar el sistema y mantener el error en un valor bajo. La última consideración que se agrega a la simulación es el ruido. Se agrega ruido a las señales que, en el manipulador, provienen de los sensores. Se ejecuta las mismas pruebas y se compara los valores de error.

2.4. Comparaciones

La efectividad del algoritmo es determinada al compararse con otros métodos. Se plantea buscar las referencias con mayor similitud y mejor desempeño. Los métodos seleccionados son implementados u obtenidos según disponibilidad. Debido a la complejidad de desarrollar el método en la interfaz relacionada con el aprendizaje en tiempo real, las comparaciones serán desarrolladas únicamente con los conjuntos de datos disponibles relacionados al control de manipuladores. Las comparaciones son ejecutadas utilizando las recomendaciones descritas por los diversos autores.

La selección de los parámetros del método propio debe basarse en diversas pruebas que cuantifiquen el efecto que cada uno tiene. Se observa los resultados ante distintas posibilidades y se selecciona el que tiene un mejor desempeño respecto al error y a al

tiempo de ejecución. Se analiza de manera teórica y práctica el efecto que cada parámetro tiene en los resultados.

2.5. Implementación y análisis

La demostración de la eficacia del método desarrollado se hace con la planta física. Se debe considerar los factores distintivos entre la simulación y la realidad. Para asegurar un comportamiento similar, se debe analizar el comportamiento de las señales obtenidas desde los sensores y filtrarlas en caso de ser necesario. Se debe considerar el desfase que resulta de las predicciones respecto a las señales y el efecto que la frecuencia de actualización puede tener.

Los resultados obtenidos de la implementación son analizados y comparados con los de la simulación. Se analiza el comportamiento del error y el efecto que tiene el ruido sobre las señales. Se compara el desempeño ante distintas frecuencias de actualización y diferentes filtros.

La etapa final corresponde al análisis de las potenciales aplicaciones que se le puede dar al método desarrollado. Se reflexiona sobre las posibles mejoras que pueden desarrollarse en trabajo posterior, así como los diversos manipuladores en los cuáles se puede implementar el controlador.

2.6. Optimización e iteración

El código utilizado para implementar el algoritmo de aprendizaje máquina debe ser estudiado para poder ser optimizado. Se utiliza diversas herramientas para determinar los segmentos críticos del algoritmo y se analiza cómo disminuir su coste computacional. Se observa el efecto que tiene en el desempeño los cambios implementados. La criticidad se determina a partir de los métodos que consumen más tiempo promedio y los que son ejecutados la mayor cantidad de veces.

Cuando se hace mejoras, el efecto debe ser analizado, por lo tanto, las pruebas deben ser iteradas con las mejoras realizadas. Se debe comparar y los tiempos y errores para justificar los cambios desarrollados.

Capítulo 3. Marco teórico

Se desarrolla un controlador que utiliza aprendizaje máquina para optimizar el comportamiento de un manipulador robótico de dos grados de libertad. En este capítulo se explica los fundamentos teóricos del trabajo desarrollado. En la sección 3.1 se describe la mecánica del manipulador y se introduce un modelo matemático del sistema. Seguidamente, en la sección 3.2 se describe el controlador que se utiliza y el principio de funcionamiento. Finalmente, en la sección 3.3 se explica el método de aprendizaje máquina que se incorpora al controlador.

3.1. Dinámica del manipulador

El modelo matemático de un manipulador robótico se obtiene mediante principios mecánicos. Utilizando aproximaciones matemáticas se puede caracterizar la dinámica de un sistema. La dinámica de un manipulador describe cómo se mueve este ante fuerzas [4]. Se puede analizar la dinámica de dos maneras: dinámica directa y dinámica inversa. Utilizando dinámica directa, se analiza el movimiento resultante ante las fuerzas, mientras que utilizando dinámica inversa se analiza las fuerzas necesarias para generar el movimiento deseado.

En aplicaciones relacionadas a manipuladores robóticos, se requiere desarrollar un análisis de la dinámica inversa. Mediante el accionamiento de los actuadores se puede obtener una trayectoria deseada. Este accionamiento debe considerar: la posición deseada respecto a la posición actual, capacidad de los actuadores de realizar el movimiento, ruido y perturbaciones.

En este proyecto se desarrolla las pruebas de concepto en brazo robótico SCARA de dos grados de libertad. La **Figura 3.1** muestra un diagrama del manipulador estudiado. Los valores descritos son: l_i es el largo de del vínculo i, r_i es la distancia desde la articulación hasta el centro de masa del vínculo i y el valor I_{ki} es el tensor de inercia en el plano k del vínculo i que coincide con el centro de masa.

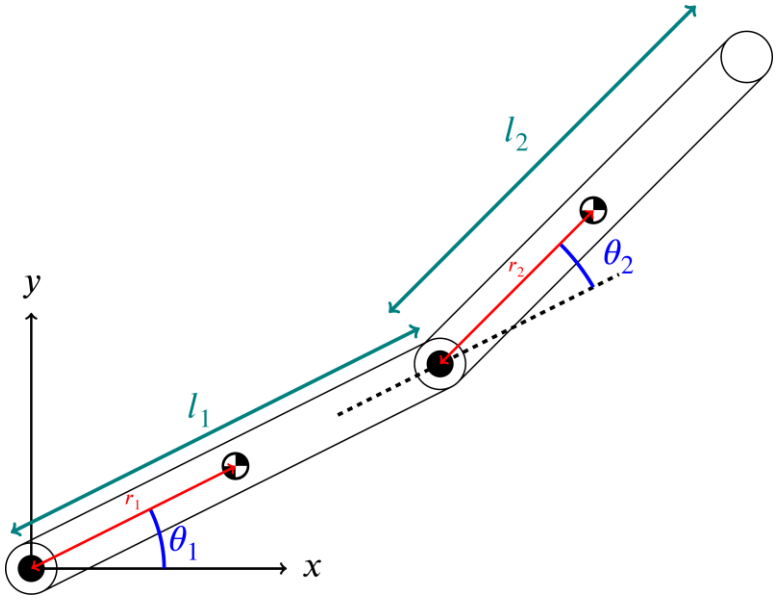

Figura 3.1. Diagrama del manipulador robótico de dos grados de libertad.

Para desarrollar el análisis del movimiento, se debe introducir el *Lagrangiano* de este sistema. Se define como *Lagrangiano* a la diferencia entre energía cinética y energía potencial del manipulador [12], de manera que:

$$L(\theta, \dot{\theta}) = E_c(\theta, \dot{\theta}) - E_p(\theta) \tag{3.1}$$

Donde $\theta = [\theta_1 \ \theta_2]^T$ es la posición angular, E_c es la energía cinética y E_p la energía potencial. Se utiliza el teorema 4.1 descrito en [12]:

Teorema 1.

Las ecuaciones mecánicas de movimiento de un sistema están dadas por:

$$\tau_i = \frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}_i} - \frac{\partial L}{\partial \theta_i} \tag{3.2}$$

Donde τ_i es la fuerza que actúa en i. Las ecuaciones (3.2) *se conocen como Ecuaciones de Lagrange.*

Utilizando las ecuaciones (3.1) y (3.2) para el sistema descrito en la **Figura 3.1** se tiene (ver Apéndice A):

$$\boldsymbol{\tau} = \boldsymbol{M}(\theta)\ddot{\theta} + \boldsymbol{V}(\theta, \dot{\theta})\dot{\theta} + \boldsymbol{G}(\theta) \tag{3.3}$$

Donde $\boldsymbol{M}$ es la matriz de masa, $\boldsymbol{V}$ es un vector de términos centrífugos y de Coriolis y $\boldsymbol{G}$ es el vector de gravedad [4]. Para el sistema de estudio se tiene:

$$\boldsymbol{M}(\theta) = \begin{bmatrix} I_{z1} + I_{z2} + m_1 r_1^2 + m_2(l_1^2 + r_2^2) + 2m_2 l_1 r_2 c_2 & I_{z2} + m_2 r_2^2 + m_2 l_1 r_2 c_2 \\ I_{z2} + m_2 r_2^2 + m_2 l_1 r_2 c_2 & I_{z2} + m_2 r_2^2 \end{bmatrix}$$

$$\boldsymbol{V}(\theta, \dot{\theta}) = \begin{bmatrix} -m_2 l_1 r_2 s_2 \dot{\theta}_2 & -m_2 l_1 r_2 s_2 (\dot{\theta}_1 + \dot{\theta}_2) \\ m_2 l_1 r_2 s_2 \dot{\theta}_1 & 0 \end{bmatrix}$$

$$\boldsymbol{G}(\theta) = \begin{bmatrix} g m_1 r_1 c_1 + g m_2 (l_1 c_1 + r_2 c_{12}) \\ g m_2 r_2 c_{12} \end{bmatrix}$$

Donde m_i es la masa del vínculo i, g es la aceleración gravitacional, $s_i = \sin\theta_i$, $c_i = \cos\theta_i$ y $c_{ij} = \cos(\theta_i + \theta_j)$.

3.2. Control de manipuladores

El controlador del manipulador se desarrolla a partir de la ecuación (3.3). Se considera el torque $\boldsymbol{\tau}$ la entrada del sistema y la posición angular $\theta = [\theta_1\ \theta_2]^T$, la velocidad angular $\dot{\theta}$ y la aceleración angular $\ddot{\theta}$ las salidas del sistema. De manera similar, los valores $\theta_d, \dot{\theta}_d$ y $\ddot{\theta}_d$ son la trayectoria deseada. Finalmente, los errores respectivos a estas variables son: $\boldsymbol{e} = \theta_d - \theta$ y $\dot{\boldsymbol{e}} = \dot{\theta}_d - \dot{\theta}$.

3.2.1. Control PD

El uso de un controlador PD resulta la opción más básica desarrollada en este trabajo para el control del manipulador robótico. El fundamento de este control es introducir un valor de ganancia proporcional y derivativa. Al ser un manipulador de dos grados de libertad, las ganancias deben ser matrices. De esta manera, se tiene el torque de entrada al sistema $\boldsymbol{u_\tau}$:

$$\boldsymbol{u_{pd}} = \boldsymbol{e} k_p + \dot{\boldsymbol{e}} k_v \tag{3.4}$$

Donde k_p y k_v son las ganancias proporcional y derivativa respectivamente. Estas están dadas por:

$$k_p = \begin{bmatrix} k_{p1} & 0 \\ 0 & k_{p2} \end{bmatrix}$$

$$k_v = \begin{bmatrix} k_{v1} & 0 \\ 0 & k_{v2} \end{bmatrix}$$

La exactitud del sistema y su rapidez de respuesta dependen de los valores de ganancias. Utilizar altos valores de ganancia resultan en menor error, sin embargo, esto representa menor seguridad en el entorno, mayor consumo energético y amplifica perturbaciones [5] [6].

3.2.2. Torque calculado

El torque calculado consiste en linealizar el sistema utilizando un sistema [4]. De esta manera se calcula el valor de entrada que el sistema requiere para ejecutar el movimiento deseado. Se debe obtener las propiedades físicas del sistema para obtener la aproximación basada en la ecuación (3.3). De esta manera, se calcula el torque de entrada mediante:

$$\boldsymbol{u_{tc}} = \widehat{\boldsymbol{M}}(\theta)\ddot{\theta}_d + \widehat{\boldsymbol{V}}\left(\theta, \dot{\theta}\right)\dot{\theta}_d + \widehat{\boldsymbol{G}}(\theta) \tag{3.5}$$

Para un modelo exacto, es decir, $\widehat{\boldsymbol{M}}(\theta) = \boldsymbol{M}(\theta)$, $\widehat{\boldsymbol{V}}\left(\theta, \dot{\theta}\right) = \boldsymbol{V}\left(\theta, \dot{\theta}\right)$ y $\widehat{\boldsymbol{G}}(\theta) = \boldsymbol{G}(\theta)$, entonces la respuesta es exacta, sin embargo, debido a las inexactitudes de la aproximación y la alteración de los parámetros, este control, por sí solo, es inexacto [13].

3.2.3. Control de aprendizaje

El fundamento del control de aprendizaje se remonta a el control adaptativo. El control adaptativo se fundamenta en adecuar el controlador ante un sistema que se conoce parcialmente y que puede cambiar con el tiempo [14]. Este control es apropiado para el manipulador robótico debido a la complejidad del sistema y el efecto que el entorno tiene en el desempeño. Debido a que el sistema de estudio presenta alteraciones relacionadas al ruido, este sistema es considerado estocástico. Un controlador adaptativo posee, generalmente, un controlador adicional, un modelo aproximado, y un mecanismo de ajuste que puede incidir directa o indirectamente en la señal de entrada al sistema [15].

Las principales limitaciones asociadas al control adaptativo son la cantidad de información que se puede registrar, la cantidad finita de configuraciones para controlar

al sistema y, como resultado, la incapacidad de utilizar la información relacionada a las circunstancias pasadas que tuvo el sistema [16] [17]. Utilizar paradigmas de inteligencia artificial representa la solución de los problemas descritos para el control adaptativo. A este control se denomina control de aprendizaje. La configuración del control adaptativo y el control de aprendizaje puede ser similar, sin embargo, el mecanismo de ajuste debe tener la capacidad de almacenar información y actuar según esta información para el control de aprendizaje.

Para este proyecto se pretende utilizar un complemento de los controles descritos en la sección 3.2. Este está descrito diagrama de control de la **Figura 3.2**.

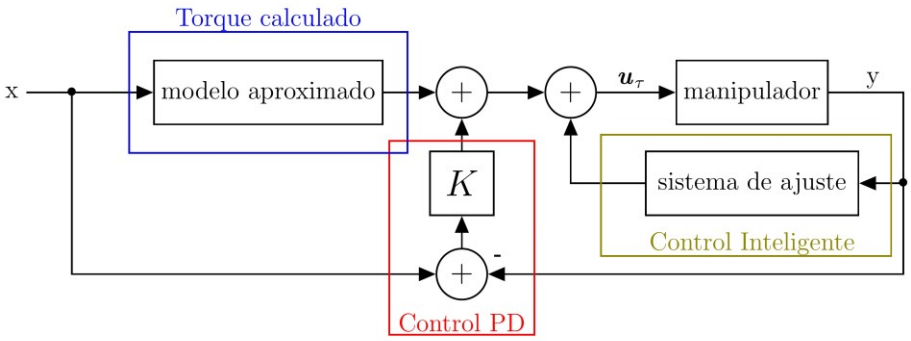

Figura 3.2. Diagrama de control planteado para el problema.

La señal de control del sistema está descrita por:

$$\boldsymbol{u_\tau = u_{pd} + u_{tc} + u_{ci}} \tag{3.6}$$

Para el sistema de estudio $\boldsymbol{u_{pd}}$ se obtiene de la ecuación (3.4) y $\boldsymbol{u_{tc}}$ de la ecuación (3.5).

El sistema de ajuste debe almacenar información del comportamiento del sistema. El sistema debe aprender, mediante el modelo aproximado y el modelo real, el error. De esta manera, el sistema podrá complementar esta falencia cuando se encuentre en una situación similar en futuras ejecuciones. Si el sistema logra aprender todas las circunstancias posibles, el error tenderá a un valor bajo, sin embargo, debido al ruido, el error nunca será cero.

3.3. Regresión mediante procesos gaussianos

3.3.1. Procesos gaussianos

Los procesos gaussianos (PG) son generalizaciones de distribuciones probabilísticas gaussianas que emplea inferencia Bayesiana [18]. Los procesos gaussianos pueden ser empleados para aprendizaje supervisado y, por lo tanto, en regresión. Se tiene un conjunto de i datos $\boldsymbol{\mathcal{D}} = \{\boldsymbol{X}, \boldsymbol{y}\}$, donde $\boldsymbol{X} = [\boldsymbol{x}_1\ \boldsymbol{x}_2 \dots\ \boldsymbol{x}_n]$, $\boldsymbol{y} = [y_1\ y_2 \dots y_n]$ y $\boldsymbol{x}_i \in \mathbb{R}^k$. El valor $\boldsymbol{x}_i$ es un vector de entrada que corresponde a la salida y_i. La relación entre entrada y salida es para el proceso gaussiano es:

$$f(\boldsymbol{x}_i) = y_i + \varepsilon_i$$

Donde $\varepsilon_i \sim \boldsymbol{\mathcal{N}}(0, \sigma_n^2)$ es el error relacionado al ruido. Se asume una distribución gaussiana multivariable de las salidas:

$$\boldsymbol{y} \sim \boldsymbol{\mathcal{N}}(0, \boldsymbol{K}_n)$$

Donde $\boldsymbol{K}_n = K(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \boldsymbol{I}$.

La función $K(\boldsymbol{x}_a, \boldsymbol{x}_b)$ se denomina función de covarianza. Existen diversas variaciones para esta función, sin embargo, para el presente trabajo, se emplea exponencial cuadrado:

$$K(\boldsymbol{x}_a, \boldsymbol{x}_b) = \sigma_f^2 \exp\left(-\frac{1}{2} \sum_{j=1}^{k} \left(\frac{\boldsymbol{x}_{a,j} - \boldsymbol{x}_{b,j}}{l} \right)^2 \right)$$

El exponencial cuadrado es frecuentemente utilizado. Tiene la propiedad de ser infinitamente diferenciable, por lo tanto, resulta en transiciones suaves [18]. El valor σ_f es la varianza de la señal y l es la escala de longitud. Estos son denominados hiperparámetros. Los hiperparámetros pueden ser optimizados al minimizar el logaritmo negativo de la verosimilitud marginal:

$$-\log p\left(\boldsymbol{y} | \boldsymbol{X}, \sigma_n, \sigma_f, l\right) = -\frac{1}{2} \boldsymbol{y} \boldsymbol{K}_n^{-1} \boldsymbol{y} - \frac{1}{2} \log \det \boldsymbol{K}_n - \frac{n}{2} \log 2\pi \qquad (3.7)$$

Los valores $\boldsymbol{X}$ y $\boldsymbol{y}$ son los datos de entrenamiento. Cuando el sistema ha aprendido se procede a desarrollar pruebas. Una entrada de prueba $\boldsymbol{x}^*$, actualiza la distribución:

$$\begin{bmatrix} \boldsymbol{y} \\ f(\boldsymbol{x}^*) \end{bmatrix} \sim \boldsymbol{\mathcal{N}}\left(\boldsymbol{0}, \begin{bmatrix} K(\boldsymbol{X},\boldsymbol{X}) + \sigma_n^2 \boldsymbol{I} & K(\boldsymbol{X},\boldsymbol{x}^*) \\ K(\boldsymbol{x}^*,\boldsymbol{X}) & K(\boldsymbol{x}^*,\boldsymbol{x}^*) + \sigma_n^2 \boldsymbol{I} \end{bmatrix}\right)$$

La distribución condicional de f también es una distribución normal:

$$f(\boldsymbol{x}^*) \,|\boldsymbol{X},\boldsymbol{y},\boldsymbol{x}^* \sim \boldsymbol{\mathcal{N}}\big(\mu(\boldsymbol{x}^*), \mathrm{var}(\boldsymbol{x}^*)\big)$$

Donde:

$$\mu(\boldsymbol{x}^*) = K(\boldsymbol{x}^*,\boldsymbol{X})\boldsymbol{K}_n^{-1}\boldsymbol{y} \tag{3.8}$$

$$\mathrm{var}(\boldsymbol{x}^*) = K(\boldsymbol{x}^*,\boldsymbol{x}^*) - K(\boldsymbol{x}^*,\boldsymbol{X})\boldsymbol{K}_n^{-1}K(\boldsymbol{X},\boldsymbol{x}^*) \tag{3.9}$$

El producto $\boldsymbol{K}_n^{-1}\boldsymbol{y}$ es comúnmente denominado vector de predicción y se denota como:

$$\boldsymbol{\alpha} = \boldsymbol{K}_n^{-1}\boldsymbol{y}$$

Alternativamente, utilizando la factorización de Cholesky:

$$\boldsymbol{K}_n = \boldsymbol{L}_n\boldsymbol{L}_n^T$$

Donde $\boldsymbol{L}_n$ es una matriz triangular inferior, se puede obtener el vector de predicción sin calcular la inversa de la matriz $\boldsymbol{K}_n$:

$$\boldsymbol{\alpha} = \boldsymbol{L}_n^T \backslash (\boldsymbol{L}_n \backslash \boldsymbol{y})$$

La operación $\boldsymbol{A}\backslash\boldsymbol{B}$ denota la solución $\boldsymbol{z}$ de la ecuación lineal $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{B}$.

3.3.2. Procesos gaussianos locales

Los procesos gaussianos son utilizados por el alto nivel de exactitud que presentan en problemas de regresión. Sin embargo, la principal limitación de este método es la complejidad de ejecución. La operación crítica para el desempeño de este método es la inversión de la matriz $\boldsymbol{K}_n$. La inversión de una matriz es una operación de complejidad $\mathcal{O}(n^3)$, donde n es el tamaño de la matriz. Esto significa que el tiempo de ejecución aumenta de manera cúbica respecto al tamaño de la matriz.

Un método utilizado para disminuir el costo computacional de la inversión se denomina procesos gaussianos locales. El principio de funcionamiento de este método es hacer las predicciones con base en diversos procesos gaussianos, multiplicados por un peso respectivo, con un límite de observaciones en cada proceso [19]. De esta manera las matrices por invertir tendrán un menor tamaño. Los pesos deben multiplicar los parámetros descritos en las ecuaciones (3.8) y (3.9), y así, se obtiene nuevamente una distribución normal [19].

De esta manera, la predicción y su respectiva varianza para p procesos gaussianos locales son [19]:

$$\mu_L(\boldsymbol{x}^*) = \sum_{j=1}^{p} W_j \mu_j(\boldsymbol{x}^*) \tag{3.10}$$

$$\mathrm{var}_L(\boldsymbol{x}^*) = \sum_{j=1}^{p} W_j \left(\mu_j(\boldsymbol{x}^*) + \mathrm{var}_j(\boldsymbol{x}^*)\right) - \left(\mu_L(\boldsymbol{x}^*)\right)^2 \tag{3.11}$$

Los valores $\mu_j(\boldsymbol{x}^*)$ y $\mathrm{var}_j(\boldsymbol{x}^*)$ se obtienen de las ecuaciones (3.8) y (3.9) respectivamente. Los valores W_j son los pesos. Algunas maneras de calcular los pesos son mediante la covariancia del valor de prueba y los valores del conjunto [20], o bien, la covariancia entre el punto de prueba y la media aritmética de los valores del conjunto [6] [21].

3.3.3. Procesos gaussianos locales divisibles

El método utilizado en este proyecto fue desarrollado con base en [22]. Para este método, los pesos están basados en distribuciones probabilísticas y la división de conjuntos mediante un árbol binario. Este método está constituido por dos partes: la actualización y la predicción.

Actualización

El fundamento del método descrito es la inserción y distribución de datos. Un conjunto debe dividirse cuando este llega al límite de datos. De esta manera se evita tener matrices de covariancia indefinidamente grandes conforme se mantiene una entrada constante de datos. Esto da lugar a una estructura de árbol binario. Un dato $(\boldsymbol{x}_n, y_n)$ es insertado en la raíz del árbol, si este nodo ya ha sido dividido y, por lo tanto, tiene hijos,

se obtiene la probabilidad de pertenencia, mediante las ecuaciones (3.15) y (3.16), para los hijos izquierdo y derecho de la raíz y se selecciona uno de estos de manera aleatoria basado en la probabilidad calculada. Se debe repetir el proceso de selección desarrollado en la raíz hasta encontrar un subconjunto que se encuentre en una hoja del árbol binario. Este proceso está ilustrado en la **Figura 3.3**.

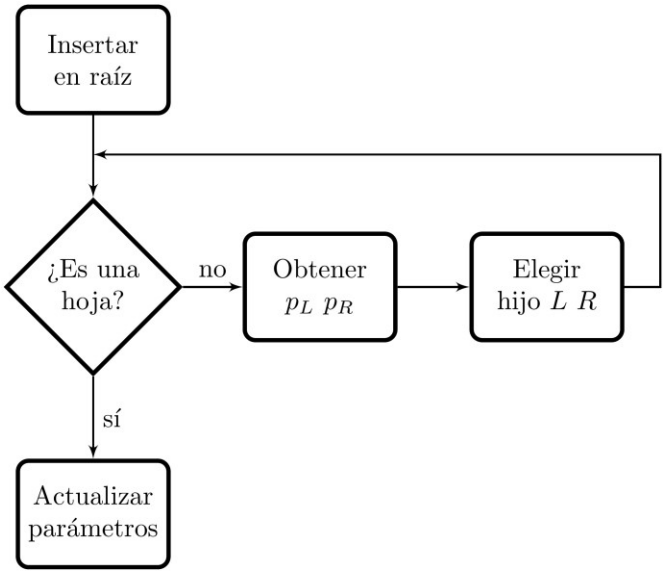

Figura 3.3. Diagrama de flujos utilizado para seleccionar la hoja en la que se inserta el nuevo dato.

Las actualizaciones de los parámetros deben desarrollarse de manera eficiente. Para las entradas y las salidas de cada conjunto basta con adjuntar el dato nuevo de entrenamiento $(\boldsymbol{x}_n, y_n)$ a los datos previamente entrenados $\{\boldsymbol{X}_o, \boldsymbol{y}_o\}$: $\boldsymbol{X} = [\boldsymbol{X}_o\ \boldsymbol{x}_n]$, $\boldsymbol{y} = [\boldsymbol{y}_o\ y_n]$. Se actualiza $\boldsymbol{K}_n$ considerando los nuevos datos sin tener que calcular la totalidad de la matriz. Para una matriz de covariancia previamente calculada $\boldsymbol{K}_o$, se desarrolla la actualización:

$$\boldsymbol{K}_n = \begin{bmatrix} \boldsymbol{K}_o & K(\boldsymbol{x}_n, \boldsymbol{X})^T \\ K(\boldsymbol{x}_n, \boldsymbol{X}) & K(\boldsymbol{x}_n, \boldsymbol{x}_n) + \sigma_n^2 \boldsymbol{I} \end{bmatrix}$$

De manera similar, para la matriz $\boldsymbol{L}_n$, teniendo previamente $\boldsymbol{L}_o$ [6]:

$$\boldsymbol{L}_n = \begin{bmatrix} \boldsymbol{L}_o & \boldsymbol{0} \\ \boldsymbol{l}_b & l_c \end{bmatrix}$$

Donde $\boldsymbol{l}_b = \boldsymbol{L}_o \backslash K(x_n, \boldsymbol{X})^T$ y $l_c = \sqrt{K(\boldsymbol{x}_n, \boldsymbol{x}_n) + \sigma_n^2 \boldsymbol{I} - \|\boldsymbol{l}_b\|}$

El proceso de división está ejemplificado en la **Figura 3.4**. Un conjunto se debe dividir cuando se llega al límite de datos. El espacio del conjunto se dividirá en tres regiones: la del hijo izquierdo, la del hijo derecho y la superpuesta. Los datos que se encuentran en la región superpuesta tendrán una probabilidad no nula de pertenecer a ambos y serán distribuidos de forma aleatoria según estas probabilidades.

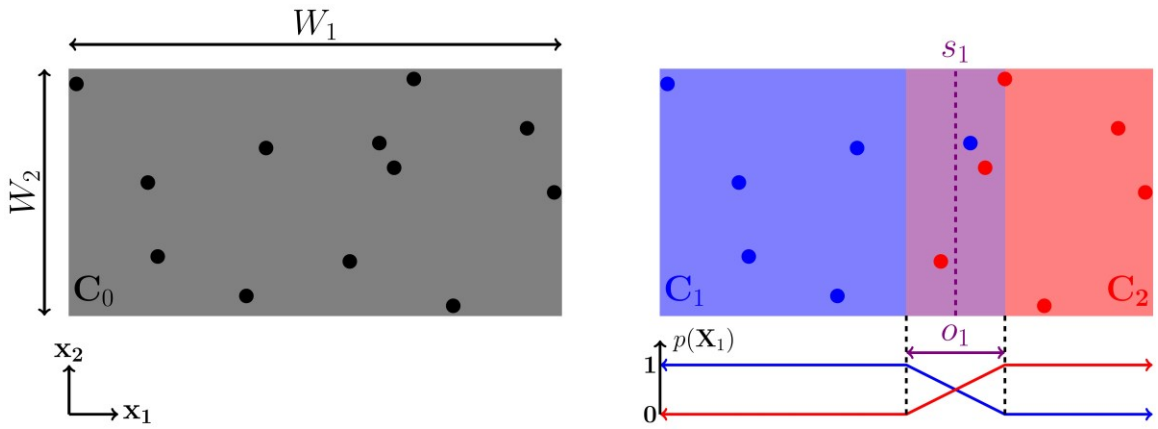

Figura 3.4. Proceso de división de un conjunto de datos. La región morada es la región superpuesta. Se ilustra la probabilidad de pertenencia. La división se lleva a cabo en el ancho W mayor.

La región superpuesta se define alrededor del hiperplano s_d, el cual representa el valor de la mediana M_e de los datos en la dimensión d donde ocurre la división:

$$s_d = M_e(\boldsymbol{X}_d) \tag{3.12}$$

Se utiliza la mediana para garantizar que la cantidad de datos en cada subconjunto sea similar. La dimensión en donde se hace la división es la que presenta un ancho máximo. El ancho está definido por:

$$W_d = \text{máx}(\boldsymbol{X}_d) - \text{mín}(\boldsymbol{X}_d) \tag{3.13}$$

El valor o_d, el cual representa el ancho de la región superpuesta, se obtiene mediante:

$$o_d = \frac{W_d}{m} \tag{3.14}$$

Donde W_d es el ancho del conjunto de datos en la dimensión d y m es una razón constante denominada factor de superposición.

La distribución de probabilidad del subconjunto izquierdo p_L y del subconjunto derecho p_R están dadas por funciones lineales saturadas:

$$p_L(\boldsymbol{X}_d) = \begin{cases} 1 & si \quad \boldsymbol{X}_d < s_d - \frac{o_d}{2} \\ \frac{1}{2} + \frac{\boldsymbol{X}_d - s_d}{o_d} & si \quad s_d - \frac{o_d}{2} < \boldsymbol{X}_d < s_d + \frac{o_d}{2} \\ 0 & si \quad s_d + \frac{o_d}{2} < \boldsymbol{X}_d \end{cases} \tag{3.15}$$

$$p_R(\boldsymbol{X}_d) = \begin{cases} 1 & si \quad s_d + \frac{o_d}{2} < \boldsymbol{X}_d \\ \frac{1}{2} + \frac{s_d - \boldsymbol{X}_d}{o_d} & si \quad s_d - \frac{o_d}{2} < \boldsymbol{X}_d < s_d + \frac{o_d}{2} \\ 0 & si \quad \boldsymbol{X}_d < s_d - \frac{o_d}{2} \end{cases} \tag{3.16}$$

Predicción

El motivo de utilizar los procesos gaussianos en este trabajo es la regresión. La predicción del método utilizado en este proyecto está basada en las ecuaciones (3.10) y (3.11). Los pesos utilizados para la predicción se obtienen mediante iteraciones de la distribución probabilística descrita por las ecuaciones (3.15) y (3.16).

La probabilidad que tiene un dato de seleccionar un nodo hijo se denomina probabilidad local. Conforme la búsqueda recorre los niveles en el árbol binario, se multiplica la probabilidad inicial con cada probabilidad local calculada. Para una hoja en el nivel t la probabilidad global del j-ésimo modelo local con probabilidad no nula se calcula mediante:

$$p_{G,j}(\boldsymbol{x}^*) = \prod_{i=1}^{t-1} p_{j,i}(\boldsymbol{x}^*) \tag{3.17}$$

Donde i es el nivel en que se encuentra la hoja, $p_{j,i}$ es la probabilidad local respectiva al nivel i del modelo j. El método descrito en la ecuación (3.17) está ilustrado en la **Figura 3.5**.

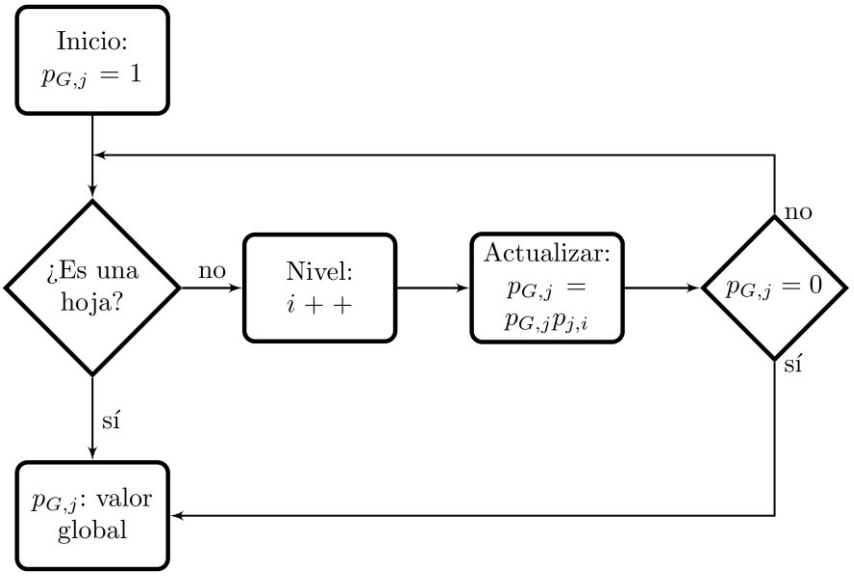

Figura 3.5. Diagrama de flujos para obtener la probabilidad global de j-ésimo modelo local.

De esta manera, se tiene el conjunto $\mathbb{U}$ de todas las hojas cuya probabilidad global es distinta de cero. La suma de estas probabilidades es igual a 1. La búsqueda en el árbol binario de los modelos activos disminuye el tiempo de ejecución pues no se debe de obtener predicción para los que tienen probabilidad global nula.

Las predicciones $f_D(\boldsymbol{x}^*|\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}^*)$ tienen, por lo tanto:

$$\mu_D(\boldsymbol{x}^*) = \sum_{j \in \mathbb{U}} p_{G,j}(\boldsymbol{x}^*)\mu_j(\boldsymbol{x}^*) \tag{3.18}$$

$$\mathrm{var}_D(\boldsymbol{x}^*) = \sum_{j \in \mathbb{U}} p_{G,j}(\boldsymbol{x}^*)\left(\mu_j(\boldsymbol{x}^*) + \mathrm{var}_j(\boldsymbol{x}^*)\right) - \left(\mu_D(\boldsymbol{x}^*)\right)^2 \tag{3.19}$$

La verosimilitud es:

$$p_D(f(\boldsymbol{x}^*)|\boldsymbol{x}^*, \boldsymbol{X}, \boldsymbol{y}) = \sum_{j \in \mathbb{U}} p_{G,j}(\boldsymbol{x}^*)p_j\left(\boldsymbol{y}|\boldsymbol{X}, \sigma_n, \sigma_f, l\right)$$

La **Figura 3.6** ilustra un ejemplo simple de predicción para un árbol con tres niveles y cuatro hojas.

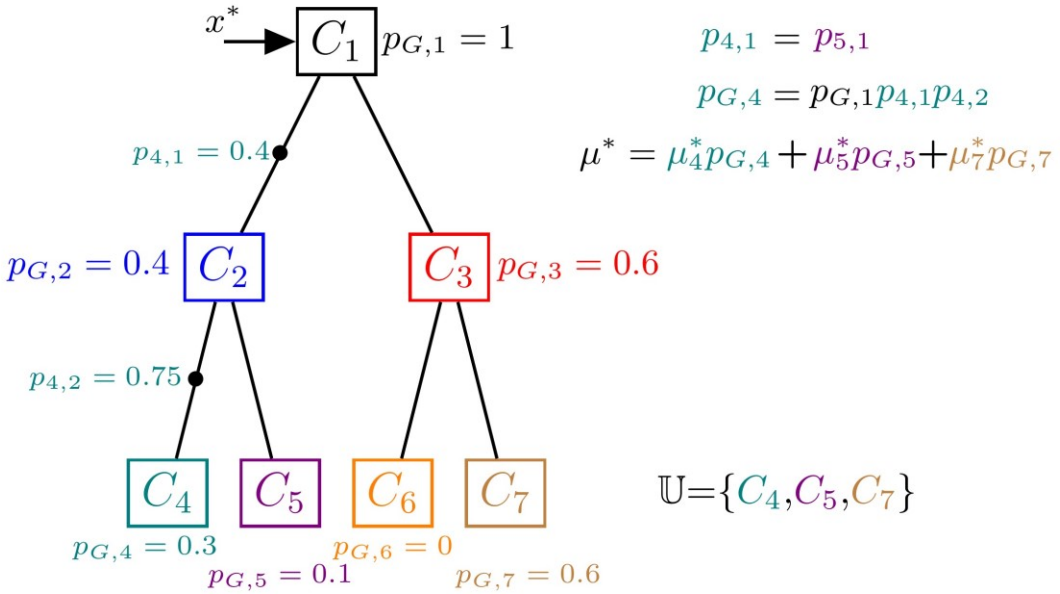

Figura 3.6. Ejemplo de estructura del método de procesos gaussianos locales utilizado. Se expone la obtención de las probabilidades globales, la predicción y el conjunto utilizado para predecir.

Capítulo 4. Implementación

En este capítulo se explica la implementación del control y el manipulado. Se describe los ambientes en los que se desarrolla las pruebas: en simulación y con un manipulador SCARA. En la sección 4.1 se describe el modelo matemático ideal de un manipulador de dos grados de libertad utilizado en simulaciones. La sección 4.2 describe el diseño del algoritmo para el bloque de procesos gaussianos locales divisibles y las consideraciones para que este pueda funcionar en tiempo real. En la sección 4.3 se ilustra el bloque resultante y la interfaz. La implementación del controlador es descrita en la sección 4.4. Finalmente, en la sección 4.5 se describe las consideraciones para la implementación en la planta física respecto con la simulación.

4.1. Modelo del manipulador

El modelo matemático aproximado del manipulador es desarrollado para llevar a cabo simulaciones del controlador. La ecuación (3.3) es utilizada para modelar el comportamiento del manipulador, sin embargo, para la simulación se requiere la dinámica directa. Por lo tanto, se ajusta la ecuación a:

$$\ddot{\theta} = \boldsymbol{M}^{-1}(\theta)\left(\boldsymbol{\tau} - \boldsymbol{V}(\theta,\dot{\theta})\dot{\theta} - \boldsymbol{G}(\theta)\right) \tag{4.1}$$

Los valores iniciales de posición θ, y velocidad $\dot{\theta}$ deben ser definidos previamente según los requerimientos del experimento y el torque $\boldsymbol{\tau}$ debe calculado por el control. La implementación desarrollada en Simulink está basada en el diagrama de la **Figura 4.1**. Las constantes físicas del manipulador son seleccionadas de manera aleatoria. Se agrega un error en el sistema aproximado para simular la desviación entre el modelo matemático y la planta física.

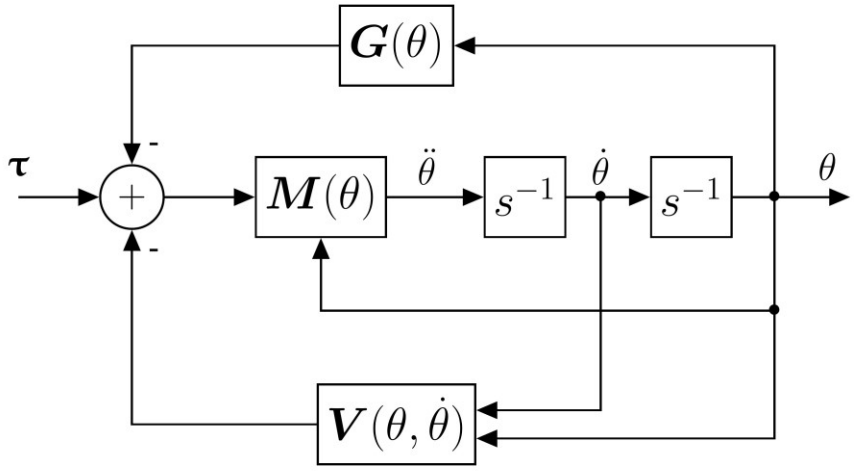

Figura 4.1. Diagrama de bloques utilizado para simular el comportamiento del manipulador.

4.2. Diseño del código

El programa es diseñado para funcionar en MATLAB Simulink. Para poder implementar instrucciones complejas en el ambiente de Simulink se requiere un bloque de sistema. Utilizando este bloque se puede desarrollar una clase del método de regresión descrito en la sección 3.3.3. Se plantea desarrollar un bloque de sistema flexible y con diversos parámetros definidos por el usuario para que este pueda ser utilizado en diversas variaciones del proyecto y hasta en otros experimentos.

Aunque el bloque de Simulink permite desarrollar código con la misma sintaxis que MATLAB, la funcionalidad es reducida. Le ejecución puede ser mediante la misma ejecución que tiene MATLAB o bien se puede generar el código en C y ejecutar este. Generalmente, la generación de código es la opción más eficiente, sin embargo, esta más restrictiva. El diseño del método se desarrolla para poder generar código, por lo tanto, este considera algunas limitaciones. Las limitaciones principales serán descritas en los siguientes párrafos.

La memoria en donde se almacena los parámetros debe ser estática. Esta debe ser definida una única vez durante el inicio de la ejecución. Esto da lugar a un nuevo parámetro definido por el usuario: la cantidad máxima de hojas en el árbol binario. A partir de este valor se define el espacio necesario para la cantidad máxima de datos $\mathcal{D}$

aprendidos y los parámetros que corresponden: matrices de covariancia $\boldsymbol{K}_n$, factores de Cholesky $\boldsymbol{L}_n$ y vector de predicción $\boldsymbol{\alpha}$.

La cantidad de funciones compatibles en el bloque de sistema es menor que el total de MATLAB. Si una función no puede ser utilizada se debe desarrollar una alternativa sin comprometer significativamente el desempeño. Asimismo, el método no admite recursión y se debe utilizar ciclos para recorrer el árbol binario y para desarrollar divisiones.

Se debe tomar en cuenta que bloque debe tener algunos métodos esenciales para el funcionamiento de este. En estos se debe definir la cantidad de entradas y salidas, el tamaño da las salidas, el tipo de dato y la inicialización. Estas configuraciones limitan la flexibilidad del método y de las salidas que este puede tener. Adicionalmente, los parámetros deben ser de acceso protegido, por lo tanto, estos no pueden ser exportados después de una ejecución.

La última limitación considerada al diseñar el código para ejecutar el paradigma de aprendizaje máquina fue la respuesta ante la limitación de datos. Aunque teóricamente el sistema puede aprender datos de manera indefinida, en el desarrollo del código se debe considerar la limitación de memoria. Por lo tanto, se debe desarrollar instrucciones que consideren el límite de información sin alterar el funcionamiento. Es decir, cuando el sistema llega al límite de aprendizaje, el sistema deberá descartar los datos y poder predecir con base en los datos aprendidos.

La clase es diseñada para tener los siguientes parámetros seleccionados por el usuario según la conveniencia de la implementación planteada:

1. Cantidad máxima de hojas que un sistema puede llegar a tener. Cuando se llega al límite de hojas, los nodos no se podrán dividir y la información nueva es descartada.
2. Límite de datos para un modelo local. Este parámetro determina el momento en que un modelo debe dividirse.
3. Tamaño de la entrada. Como fue descrito en la sección 3.3.1, el vector de entrada $\boldsymbol{x}_i \in \mathbb{R}^k$. El valor de k debe ser definido por el usuario según la estructura de la entrada.

4. Cálculo del hiperplano s_d. Aunque en la ecuación (3.12), el hiperplano es calculado mediante la mediana, se implementa dos variaciones basadas en la media aritmética M_a:

$$s_d = M_a(\boldsymbol{X}_d) \tag{4.2}$$

$$s_d = M_a(\text{máx}(\boldsymbol{X}_d)\,, \text{mín}(\boldsymbol{X}_d)) \tag{4.3}$$

5. El factor de superposición m. La razón entre el ancho de la dimensión donde sucede la división y la región superpuesta. Tener un valor alto de m resulta en regiones superpuestas pequeñas.
6. Los hiperparámetros σ_f, l y la desviación estándar del ruido σ_n. Los valores utilizados para caracterizar el comportamiento de la función de predicción. Asimismo, una variable lógica que permita:
    a. Cargar los valores desde un archivo previamente desarrollado por el usuario.
    b. Definir los hiperparámetros por el usuario en el momento de la ejecución.
7. Tiempo de muestreo. Se define la rapidez con la que se requiere que el método se actualice y entregue una predicción.

4.3. Bloque de Simulink

El ambiente Simulink es utilizado en este proyecto para conectar en tiempo real al manipulador robótico con un computador. Utilizando esta plataforma, se puede implementar el código desarrollado en la sección 4.2 en el controlador. Se diseña el bloque de manera que sea sencillo de utilizar para el usuario y que los parámetros sean fácilmente afinados. El bloque resultante está ilustrado en la **Figura 4.2**.

Figura 4.2. Bloque de procesos gaussianos locales divisibles utilizado en Simulink.

La ventana de ajuste de parámetros está ilustrada en la **Figura 4.3**. Estos parámetros están descritos en la sección 4.2.

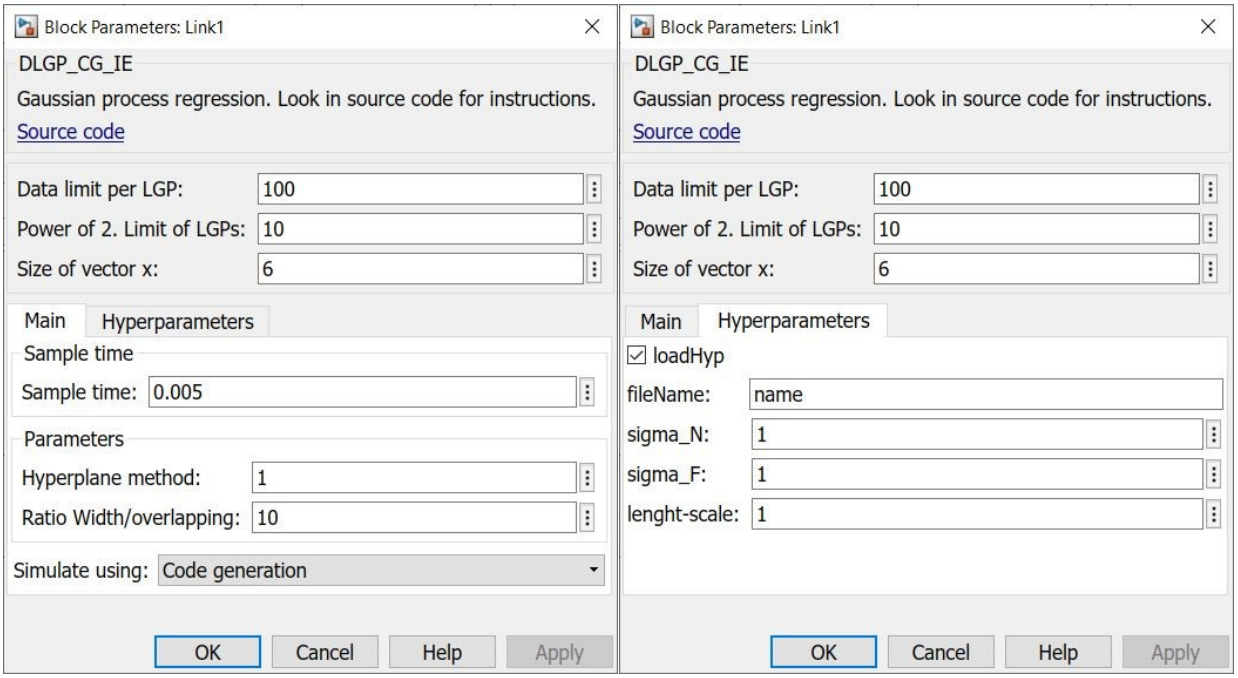

Figura 4.3. Ventana de selección de parámetros para el bloque de procesos gaussianos locales divisibles.

4.4. Implementación del controlador

El controlador implementado está basado en una adaptación en tiempo real de [5]. En la **Figura 4.4** se observa el controlador implementado, así como la distribución de los bloques según el tipo de aporte a la señal de control.

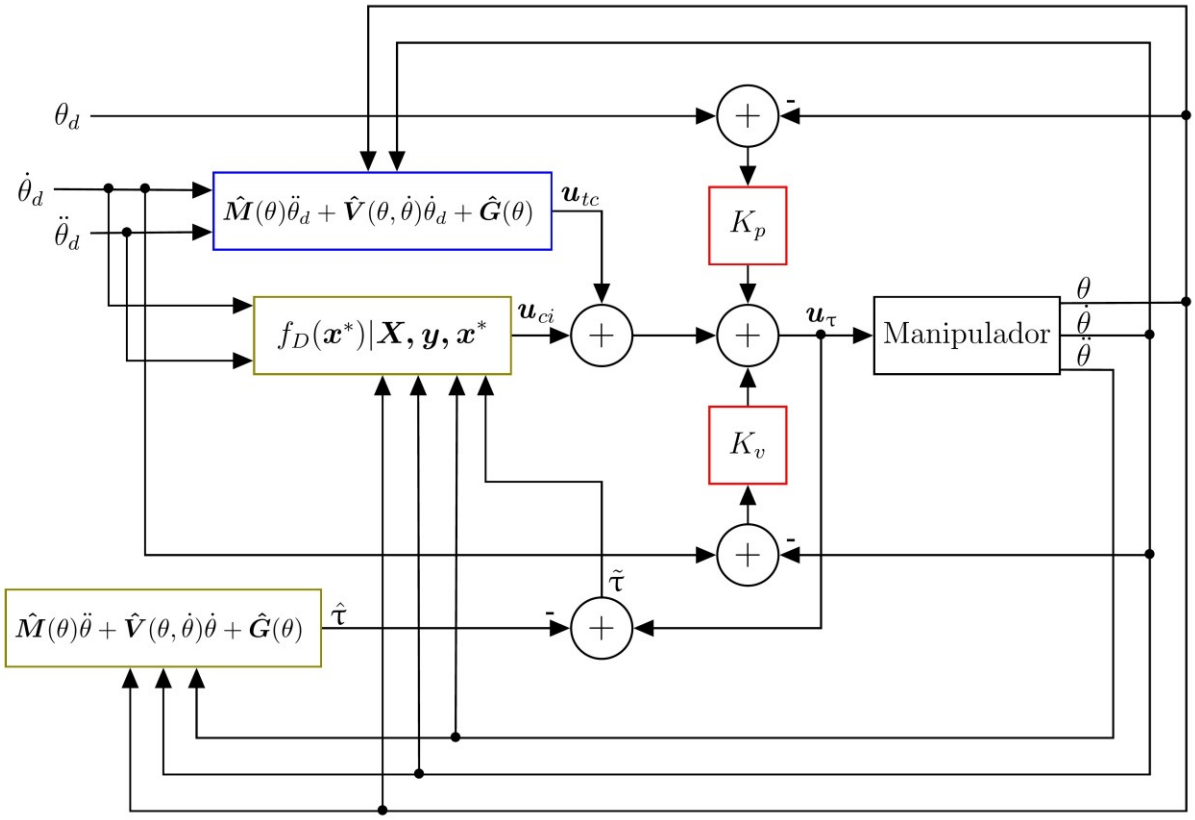

Figura 4.4. Controlador implementado mediante aprendizaje máquina (en color oliva), control proporcional derivativo (rojo) y torque calculado (azul).

Se utiliza un modelo aproximado para definir el valor de salida con el que se pretende entrenar el sistema. Se utiliza los mismos parámetros de la ecuación (3.5), sin embargo, no se multiplica por los valores deseados de aceleración y velocidad, sino por los valores reales:

$$\hat{\tau} = \widehat{\boldsymbol{M}}(\theta)\ddot{\theta} + \widehat{\boldsymbol{V}}\left(\theta, \dot{\theta}\right)\dot{\theta} + \widehat{\boldsymbol{G}}(\theta) \tag{4.4}$$

Utilizando el resultado de la ecuación (4.6) se puede definir la diferencia entre la señal de entrada, la cual considera el control de torque calculado, el control PD, el control inteligente, y el valor aproximado:

$$\tilde{\tau} = \boldsymbol{u}_\tau - \hat{\tau} \tag{4.5}$$

Se desarrolla la función de los procesos gaussianos locales divisibles $f_D(\boldsymbol{x}^*|\boldsymbol{X},\boldsymbol{y},\boldsymbol{x}^*)$. El vector de entrada para entrenamiento, así como el valor de salida están definidos por las ecuaciones:

$$\boldsymbol{x}_i = \left[\theta^T \;\; \dot{\theta}^T \;\; \ddot{\theta}^T\right]_i \tag{4.6}$$

$$y_i = [\tilde{\tau}]_i \tag{4.7}$$

El vector de predicción está dado por:

$$\boldsymbol{x}_i^* = \left[\theta^T \;\; \dot{\theta}_d^T \;\; \ddot{\theta}_d^T\right]_i$$

La señal de control generada por el bloque de procesos gaussianos locales divisibles es:

$$\boldsymbol{u_{ci}} = f_D(\boldsymbol{x}_i^*|\boldsymbol{X},\boldsymbol{y},\boldsymbol{x}_i^*)$$

Por lo tanto, la ecuación (3.6) puede ser reescrita como:

$$\boldsymbol{u}_\tau = \boldsymbol{e}k_p + \dot{\boldsymbol{e}}k_v + \widehat{\boldsymbol{M}}(\theta)\ddot{\theta}_d + \widehat{\boldsymbol{V}}\left(\theta,\dot{\theta}\right)\dot{\theta}_d + \widehat{\boldsymbol{G}}(\theta) + f_D(\boldsymbol{x}_i^*|\boldsymbol{X},\boldsymbol{y},\boldsymbol{x}_i^*) \tag{4.8}$$

4.5. Consideraciones del sistema real

La implementación del controlador para el sistema físico tiene algunas consideraciones por tomar en cuenta respecto al controlador de las simulaciones. Se debe definir el tiempo de muestreo de los bloques de aprendizaje máquina considerando el hardware y software disponible. Si se designa un valor muy alto de tiempo de muestro el manipulador generará vibraciones y no el aprendizaje no será apropiado. Sin embargo, si el tiempo de muestreo es un valor muy bajo el computador encargado del control podrá desfasarse con el comportamiento del manipulador.

La capacidad del controlador de sincronizarse con el manipulador depende, a su vez, de los parámetros seleccionados. La cantidad de datos permitidos en un modelo local antes de dividirse, así como el factor de superposición afectan el tiempo de ejecución del programa, sin embargo, si los parámetros no son óptimos el error puede disminuir considerablemente. Adicionalmente, se debe definir una cantidad de puntos de

aprendizaje en la que se pueda garantizar que el tiempo de predicción pueda cumplir con el tiempo de muestreo.

El manipulador, al ser un sistema físico real, tiene limitaciones respecto al torque de entrada y al movimiento. Por lo tanto, se debe diseñar las pruebas de manera que los movimientos no excedan los límites. Asimismo, el controlador debe tener un bloque de saturación para que los valores entregados de torque no representen un riesgo para el equipo. Aproximarse a los límites del manipulador puede resultar en un peor desempeño del controlador.

Los valores de la señal de salida de los bloques de aprendizaje máquina pueden tener cambios bruscos que, a su vez, pueden causar cambios bruscos en la señal de entrada al manipulador. Este comportamiento es mitigado mediante un filtro de media en la salida de los bloques de procesos gaussianos. La cantidad de datos utilizados en el cálculo de la salida del filtro es definida de manera empírica según el comportamiento del manipulador.

El aprendizaje se realiza a partir de las condiciones de movimiento que tiene el manipulador. El sistema no puede responder ante las condiciones conforme estas ingresan porque este debe cumplir un tiempo de muestreo, por lo tanto, el resultado generado por los bloques de aprendizaje corresponde a las condiciones de movimiento del lapso de muestreo anterior. En la implementación física esto debe ser corregido agregando un retraso en la señal de salida de los bloques de aprendizaje. Esto resulta en una fuente de error para el sistema, sin embargo, para valores menores de tiempo de muestreo, este error se torna menos significativo.

Capítulo 5. Resultados y análisis

En este capítulo se describe las evaluaciones que se realizan tanto para el algoritmo de aprendizaje máquina como para el controlador que lo incorpora. En la sección 5.1 se analiza el comportamiento del aprendizaje ante un conjunto de datos de un manipulador SARCOS de siete grados de libertad relacionados al control mediante torque del sistema; el efecto de los parámetros es analizado. En la sección 5.2 se compara el método de aprendizaje ante otros métodos utilizando la misma evaluación de la sección 5.1. En la sección 5.3 se presenta el desempeño del control en simulación para un manipulador SCARA de dos grados de libertad. Finalmente, en la sección 5.4 se presenta los resultados de la evaluación en el sistema físico.

5.1. Parámetros del método

El primer paso desarrollado en experimentación es el análisis de los parámetros y el efecto que estos tienen en el desempeño del aprendizaje. Estas pruebas son desarrolladas mediante simulación y utilizando datos reales de manipuladores robóticos. Los resultados expuestos en esta sección representan el comportamiento del método ante datos del sistema de trabajo (manipulador de dos grados de libertad) o sistemas similares (manipulador de siete grados de libertad), por lo tanto, otros sistemas, que puedan utilizar este método de aprendizaje, no necesariamente tendrán un comportamiento similar.

El conjunto de datos[1] que se expone en esta sección es el que fue utilizado en [18]. Estos están compuestos por una entrada de dimensión 21 (la posición, velocidad y aceleración angular de las siete articulaciones) y los siete torques correspondientes a esa entrada. El conjunto de datos está compuesto por 44484 datos de entrenamiento y 4449 datos de prueba. Todos los experimentos son desarrollados siguiendo los pasos:

1. Dividir los datos en 100 subconjuntos equidistantemente distanciados.
2. Entrenar el primer subconjunto.

[1] Disponible en http://www.gaussianprocess.org/gpml/data/

3. Evaluar todos los puntos de prueba y almacenar resultados.
4. Repetir los pasos 2 y 3 para todos los demás subconjuntos.

Los resultados son analizados en términos de:

- Error: se utiliza el error medio cuadrado y se normaliza con la varianza de las salidas:

$$nMSE = \frac{\sum_{i=1}^{k}\left(\boldsymbol{y_{p,i}} - \boldsymbol{y_{d,i}}\right)^2}{k\left(\mathrm{var}(y_d)\right)} \tag{5.1}$$

Donde y_p son las predicciones del algoritmo y y_d son los valores reales dados en el conjunto de datos.

- Tiempo de actualización (t_{update}): tiempo que requiere el algoritmo para aprender un nuevo dato.
- Tiempo de predicción (t_{pred}): tiempo que requiere el algoritmo para predecir un dato a partir de una entrada.
- Logaritmo negativo de verosimilitud (NLL): valor que representa la bondad de ajuste el modelo a los datos. Se utiliza el valor negativo porque al optimizar los hiperparámetros los métodos buscan mínimos locales.
- Modelos de predicción (M): cantidad máxima de modelos locales utilizados para una predicción.
- Porcentaje de superposición (O_D): porcentaje de datos que se encuentran en alguna región de superposición.
- Divisiones (N_D): cantidad de divisiones que se llevan a cabo en una prueba.

5.1.1. Hiperparámetros

Los hiperparámetros son determinados al maximizar la ecuación (3.7). Se utiliza 100 iteraciones, con los primeros 500 datos, del método Rprop descrito en [23], y expuesto en el Anexo A, para obtener los hiperparámetros debido a la facilidad de implementación y a los buenos resultados que muestran respecto a otros métodos. Se lleva a cabo la comparación del desempeño al utilizar hiperparámetros optimizados respecto a hiperparámetros con valor unitario.

Las pruebas son desarrolladas utilizando los mismos parámetros (factor de superposición $m = 20$, límite de datos: 100, cálculo del hiperplano s_D mediante la ecuación (4.3)). Los resultados de error y verosimilitud para la primera articulación son ilustrados en la **Figura 5.1**. Para esta prueba se describe únicamente estas variables porque son las únicas que dependen de los hiperparámetros.

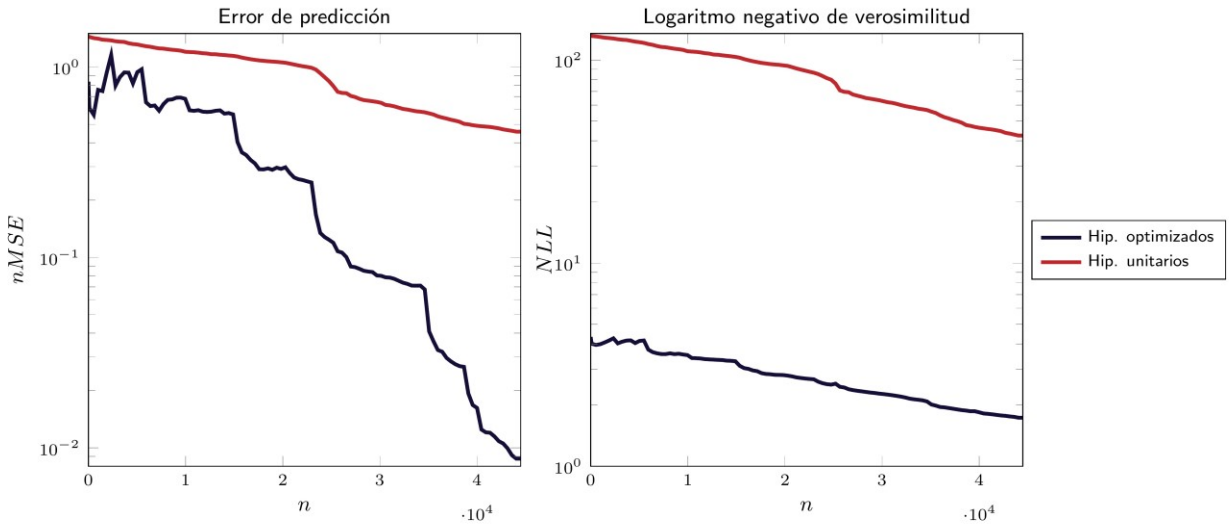

Figura 5.1. Resultados de las pruebas desarrolladas con hiperparámetros optimizados e hiperparámetros unitarios para la articulación J_1 respecto a n datos aprendidos.

Es evidente, mediante la **Figura 5.1**, que tener hiperparámetros óptimos mejora el desempeño del aprendizaje respecto al error. Asimismo, la verosimilitud permite identificar que el modelo optimizado se ajusta mejor al conjunto de datos desde el inicio del aprendizaje. Se debe destacar que, aunque el desempeño de los hiperparámetros unitarios es deficiente respecto a su contraparte, es evidente que el error y el NLL disminuyen con respecto a los datos aprendidos. Esto evidencia que, aunque los hiperparámetros pueden mejorar el desempeño, tener hiperparámetros no óptimos también resulta en aprendizaje [18] y disminución del error y el NLL. Por lo tanto, ante una infinita cantidad de datos aprendidos, ambos casos descritos en esta sección tendrían un desempeño similar.

Los resultados finales de las demás articulaciones están compilados en la **Tabla 5.1**. Es evidente que, en todas las articulaciones, el error y el NLL son menores cuando se tiene hiperparámetros optimizados. En todas las siguientes experimentaciones, los

hiperparámetros utilizados son los obtenidos mediante la optimización descrita en esta sección.

Tabla 5.1. Resultados comparativos del desempeño del aprendizaje según los hiperparámetros para cada articulación J. El texto en negrita denota los mejores resultados.

Articulación	Hip. Optimizados		Hip. Unitarios	
	$nMSE$	NLL	$nMSE$	NLL
J_1	**0.009**	**1.749**	0.458	42.450
J_2	**0.007**	**1.245**	1.087	52.464
J_3	**0.007**	**0.534**	0.710	16.307
J_4	**0.002**	**0.613**	1.520	61.026
J_5	**0.010**	**-1.048**	0.478	1.268
J_6	**0.010**	**-0.781**	0.387	1.452
J_7	**0.004**	**-0.724**	1.119	2.911

5.1.2. Hiperplano

En la sección 4.2 se describe las posibles variaciones a la ecuación (3.12), la cual consiste en la mediana, para el cálculo del hiperplano. En esta sección se describe el desempeño que cada método tiene en los resultados. Se utiliza un límite de 100 datos previo a la división y un factor de superposición $m = 20$.

Los resultados según tiempo de predicción y actualización para los diversos métodos están ilustrados en la Figura 5.2. Se evidencia que los tiempos de ejecución son similares. El tiempo de actualización es prácticamente constante y el tiempo de predicción aumenta lentamente en forma logarítmica. Sin embargo, el tiempo de predicción requerido por la mediana M_e es visiblemente mayor que los otros métodos a partir de los $3\text{x}10^4$ datos aprendidos. La

Tabla **5.2** muestra que, aunque los tiempos son similares, el método más rápido es $M_a(\text{máx mín})$.

Tabla 5.2. Tiempo promedio de predicción y actualización según el método de selección del hiperplano.

Método	t_{update} (µs)	t_{pred} (µs)	$t_{update} + t_{pred}$ (µs)
M_e	164.08	101.24	265.33
M_a	161.59	90.68	252.28
$M_a(\text{máx mín})$	160.28	87.73	248.02

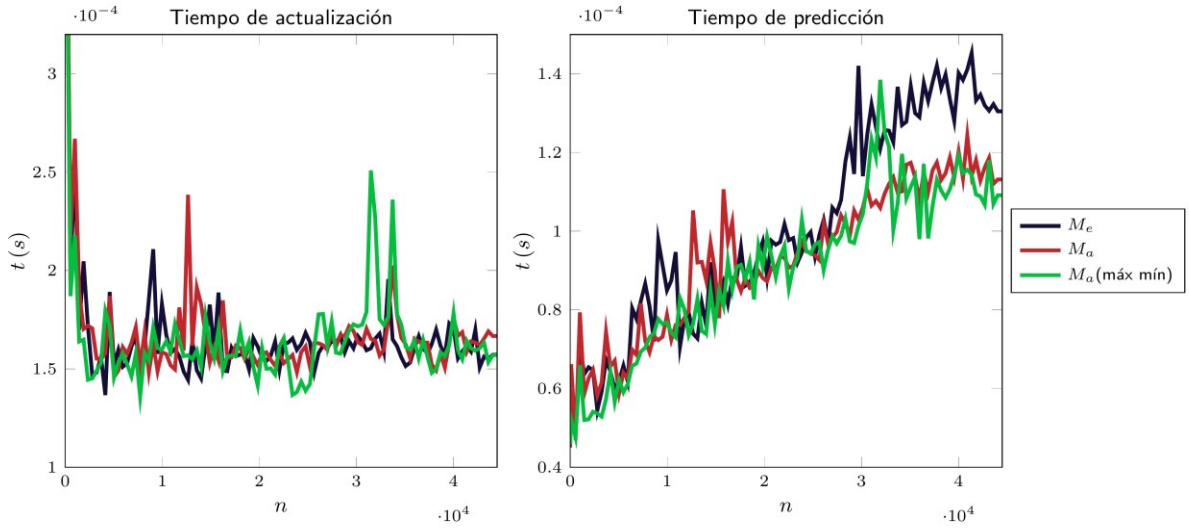

Figura 5.2. Tiempo de actualización y predicción durante el aprendizaje ante distintas maneras de calcular el hiperplano para la articulación J_1.

La razón por la cual el uso de $M_a(\text{máx mín})$ es más rápido se debe a dos razones. El cálculo de la media de dos valores es menos costoso que el de la media de todos los datos y menos aún que el de la mediana. Además, como se observa en la **Figura 5.3**, la cantidad de datos en regiones de superposición es menor. Esto resulta en menos procesos locales para la predicción y, por lo tanto, una predicción más rápida. Sin embargo, al tener una mayor cantidad de divisiones el tiempo de actualización aumenta.

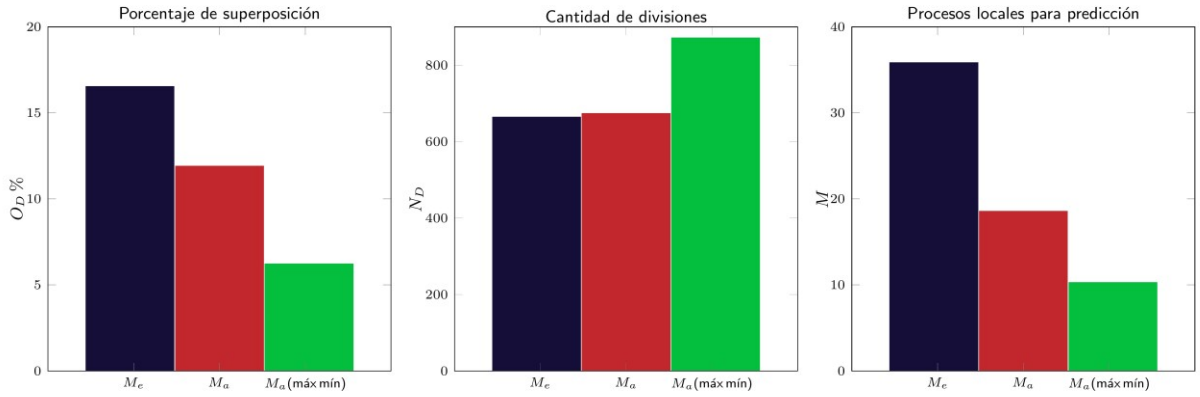

Figura 5.3. Efecto de los distintos métodos de hiperplano en la el desempeño.

Tabla 5.3. Resultados comparativos del desempeño del aprendizaje según el método de hiperplano para cada articulación J. El texto en negrita denota los mejores resultados.

Articulación	M_e		M_a		$M_a(\text{máx mín})$	
	$nMSE$	NLL	$nMSE$	NLL	$nMSE$	NLL
J_1	0.011	1.811	**0.009**	1.765	0.009	**1.749**
J_2	0.009	1.334	0.008	1.297	**0.007**	**1.245**
J_3	0.008	0.639	0.008	0.596	**0.007**	**0.534**
J_4	0.002	0.673	**0.002**	0.641	0.002	**0.613**
J_5	0.012	-0.983	0.012	-1.006	**0.010**	**-1.048**
J_6	0.015	-0.652	0.013	-0.712	**0.010**	**-0.781**
J_7	0.005	-0.645	0.004	-0.678	**0.004**	**-0.724**

La **Tabla 5.3** muestra que el método $M_a(\text{máx mín})$ tiene el mejor desempeño respecto al error y verosimilitud. Asimismo, este método fue el de mayor rapidez. Por lo tanto, todos los experimentos desarrollados en las siguientes secciones utilizan $M_a(\text{máx mín})$ para obtener el hiperplano.

5.1.3. Factor de superposición

El factor de posición se refiere a la razón que hay entre el ancho del subconjunto respecto al ancho de superposición en la dimensión en que se hará la división. Un factor de

superposición menor significa regiones de superposición más amplias. Esto resulta en una mayor probabilidad de pertenecer a múltiples modelos locales. Las evaluaciones descritas en esta sección son desarrolladas utilizando un límite de 100 datos previo a división y variando el factor de superposición.

En la **Figura 5.4** se observa el comportamiento del error y el tiempo de predicción ante diferentes factores de superposición. Se observa que un menor valor resulta en mayor tiempo de predicción. Esto se debe a que, como se observa en la **Figura 5.5**, más datos pertenecen a las regiones de superposición. Asimismo, esto resulta en una mayor cantidad de modelos locales utilizados en la predicción. Se debe tomar en cuenta que la variación del factor de superposición, como se observa en la **Tabla 5.4**, no afecta significativamente el tiempo de actualización.

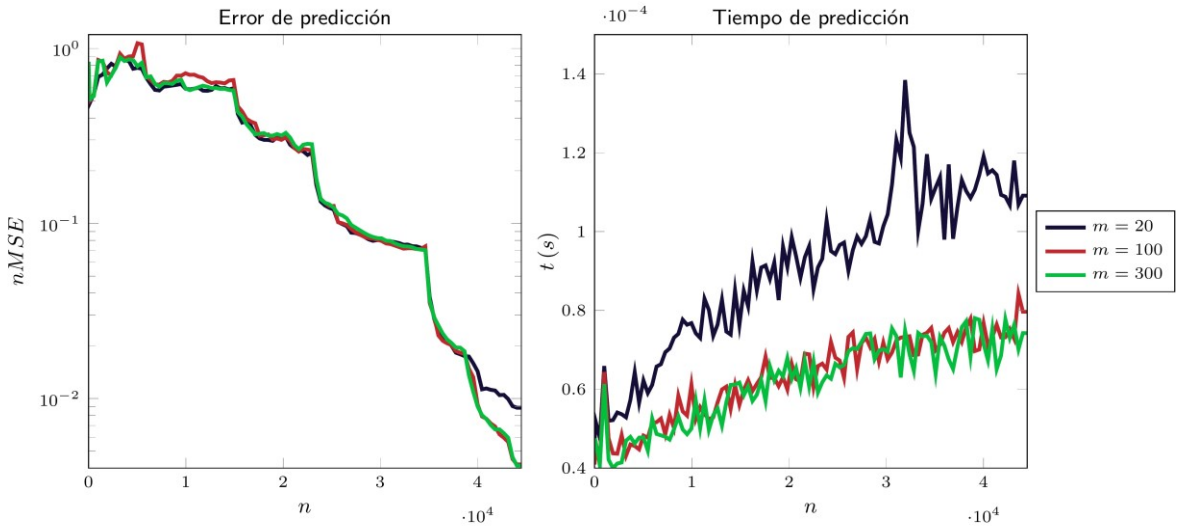

Figura 5.4. Resultados del error y el tiempo de predicción según el factor de superposición para la articulación J_1.

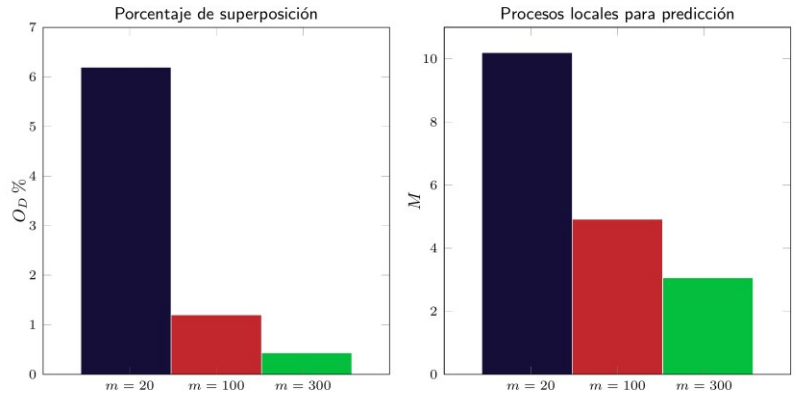

Figura 5.5. Efecto de distintos factores de superposición en el porcentaje de superposición y el promedio del máximo número de procesos gaussianos locales utilizados en la predicción.

La **Figura 5.4** demuestra también que tener una mayor cantidad de datos en las regiones de superposición no necesariamente resulta en un mejor desempeño, porque el error de $m = 20$ es visiblemente mayor que el de los otros valores. La **Tabla 5.5** muestra los resultados relacionados al error y al NLL finales. Es evidente, tanto por rapidez como por error y verosimilitud que el factor $m = 300$ tiene un mejor desempeño que los otros factores. Los resultados expuestos en el resto de este capítulo utilizan el factor $m = 300$.

Tabla 5.4. Tiempo promedio de predicción y actualización según el factor de superposición.

Factor	t_{update} (μs)	t_{pred} (μs)	$t_{update} + t_{pred}$ (μs)
$m = 20$	160.28	87.73	248.02
$m = 100$	158.54	64.00	222.54
$m = 300$	157.63	61.65	219.28

Tabla 5.5. Resultados comparativos del desempeño del aprendizaje según el factor de superposición para cada articulación J. El texto en negrita denota los mejores resultados.

Articulación	$m = 20$		$m = 100$		$m = 300$	
	$nMSE$	NLL	$nMSE$	NLL	$nMSE$	NLL
J_1	0.009	1.749	0.004	1.640	**0.004**	**1.608**
J_2	0.007	1.245	0.004	1.154	**0.002**	**1.129**
J_3	0.007	0.534	0.002	0.414	**0.001**	**0.383**
J_4	0.002	0.613	0.0008	0.506	**0.0005**	**0.481**
J_5	0.010	-1.048	0.006	-1.135	**0.005**	**-1.149**
J_6	0.010	-0.781	0.005	-0.890	**0.003**	**-0.920**
J_7	0.004	-0.724	0.002	-0.819	**0.001**	**-0.834**

5.1.4. Límite de datos

El límite de datos denota cuándo un sistema debe de dividirse. Al tener un límite de datos mayor las divisiones serán menos frecuentes y cada modelo tendrá más datos relacionados al valor por predecir, sin embargo, las divisiones, así como las actualizaciones tendrán un costo computacional mayor.

Los resultados respecto al error y al NLL según el límite de datos están expuestos en la **Figura 5.6**. Se observa que, cuando el aprendizaje finaliza, el límite de 50 tiene menor error y NLL. Sin embargo, durante el aprendizaje, hay valores donde el menor error resulta del límite 100 o del límite 300.

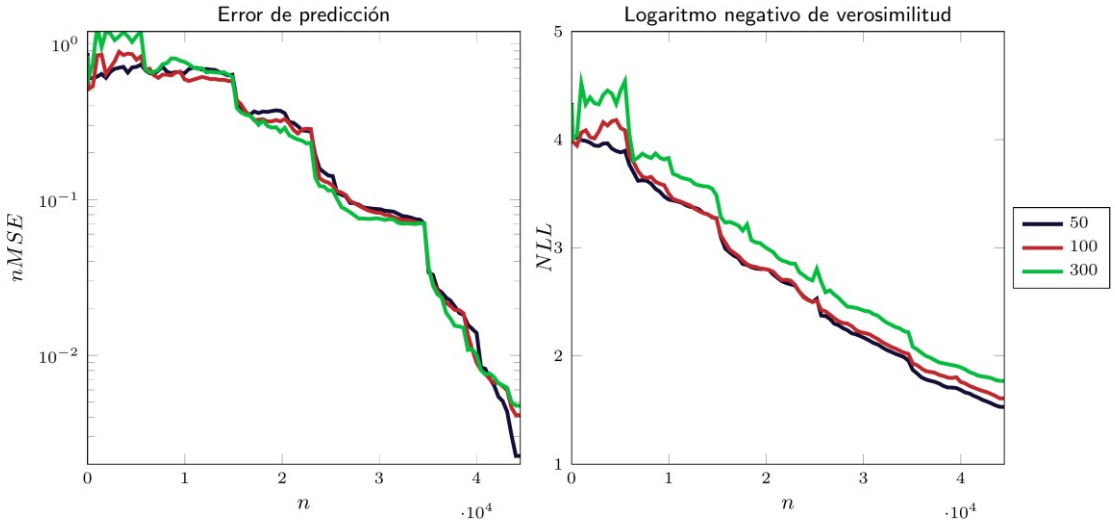

Figura 5.6. Error y de predicción y vesoimilitud del sistema al variar el límite de datos.

Tener una mayor cantidad de límite de datos, según evidencia esta experimentación, no necesariamente resulta beneficioso en el desempeño respecto al error. Esto puede suceder porque, al aceptar menos datos en cada proceso local, estos tienen una correlación mayor que con una mayor cantidad de puntos. Sin embargo, tener un menor límite de datos puede también resultar en un peor desempeño del error. La naturaleza de los datos de estudio influencia el desempeño respecto al límite de datos.

En la **Figura 5.7** se observa el desempeño en términos de tiempo de la evaluación. Es evidente que, tanto para predicción como para actualización, un menor límite de datos resulta en una ejecución más rápida. En la actualización, esto se debe a que determinar la matriz de covarianza, así como los respectivos factores de Cholesky, tiene un menor costo computacional. Como el límite de datos no incide fundamentalmente en la cantidad de procesos locales utilizados en actualización, el cálculo de la covarianza del punto de prueba, así como la predicción resultan en un menor consumo de recursos computacionales.

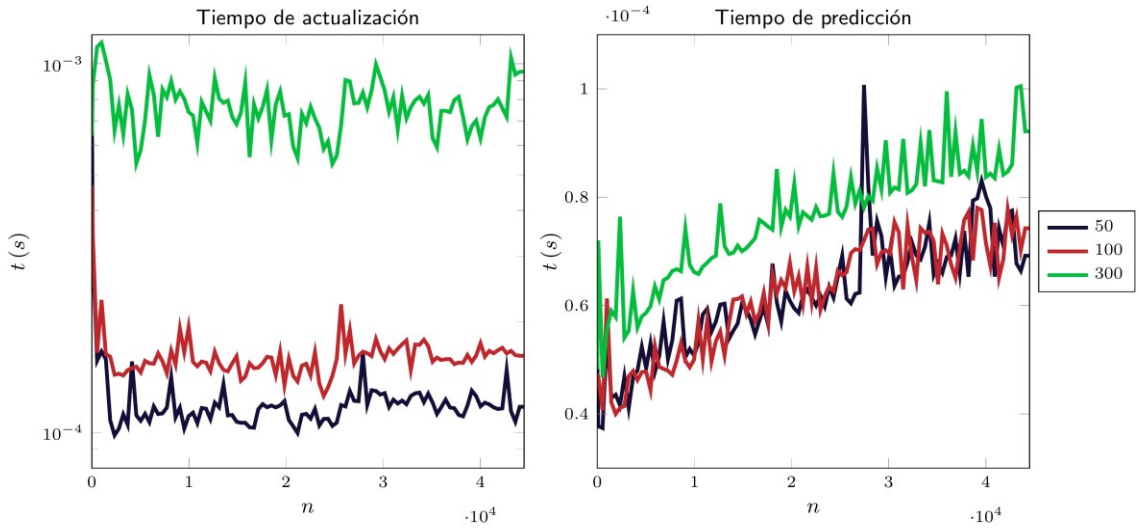

Figura 5.7. Tiempo de actualización y predicción del sistema según el límite de datos por proceso local.

Los promedios del tiempo son compilados en la **Tabla 5.6**. Se observa que, aunque el tiempo de predicción aumenta según el límite de datos, este el cambio entre los límites de 50 y 100 es aproximadamente de un 23 %. La misma comparación desarrollada en el tiempo de actualización es de aproximadamente 653 %. Por lo tanto, el principal efecto del límite de datos resulta en el tiempo de actualización.

Tabla 5.6. Tiempo promedio de predicción y actualización según el límite de datos.

Límite	t_{update} (µs)	t_{pred} (µs)	$t_{update} + t_{pred}$ (µs)
50	115.74	61.35	177.08
100	157.63	61.65	219.28
300	756.75	76.06	832.80

La **Tabla 5.7** resume el desempeño, en términos de error y NLL, de los siete grados de libertad. Se observa que el mejor desempeño de error es distribuido entre los límites de 50 y 100 datos. El NLL, por su parte, resulta mejor en todas las articulaciones cuando se limita a 50 datos.

Tabla 5.7. Resultados comparativos del desempeño del aprendizaje según el límite de datos para cada articulación J. El texto en negrita denota los mejores resultados.

Articulación	50		100		300	
	$nMSE$	NLL	$nMSE$	NLL	$nMSE$	NLL
J_1	**0.002**	**1.530**	0.004	1.608	0.005	1.770
J_2	**0.002**	**1.058**	0.002	1.129	0.003	1.276
J_3	**0.0008**	**0.333**	0.001	0.383	0.002	0.511
J_4	0.0005	**0.475**	**0.0005**	0.481	0.0006	0.507
J_5	**0.004**	**-1.183**	0.005	-1.149	0.006	-1.096
J_6	0.003	**-0.950**	**0.003**	-0.920	0.003	-0.838
J_7	0.002	**-0.865**	**0.001**	-0.834	0.002	-0.776

5.2. Comparaciones

La eficiencia del método, así como el desempeño del error, son comparados frente a diversos algoritmos desarrollados en años recientes. Se desarrolla una comparación similar a la expuesta en [22], sin embargo, se ajusta los hiperparámetros a los obtenidos mediante Rprop [23] y se compara únicamente los métodos basados en procesos gaussianos que son propuestos potencialmente para aprendizaje en tiempo real. El conjunto de datos utilizado es el mismo de la sección anterior. Los algoritmos utilizados en la comparación son:

- Procesos gaussianos locales divisibles (PGLD), propio de este trabajo y de [22], con los parámetros $m = 300$, límite de datos: 50 y utilizando $M_a(\text{máx mín})$ para determinar el hiperplano.
- El método de procesos gaussianos locales (LGP) expuesto en [6], con un máximo de 50 datos por cada subproceso.
- El método procesos gaussianos dispersos en-línea con entrada ruidosa (SONIG) de [24].
- El método de procesos gaussianos con espectro de dispersión incremental (I-SSGP) con el parámetro de 200 rasgos aleatorios.

La **Figura 5.8** muestra los resultados de error y NLL de las comparaciones desarrolladas para el primer grado de libertad. Se observa que el método PGLD tiene menor después de los $3\text{x}10^4$ datos aprendidos y significativamente mejor en los últimos datos aprendidos. Es evidente, a su vez, que, desde el inicio del aprendizaje, la bondad de ajuste es buena, pues el NLL es muy bajo comparado con los otros métodos; el método I-SSGP se aproxima conforme aprende, sin embargo, no obtiene un valor equivalente.

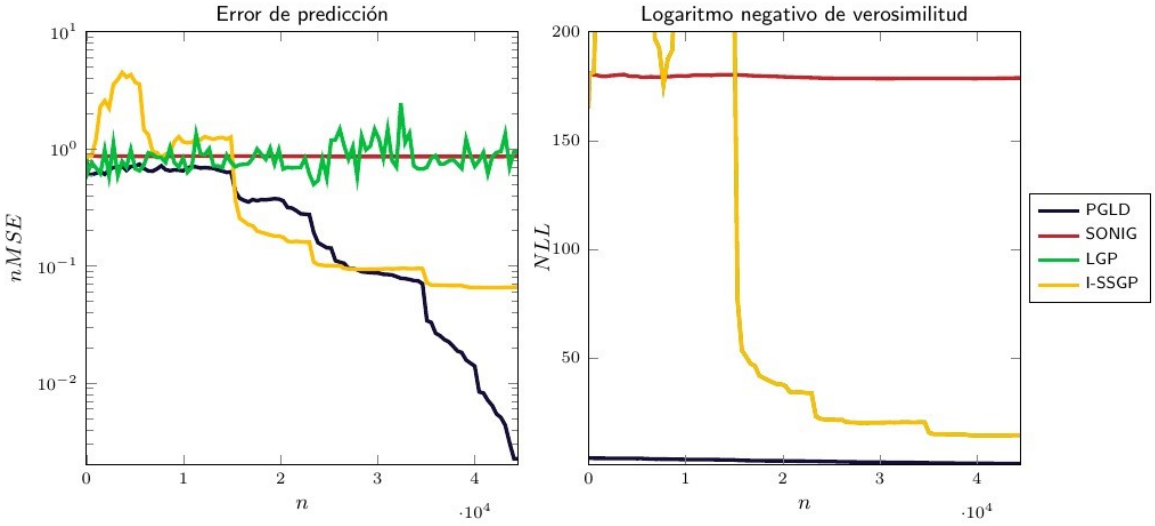

Figura 5.8. Comparación del desempeño de diversos métodos que emplean procesos gaussianos para regresión de J_1. El método LGP no describe una cálculo de NLL.

El desempeño en cuestión de tiempo está ilustrado en la **Figura 5.9**. Se observa que el método PGLD se actualiza al menos 5 veces más rápido que los demás métodos. Con respecto al tiempo de predicción, se observa que el método I-SSGP es aproximadamente 2.4 veces más rápido que PGLD, sin embargo, considerando ambos tiempos, como es evidente por la **Tabla 5.8**, el método PGLD es considerablemente más rápido que los demás.

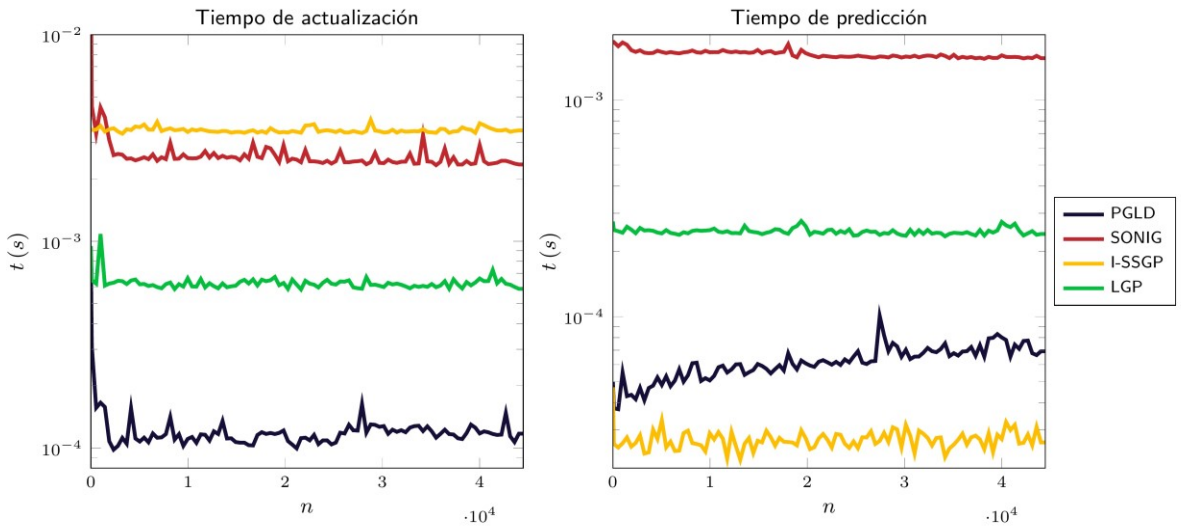

Figura 5.9. Comparación del tiempo de ejecución de diversos métodos que emplean procesos guassianos para regresión de J_1.

Los valores de errores y NLL para todas las articulaciones están compilados en la **Tabla 5.9**. Como se observó en la **Figura 5.8** el único método que tiene un descenso similar de error y NLL es I-SSGP, sin embargo, el desempeño de PGLD es superior en todas las articulaciones. Es evidente que tanto en el desempeño del error y bondad de ajuste, así como en el tiempo de computación, el método desarrollado en este proyecto presenta mejores resultados para el conjunto de datos relacionados al comportamiento de un manipulador SARCOS de siete grados de libertad.

Tabla 5.8. Tiempo promedio de predicción y actualización según el método de aprendizaje.

Límite	t_{update} (µs)	t_{pred} (µs)	$t_{update} + t_{pred}$ (µs)
PGLD	115.74	61.35	177.08
SONIG	2654.41	1584.83	4239.24
I-SSGP	3358.75	25.96	3384.71
LGP	628.35	483.03	1111.39

Tabla 5.9. Resultados comparativos del desempeño del aprendizaje según método de aprendizaje para cada articulación J. El texto en negrita denota los mejores resultados.

Articulación	PGLD		SONIG		I-SSGP		LGP
	$nMSE$	NLL	$nMSE$	NLL	$nMSE$	NLL	$nMSE$
J_1	**0.002**	**1.530**	0.858	178.698	0.0658	14.541	0.946
J_2	**0.002**	**1.058**	1.135	126.878	0.0640	8.023	0.434
J_3	**0.0008**	**0.333**	1.074	54.070	0.0551	3.649	0.628
J_4	**0.0005**	**0.475**	0.827	79.218	0.0252	3.301	0.391
J_5	**0.004**	**-1.183**	1.136	1.456	0.0750	0.954	0.674
J_6	**0.003**	**-0.950**	1.061	2.477	0.1144	1.087	0.839
J_7	**0.002**	**-0.865**	1.251	5.128	0.0312	1.024	0.654

5.3. Simulación

En esta sección se describe las evaluaciones del controlador de la **Figura 4.4** en el sistema aproximado. Las simulaciones se llevan a cabo en Simulink de MATLAB versión 2018a con un procesador Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz y 16 GB de memoria RAM.

En el desarrollo del control se emplea el bloque de aprendizaje descrito en la sección 4.3. Los parámetros seleccionados corresponden a los que son analizados en la sección 5.1 (límite de datos: 50, $m = 300$ y $s_d = M_a(\text{máx mín})$) Asimismo, se agrega un filtro de media aritmética de 30 valores a las predicciones. Esto con el fin de evitar señales con cambios abruptos y evitar vibraciones indeseadas. Finalmente, se agrega ruido aleatorio normalmente distribuido a la señal entrenamiento. El ruido está definido por:

$$n_{yTrain} \sim \mathcal{N}(0,1)$$

La simulación desarrollada no es determinística: el ruido es aleatorio y el aprendizaje tiene, a su vez, instrucciones fundamentadas en la aleatoriedad. Por lo tanto, se desarrolla 50 pruebas y se analiza la precisión de los resultados. Las ganancias utilizadas en estas evaluaciones son:

$$K_p = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} \qquad K_v = \begin{bmatrix} 20 & 0 \\ 0 & 20 \end{bmatrix}$$

El error RMS es calculado considerando todos los resultados de las pruebas. La **Figura 5.10** muestra el comportamiento del error, en radianes, a través del tiempo de ejecución de la evaluación. En los primeros segundos de la evaluación, es evidente que el error es máximo. Conforme avanza, se observa que el error disminuye hasta estabilizarse en un valor entre 10^{-2} y $10^{-3}\ rad$. Los mínimos locales resultan de los momentos en que las articulaciones cambian de dirección o cuando la articulación J_2 atraviesa el eje de $0°$.

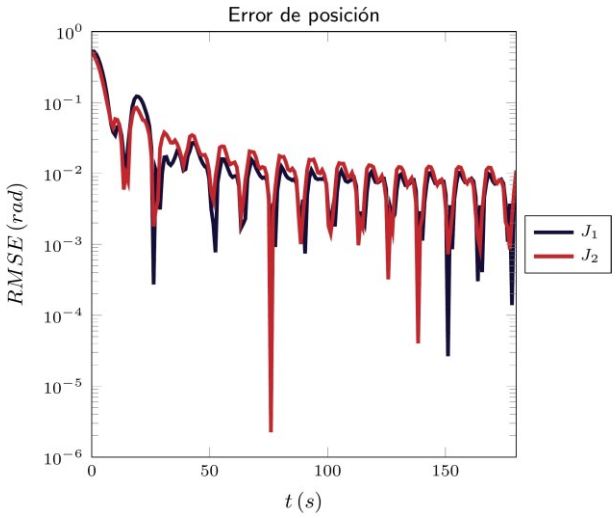

Figura 5.10. Error RMS de la posición del manipulador de dos grados de libertad para 50 evaluaciones.

La **Figura 5.11** muestra la señal de entrada al bloque de aprendizaje. Aunque la señal es alterada por el ruido, el sistema debe aprender a contrarrestar este error. Se observa que la predicción respectiva a estas entradas es menos ruidosa; esto resulta en beneficio para el control. En sistemas reales, el ruido es un componente fundamental por tomar en cuenta para la implementación de aprendizaje máquina. El hiperparámetro σ_n es, en los procesos gaussianos, el que define el nivel de ruido en el sistema.

La **Figura 5.12** muestra el desempeño del tiempo de actualización y de predicción del sistema para los bloques de ambos grados de libertad. Es evidente, a partir del rango 2σ que el inicio tiene la mayor variabilidad de tiempo. Sin embargo, en los datos subsiguientes el rango resulta en, aproximadamente, $100\ \mu s$ y $80\ \mu s$. Estos valores son suficientemente bajos para ser utilizados en aplicaciones de control con aprendizaje en

tiempo real aún con múlitples bloques de aprendizaje. Los resultados de esta sección difieren con los de la secciones 5.1 y 5.2 principalemente por ser desarrollados en una diferente plataforma y porque se lleva a cabo los cálculos de los bloques restantes del lazo de control.

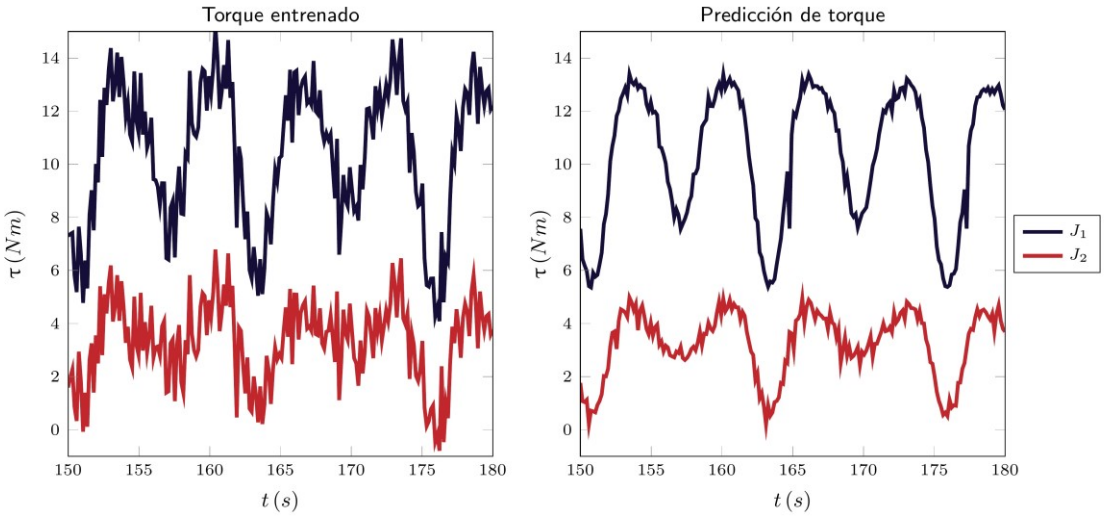

Figura 5.11. Resultado de predicción del torque a partir del entrenamiento.

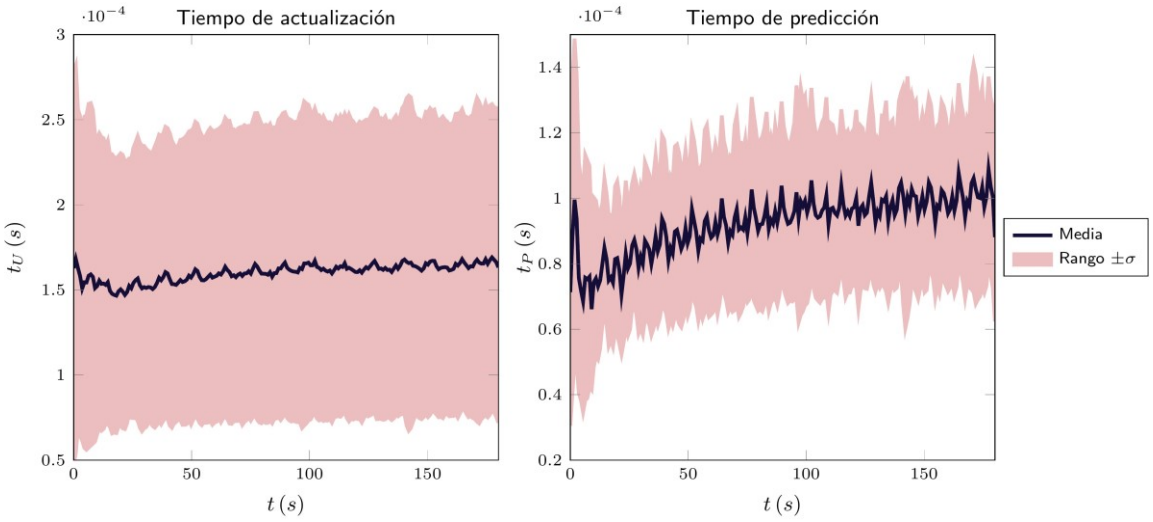

Figura 5.12. Desempeño, en tiempo, del aprendizaje máquina en el control.

5.4. Manipulador

El funcionamiento del controlador es comprobado mediante una implementación similar a la descrita en la sección 5.3 para el manipulador de la **Figura 5.13**. La implementación es llevada a cabo con Simulink de MATLAB versión 2017a con un procesador AMD Phenom(TM) II X6 1075T x 6 y 7.8 GB de memoria RAM. El manipulador se comunica directamente con el computador y el control es, por lo tanto, implementado en Simulink.

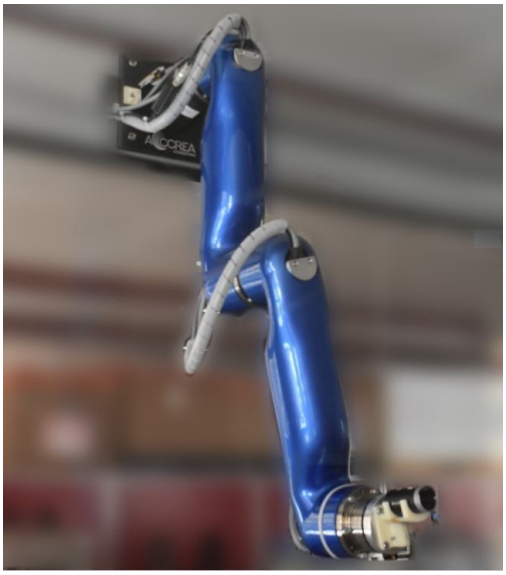

Figura 5.13. Captura del mainpulador SCARA utilizado en la evaluación.

Los hiperparámetros utilizados en esta sección son obtenidos mediante *Statistics and Machine Learning Toolbox* de MATLAB. La implementación coincide con la simulación en la incorporación de un filtro de media de 30 predicciones previas y en los valores de los parámetros. El funcionamiento del aprendizaje (predicción y actualización en conjunto) se desarrolla a 400 Hz.

Los controladores evaluados en esta sección son:

- **LG**: Control de torque calculado con ganancias bajas:

$$K_p = \begin{bmatrix} 4 & 0 \\ 0 & 3 \end{bmatrix} \qquad K_v = \begin{bmatrix} 20 & 0 \\ 0 & 20 \end{bmatrix}$$

- **MG**: Control de torque calculado con ganancias medias:

$$K_p = \begin{bmatrix} 40 & 0 \\ 0 & 40 \end{bmatrix} \qquad K_v = \begin{bmatrix} 20 & 0 \\ 0 & 20 \end{bmatrix}$$

- **HG**: Control de torque calculado con ganancias altas:

$$K_p = \begin{bmatrix} 250 & 0 \\ 0 & 250 \end{bmatrix} \qquad K_v = \begin{bmatrix} 20 & 0 \\ 0 & 20 \end{bmatrix}$$

- **PLDG-LG**: Control de torque calculado con ganancias bajas y aprendizaje máquina PLDG.

5.4.1. Control de aprendizaje

La **Figura 5.14** muestra el resultado de torque ante 5 repeticiones de PLDG-LG. Es evidente que los rangos mayores de desviación estándar se encuentran en los primeros $10\ s$. Esto es resultado de la condición no determinística del sistema y del control. Esto es evidentemente contrarrestado con el aprendizaje, pues se observa una disminución considerable de la desviación a partir de $10\ s$.

El error de posición de las evaluaciones está ilustrado en la **Figura 5.15**. La disminución del error sucede en los primeros $50\ s$. A partir de este momento el error oscila entre 10^{-3} y $10^{-2}\ rad$. El comportamiento, al igual que se describe en la sección 5.3 resulta de los cambios de dirección de los grados de libertad.

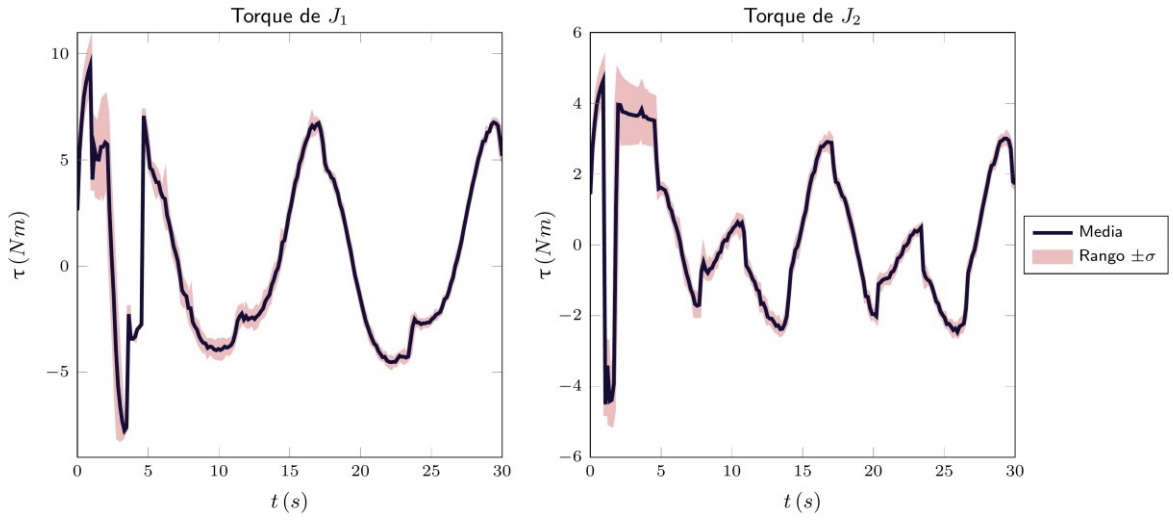

Figura 5.14. Torque de entrada para control PLDG-LG para 5 evaluaciones.

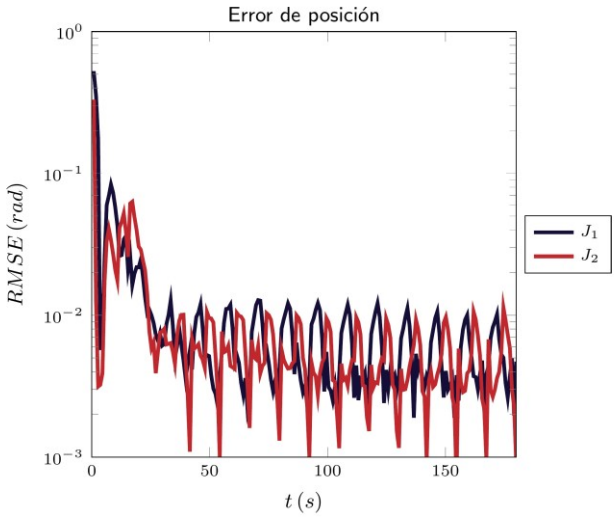

Figura 5.15. Error RMS de la posición del manipulador de dos grados de libertad para 5 evaluaciones.

5.4.2. Comparación de controles

El resultado del error de posición según el controlador está ilustrado en la **Figura 5.16**. Es evidente que un mayor valor de ganancia resulta en un menor error. Asimismo, los controles que no poseen aprendizaje convergen rápidamente a un valor. El control con aprendizaje, por su parte, aunque tiene las mismas ganancias que LG, el error es aproximadamente 16 (J_1) y 9 (J_2) veces menor. El error converge a un valor bajo y disminuye con el tiempo.

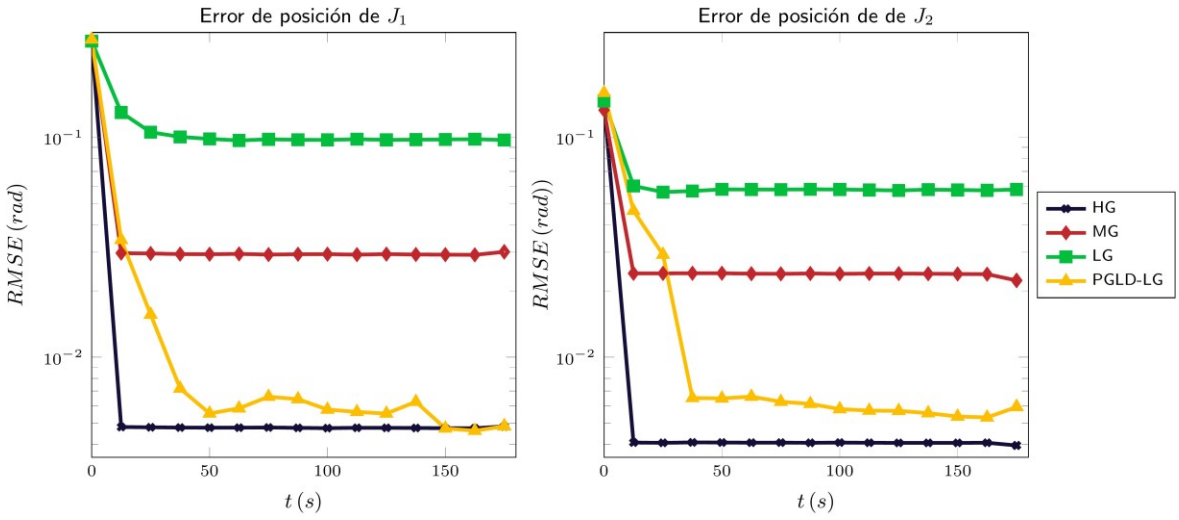

Figura 5.16. Comportamiento de RMSE cada 2500 mediciones según el controlador.

Las **Figura 5.17** y **Figura 5.18** muestran el comportamiento del torque según el controlador. Según lo observado sobre el error, se considera el control HG como el torque ideal en la comparación (aunque como es descrito en la sección 1.2 tener altas ganancias tiene otras desventajas). Para el control PGLD, la diferencia es prácticamente nula. El control LG presenta la diferencia más significativa, especialmente en los picos y los valles de la onda. El control MG tiene diferencia menor que LG, sin embargo, esta es notable, lo cual evidencia la sensibilidad que tiene el torque en el comportamiento del sistema. Es también evidente que el comportamiento durante los primeros segundos tiene un efecto significativo en el desempeño del control respecto al error de posición.

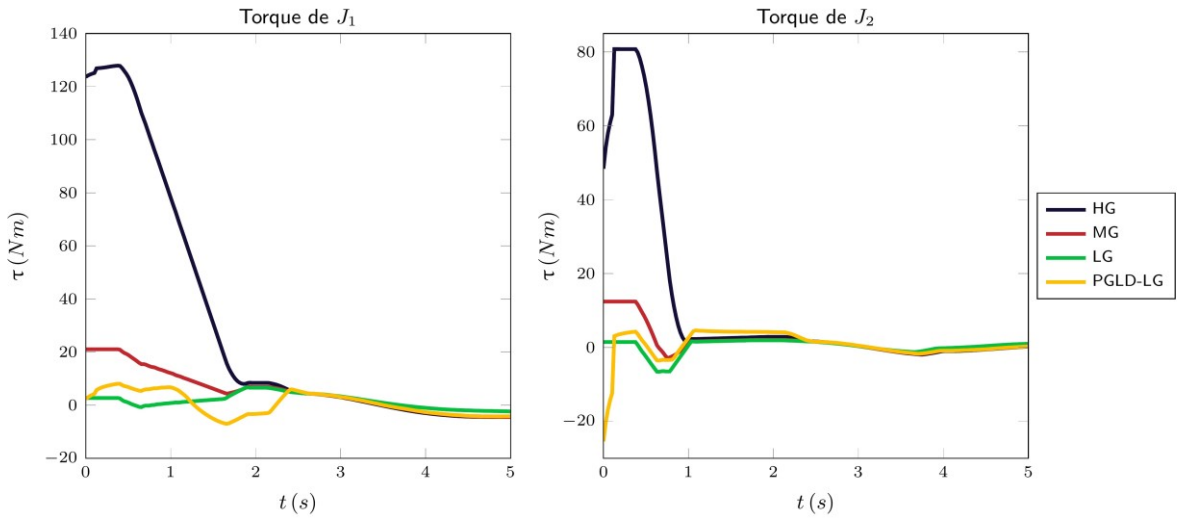

Figura 5.17. Comportamiento del torque según el controlador en los primeros segundos de ejecución

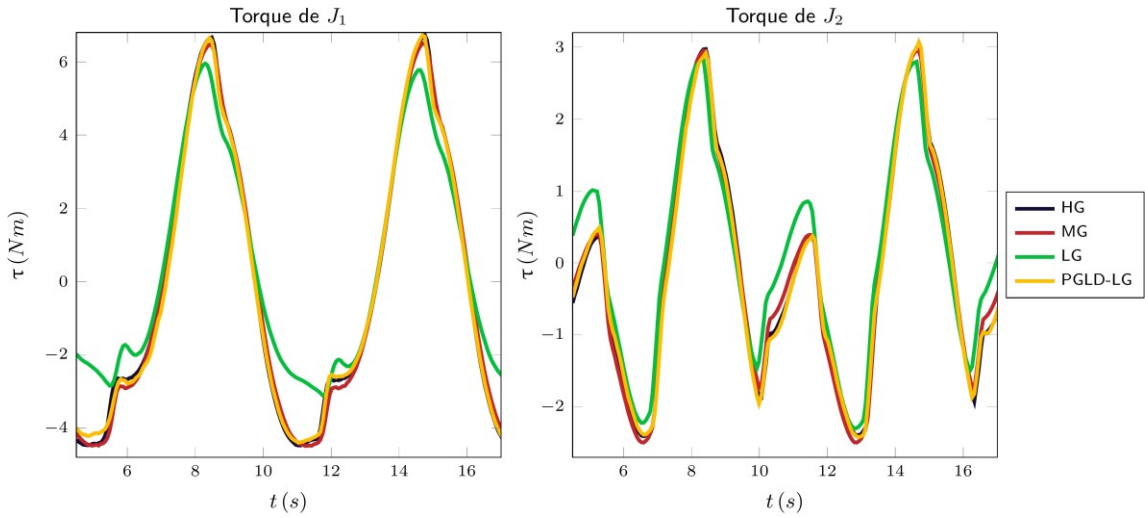

Figura 5.18. Comportamiento del torque según el controlador.

Capítulo 6. Conclusiones

Este proyecto describe un controlador novedoso que implementa aprendizaje máquina para contrarrestar los errores de modelado matemático y adaptarse a un ambiente cambiante. Se utiliza la linealización torque calculado y se complementa con procesos gaussianos locales divisibles.

El algoritmo de aprendizaje máquina implementado en este proyecto tiene un mejor desempeño que otras aproximaciones con procesos gaussiano. El tiempo de predicción y actualización es considerablemente menor que los demás métodos comparados. Asimismo, el valor del error, tanto al final de las evaluaciones como durante el aprendizaje, fue considerablemente mejor para el algoritmo desarrollado.

El controlador desarrollado fue evaluado a 400 Hz en el manipulador físico. Esto demuestra que el aprendizaje se desarrolla lo suficientemente rápido para representar una mejoría en el desempeño. Si el control no es lo suficientemente rápido, las señales se desfasan y el sistema no se lograría estabilizar, lo cual resultaría en un control inapropiado.

La distribución de los datos relacionados con el control es crítica para mantener el funcionamiento óptimo. Se pretende que el sistema pueda acceder a la mayor cantidad de datos para tener una mejor predicción sin que esto implique un costo computacional excesivo. Asimismo, utilizar hiperparámetros obtenidos a partir de mediciones acelera la disminución del error, pero implica una evaluación adicional previa a la ejecución.

Las evaluaciones, tanto en simulación como en el sistema físico de un manipulador de dos grados de libertad, revelan que el sistema logra aprender en tiempo real y funcionar con un desempeño similar al torque calculado con altos valores de ganancia sin las desventajas que este implica. El error converge rápidamente a un valor bajo, lo cual indica que el sistema podría adaptarse eficientemente a cambios en el ambiente.

6.1. Recomendaciones

Las evaluaciones demuestran que el controlador desarrollado con aprendizaje máquina converge a un valor de error con menor rapidez que los otros métodos. Por lo tanto, complementar el aprendizaje con el control de alta ganancia únicamente durante el inicio de las ejecuciones puede resultar en un control que incluya los beneficios de ambas implementaciones.

Un importante beneficio del controlador implementado es la facilidad de uso. La cantidad de parámetros es baja y fácilmente ajustable. Los hiperparámetros, por su parte, requieren de un conocimiento o evaluación previa del sistema para acelerar el aprendizaje. Una mejora potencial resulta de implementar la optimización, mediante el gradiente de la verosimilitud, de los hiperparámetros conforme se desarrolla el aprendizaje.

La robustez del controlador y el algoritmo de aprendizaje máquina deben ser evaluados en otros manipuladores robóticos y otros sistemas. Diferentes aplicaciones se pueden beneficiar en distinto grado del aprendizaje en tiempo real.

El aprendizaje del controlador puede verse beneficiado mediante aprendizaje condicionado. No todos los datos son significativos y representan una mejoría en el desempeño. Identificar datos prácticamente iguales, datos sesgados o datos de menor impacto puede resultar en una mejora para el aprendizaje máquina.

Implementar el control, así como el aprendizaje máquina, en otras plataformas representa un conjunto amplio de aplicaciones potenciales, mayor análisis o mejoras por parte de la comunidad relacionada al control con aprendizaje máquina.

Capítulo 7. Referencias

[1] J. Furman y R. Seamans, «AI and the Economy ©,» *Innovation Policy and the Economy,* vol. 19, nº 1, pp. 161-191, 2019.

[2] K. De Becker, T. DeStefano, C. Menon y J. Ran Suh, «Industrial robotics and the global organisation of production,» OECD Publishing, Paris, 2018.

[3] P. R. Daugherty y H. J. Wilson, Human + machine : reimagining work in the age of AI, Boston: Harvard Business Review Press, 2018.

[4] J. J. Craig, Introduction to Robotics, Upper Saddle River: Pearson Education, Inc, 2005.

[5] T. Beckers, J. Umlauft y S. Hirche, «Stable model-based control with Gaussian process regression for robot manipulators,» de *Proceedings of the 20th IFAC World Congress. 2017*, Múnich, 2017.

[6] D. Nguyen-Tuong, M. W. Seeger y J. Peters, «Model Learning with Local Gaussian Process Regression,» *Advanced Robotics,* vol. 23, pp. 2015-2034, 2009.

[7] S. Shi, R. J. Patton y Y. Liu, «Short-term Wave Forecasting using Gaussian Process for Optimal Control of Wave Energy Converters,» de *11th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS 2018*, Opatija, 2018.

[8] T. Kim, W. Kim, S. Choi y H. J. Kim, «Path Tracking for a Skid-steer Vehicle using Model Predictive Control with On-line Sparse Gaussian Process,» de *20th IFAC World Congress*, Seúl, 2017.

[9] Y. Ito, K. Fujimoto, Y. Tadokoro y T. Yoshimura, «On Stabilizing Control of Gaussian Processes for Unknown Nonlinear Systems,» de *20th IFAC World Congress*, 2017.

[10] M. Zarghami, S. H. Hosseinnia y M. Babazadeh, «Optimal Control of EGR System in Gasoline Engine Based on Gaussian Process,» de *20th IFAC World Congress*, 2017.

[11] J. Yang y C. Peng, «Evolving control of human-exoskeleton system using Gaussian,» *Biomedical Signal Processing and Control,* vol. 58, 2020.

[12] R. M. Murray, Z. Li y S. S. Sastry, A Mathematical Introduction to Robotic Manipulation, CRC Press, Inc., 1994.

[13] B. R. Markiewicz, «Analysis of the computed torque drive method and comparison with conventional position servo for a computer-controlled manipulator,» NASA, Pasadena, 1973.

[14] H.-F. Chen y L. Guo, Identification and stochastic adaptive control, Springer Science & Business Media, 2012.

[15] P. Ioannou y S. Baldi, «Robust Adaptive Control,» de *The Control Systems Handbook, Second Edition: Control System Advanced Methods*, CRC Press, 2010, pp. 35-1-35-22.

[16] K. J. Åström y K. E. Årzén, «Expert Control,» de *An Introduction to Intelligent and Autonomous Control*, Lund, Kluwer Academic Publishers, 1993, pp. 163-189.

[17] J. Farrel y W. Baker, «Learning Control Systems,» de *An Introduction to Intelligent and Autonomous Control*, Cambridge, Kluwer Academic Publishers, 1993, pp. 237-262.

[18] C. E. Rasmussen y C. K. I. Williams, Gaussian Processes for Machine Learning, Cambridge: MIT Press, 2006.

[19] G. Gregorčič y G. Lightbody, «Gaussian process approaches to nonlinear modelling for control,» de *Intelligent Control Systems Using Computational Intelligence Techniques*, The Institution of Engineering and Technology, 2005, pp. 177-218.

[20] Y. Shen, A. Y. Ng y M. Seeger, «Fast Gaussian Process Regression using KD-Trees,» de *Advances in Neural Information Processing Systems 18 (NIPS 2005)*, 2005.

[21] M. Schneider y W. Ertel, «Robot Learning by Demonstration with local Gaussian process regression,» de *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, 2010.

[22] A. Lederer, A. J. Ordóñez Conejo, K. Maier, W. Xiao, J. Umlauft y S. Hirche, «Real-Time Regression with Dividing Local Gaussian Processes,» arXiv 2006.09446, Munich, 2020.

[23] M. Blum y M. Riedmiller, «Optimization of Gaussian Process Hyperparameters using Rprop,» de *European Symposium on Artificial Neural Networks*, 2013, 2013.

[24] H. Bijl, T. B. Schön, J.-W. van Winderden y M. Verhaegen, «System Identification through Online Sparse Gaussian,» *IFAC Journal of Systems and Control,* vol. 2, pp. 1-11, 2017.

Apéndice A. Desarrollo dinámico

El desarrollo del modelo matemático del manipulador se obtiene a partir del Lagrangiano:

$$L(\theta, \dot{\theta}) = E_c(\theta, \dot{\theta}) - E_p(\theta)$$

Las posiciones cartesianas del centro de masa de cada articulación son:

$$x_1 = r_1 c_1$$

$$y_1 = r_1 s_1$$

$$x_2 = l_1 c_1 + r_2 c_{12}$$

$$y_2 = l_1 s_1 + r_2 s_{12}$$

Donde $c_i = \cos(\theta_i)$, $s_i = \sin(\theta_i)$ además, $c_{ij} = \cos(\theta_i + \theta_j)$ y $s_{ij} = \sin(\theta_i + \theta_j)$.

Las velocidades respectivas son:

$$\dot{x}_1 = -r_1 s_1$$

$$\dot{y}_1 = r_1 c_1$$

$$\dot{x}_2 = -l_1 s_1 \dot{\theta}_1 - r_2 s_{12} \dot{\theta}_1 - r_2 \, s_{12} \, \dot{\theta}_2$$

$$\dot{y}_2 = l_1 c_1 \dot{\theta}_1 + r_2 c_{12} \dot{\theta}_1 + r_2 \, c_{12} \, \dot{\theta}_2$$

Las energías cinéticas para cada articulación son:

$$E_{c,1} = \frac{1}{2} m_1 (\dot{x}_1^2 + \dot{y}_1^2) \dot{\theta}_1^2 + \frac{1}{2} I_1 \dot{\theta}_1^2$$

$$E_{c,2} = \frac{1}{2} m_1 (\dot{x}_2^2 + \dot{y}_2^2) \dot{\theta}_1^2 \dot{\theta}_1^2 + \frac{1}{2} I_2 (\dot{\theta}_1^2 + \dot{\theta}_2^2)$$

Done I_i representa el momento de inercia en el eje z de la articulación i.

Las energías potenciales son:

$$E_{p,1} = m_1 g r_1 s_1$$

$$E_{p,2} = m_2 g(l_1 \mathrm{s}_1 + r_2 s_{12})$$

Al aplicar:

$$\tau_i = \frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}_i} - \frac{\partial L}{\partial \theta_i}$$

Se obtiene la ecuación (3.3):

$$\begin{bmatrix}\tau_1 \\ \tau_2\end{bmatrix} = \begin{bmatrix} I_1 + I_2 + m_1 r_1^2 + m_2(l_1^2 + r_2^2) + 2m_2 l_1 r_2 c_2 & I_2 + m_2 r_2^2 + m_2 l_1 r_2 c_2 \\ I_2 + m_2 r_2^2 + m_2 l_1 r_2 c_2 & I_2 + m_2 r_2^2 \end{bmatrix} \begin{bmatrix}\ddot{\theta}_1 \\ \ddot{\theta}_2\end{bmatrix}$$
$$+ \begin{bmatrix} -m_2 l_1 r_2 s_2 \dot{\theta}_2 & -m_2 l_1 r_2 s_2 (\dot{\theta}_1 + \dot{\theta}_2) \\ m_2 l_1 r_2 s_2 \dot{\theta}_1 & 0 \end{bmatrix} \begin{bmatrix}\dot{\theta}_1 \\ \dot{\theta}_2\end{bmatrix}$$
$$+ \begin{bmatrix} m_1 g r_1 c_1 + m_2 g(l_1 c_1 + r_2 c_{12}) \\ m_2 g r_2 c_{12} \end{bmatrix}$$

Anexo A. Algoritmo Rprop

El algoritmo utilizado para la optimización de hiperparámetros es descrito en [23] para procesos gaussianos. El método consiste en obtener el vector $\boldsymbol{\vartheta}$ con hiperparámetros que minimicen el NLL. El valor de NLL se obtiene mediante:

$$f = -\log p\left(\boldsymbol{y}|\boldsymbol{X}, \sigma_n, \sigma_f, l\right) = -\frac{1}{2}\boldsymbol{y}\boldsymbol{K}_n^{-1}\boldsymbol{y} - \frac{1}{2}\log\det\boldsymbol{K}_n - \frac{n}{2}\log 2\pi$$

La derivada respecto al hiperparámetro j es:

$$\frac{\partial f}{\partial \vartheta_j} = \mathrm{tr}\left[(\boldsymbol{\alpha}\boldsymbol{\alpha}^T - \boldsymbol{K}_n^{-1})\frac{\partial \boldsymbol{K}_n}{\partial \vartheta_j}\right]$$

El método es iterativo. Se debe obtener el gradiente de NLL. Para cada hiperparámetro j el valor de la iteración $k+1$ es:

$$\vartheta_{j,k+1} = \vartheta_{j,k} - \mathrm{sign}\left(\frac{\partial f}{\partial \vartheta_{j,k}}\right)\Delta_{j,k}$$

Los valores iniciales son $\vartheta_{j,0} = 0$ para todo j. Además $\mathrm{sign}(x) = -1$ para $x < 0$, 1 para $x > 0$ y 0 para $x = 0$. Además $\Delta_{j,k+1}$, la magnitud del paso es:

$$\Delta_{j,k+1} = \begin{cases} \eta^+\Delta_{j,k} & \dfrac{\partial f}{\partial \vartheta_{j,k+1}}\dfrac{\partial f}{\partial \vartheta_{j,k}} > 0 \\ \eta^-\Delta_{j,k} & \dfrac{\partial f}{\partial \vartheta_{j,k+1}}\dfrac{\partial f}{\partial \vartheta_{j,k}} < 0 \\ \Delta_{j,k} & \dfrac{\partial f}{\partial \vartheta_{j,k+1}}\dfrac{\partial f}{\partial \vartheta_{j,k}} = 0 \end{cases}$$

Los valores η^+ y η^- son constantes que alteran la magnitud del paso. En este proyecto se utiliza 100 datos para la optimización y se hace 500 iteraciones. Además, las constantes utilizadas son: $\eta^+ = 1.2$, $\eta^- = 0.5$, $\Delta_{j,0} = 0.1$ para todo j y se limita los pasos a $\Delta_{j,k+1} \in [1\mathrm{x}10^{-6}, 50]$.