UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

TRABAJO FIN DE MASTER EN INGENIERÍA INDUSTRIAL

DISEÑO DE CONTROLADORES MEDIANTE TÉCNICAS DE APRENDIZAJE DE REGRESIONES LOCALES. APLICACIÓN AL CONTROL DE UN ROBOT PARALELO DE REHABILITACIÓN DE MIEMBRO INFERIOR

AUTOR: JORGE LÓPEZ GAMARRA

TUTOR: DR. ÁNGEL VALERA FERNÁNDEZ

COTUTOR: RAFAEL JOSÉ ESCARABAJAL SÁNCHEZ

Curso Académico: 2020-21

Agradecimientos

“Quiero aprovechar la ocasión para agradecer a mis padres y amigos, sin cuyo apoyo no habría llegado hasta aquí. Así mismo, al equipo del laboratorio por estar siempre disponibles para ayudar y animar, y en especial a Rafa sin en el que este trabajo nunca hubiera llegado a buen puerto”.

Resumen

En la actualidad las técnicas de aprendizaje autónomo suponen una de las grandes vanguardias de la estadística y los nuevos métodos de control y predicción. Su integración en los controladores de robots permite una mejora en el desempeño de todo tipo de tareas. A lo largo de este trabajo se presentarán las técnicas de aprendizaje basadas en regresiones locales, y porqué son las más idóneas para las aplicaciones robotizadas.

Posteriormente, se abordará la implementación de estas técnicas sobre un robot paralelo de cuatro grados de libertad, diseñado para realizar tareas de rehabilitación de miembros inferiores. Estos algoritmos lograrán aprender del comportamiento dinámico del robot, para realizar un control de posición capaz no solo de igualar el desempeño de los controladores tradicionales, sino de mejorarlos.

Esta tipología de regresiones será también utilizada para obtener un modelo del robot, que permita predecir su comportamiento, atajando el problema dinámico y cinemático directo de la robótica, sin la necesidad de resolver el complejo planteamiento matemático que involucra.

Adicionalmente, se implementarán técnicas de aprendizaje iterativo junto a las descritas anteriormente, que podrían mejorar el comportamiento para movimientos cíclicos típicos de tareas de rehabilitación, al aprovechar la información de los ciclos previos para reducir los errores en los siguientes.

Palabras Clave: Control de Robots, Inteligencia Artificial, Técnicas de Aprendizaje, Robot Paralelo, Robot de Rehabilitación, Regresiones Locales, Aprendizaje Iterativo, LWPR, ILC.

Resum

En l'actualitat les tècniques d'aprenentatge autònom suposen una de les grans avantguardes de l'estadística i els nous mètodes de control i predicció. La seva integració en els controladors de robots permet una millora en l'acompliment de tot tipus de tasques. Al llarg d'aquest treball es presentaran les tècniques d'aprenentatge basades en regressions locals, i per què són les més idònies per a les aplicacions robotitzades.

Posteriorment, s'abordarà la implementació d'aquestes tècniques sobre un robot paral · lel de quatre graus de llibertat, dissenyat per realitzar tasques de rehabilitació de membres inferiors. Aquests algoritmes aconseguiran aprendre del comportament dinàmic del robot, per a realitzar un control de posició capaç no només d'igualar l'acompliment dels controladors tradicionals, sinó de millorar-los.

Aquesta tipologia de regressions serà també utilitzada per a obtenir un model de el robot, que permeta predir el seu comportament, aturant el problema dinàmic i cinemàtic directe de la robòtica, sense la necessitat de resoldre el complex plantejament matemàtic que involucra.

Addicionalment, s'implementaran tècniques d'aprenentatge iteratiu junt a les descrites anteriorment, que podrien millorar el comportament per a moviments ciclics típics de tasques de rehabilitació, en aprofitar la informació dels cicles previs per reduir els errors en els següents.

Paraules Clau: Control de Robots, Intel · ligència Artificial, Tècniques d'aprenentatge, Robot Paral · lel, Robot de Rehabilitació, Regressions Locals, Aprenentatge Iteratiu, LWPR, ILC.

Abstract

Nowadays, machine learning techniques represent one of the great avant-garde of statistics and new methods of control and prediction. Their integration in robot controllers allows an improvement in the performance of all kinds of tasks. Throughout this paper we will present the learning techniques based on local regressions, and why they are the most suitable for robotic applications.

Subsequently, the implementation of these techniques on a parallel robot with four degrees of freedom, designed to perform lower limb rehabilitation tasks, will be addressed. These algorithms will be able to learn from the dynamic behaviour of the robot, to perform a position control capable not only of matching the performance of traditional controllers, but also of improving them.

This type of regression will also be used to obtain a model of the robot, which will allow predicting its behaviour, avoiding the direct dynamic and kinematic problem of robotics, without the need to solve the complex mathematical approach involved.

Additionally, iterative learning techniques will be implemented together with those described above, which could improve the behaviour for cyclic movements typical of rehabilitation tasks, by taking advantage of the information of the previous cycles to reduce the errors in the following ones.

Keywords: Robot Control, Artificial Intelligence, Learning Techniques, Parallel Robot, Rehabilitation Robot, Local Regressions, Iterative Learning, LWPR, ILC.

Índice de la Memoria

Índice del Presupuesto

Indíce de Figuras

Índice de tablas

Sección I. Memoria

1 Introducción

Si hay dos disciplinas de la ingeniería que cada vez suenan con más fuerza hasta en medios generalistas, estas son: la robótica y la inteligencia artificial. Estas ramas se han convertido en el núcleo central de toda la tecnología actual. Por separado, se pueden encontrar en aplicaciones industriales de todo tipo: motores de búsqueda, modelos de predicción complejos, automatismos domésticos y un largo etc. Pero aún más, se pueden llegar a ver trabajando en sincronía en aplicaciones como los vehículos autónomos, donde algoritmos de mapeado, visión 3D o predicción, se ponen al servicio de la unidad de control del robot móvil para realizar tareas complejas, como evitar obstáculos o trazar trayectorias óptimas.

El presente proyecto se enfoca en el control de robots paralelos para el seguimiento de trayectorias, esto es, lograr alcanzar las posiciones y velocidades deseadas aplicando unas ciertas acciones de control. Para lograr seguir eficazmente cualquier trayectoria, es necesario tener un complejo modelo dinámico y cinemático del sistema, capaz de conocer los momentos y fuerzas necesarios para lograr alcanzar un estado cinemático concreto. La dificultad para predecir con la suficiente precisión estos parámetros, lo vuelve un problema no trivial, en el que algoritmos clásicos de inteligencia artificial resultarían impracticables.

Por ello, se aplicarán las regresiones locales, técnicas de aprendizaje autónomo a menudo olvidadas, que permiten aproximarse a problemas no lineales de alta complejidad, mediante el desarrollo de modelos localizados espacialmente a lo largo de todo el espacio de entrada. Y es que sus características principales: velocidad de aprendizaje, flexibilidad, capacidad de generalización y de aprender de manera incremental, las vuelven idóneas para el control de robots de toda clase.

En concreto, el algoritmo utilizado es el LWPR (Locally Weighted Projection Regression), que combina técnicas estadísticas de vanguardia: la aproximación de problemas de manera local, y la proyección de regresiones como método para reducir la dimensionalidad del problema y evitar la conocida como 'maldición de las dimensiones'.

En este proyecto, se va a tratar de controlar un robot paralelo de 4 grados de libertad, destinado a la rehabilitación de miembros inferiores. El robot ha sido desarrollado por un equipo de trabajo de la Universidad Politécnica de Valencia, con el proyecto de investigación IMBiO3R, desarrollado bajo el Plan Nacional del Ministerio de Economía, Industria e Innovación del Gobierno de España.

La utilización de la robótica en medicina y rehabilitación, es todavía un campo por explorar, pero las posibilidades son muy amplias. Las aplicaciones robóticas podrían suponer un salto en la calidad de vida en pacientes de rehabilitación, al dejar de depender de la atención continua de personal especializado. Si los prototipos actuales siguen desarrollándose, podríamos ver en unos años la comercialización de equipos domésticos, que permitirían a los pacientes realizar sus rutinas desde la comodidad de su casa, sin requerir de desplazamientos muy costosos para personas en esta situación.

La implantación de novedosos controladores que integren las nuevas metodologías de la IA, podrían suponer importantes mejoras en la seguridad, precisión y control de los robots paralelos de rehabilitación, o de cualquier otra clase.

1.1 Objetivos

El objetivo principal de este trabajo es el estudio de las técnicas de regresiones locales, y el diseño de metodologías que permitan incorporarlas al sistema de control y seguimiento de trayectorias de robots paralelos de rehabilitación. Para lograr este fin, se proponen los siguientes objetivos específicos:

- Realizar una revisión bibliográfica del estado del arte, para analizar el desempeño de las técnicas de regresiones locales en el paradigma de la robótica, en concreto el algoritmo LWPR.
- Proponer y desarrollar soluciones que integren el algoritmo LWPR para el correcto control del robot paralelo de 4 grados de libertad, y para su uso como predictor de la posición.

- Desarrollar los esquemas de control diseñados, en el entorno de simulación, para poder validar sus propiedades con un modelo virtual del robot.
- Implementar en el hardware y software utilizado por el robot, las soluciones verificadas en simulación.
- Realizar ensayos experimentales con el robot real de los distintos algoritmos, para poder medir la eficacia de las soluciones propuestas.
- Implementación de algoritmos iterativas para aumentar la precisión del control mediante LWPR.
- Analizar los resultados obtenidos experimentalmente, para ajustar los diseños a fin de optimizar y mejorar el desempeño final.

1.2 Estructura de la memoria

En los próximos capítulos, se desarrollarán los siguientes apartados en los que se ha dividido la memoria del proyecto:

Planteamiento Teórico: en este capítulo, se desarrollará una descripción conceptual de las técnicas de aprendizaje por regresiones locales, justificando su uso en este proyecto, además, de realizar una descripción pormenorizada de los mecanismos matemáticos del algoritmo LWPR utilizado. También se introducirá el algoritmo iterativo ILC que se propone como optimización para tareas repetitivas. Finalmente, se realizará una breve introducción a los robots paralelos, y las características particulares que los diferencian del resto.

Modelo matemático del robot 3UPE-RPU: este capítulo incluye una descripción de los problemas cinemático y dinámico a resolver para poder obtener un modelo fiel del robot utilizado, desarrollando la formulación y soluciones adoptadas para este fin.

Materiales e Implementación: a lo largo de este capítulo se enumerarán los distintos elementos software y hardware que han sido requeridos para

elaborar el proyecto. Además, de exponerse la implementación específica de los distintos algoritmos.

Resultados: este capítulo recoge los resultados obtenidos en la fase experimental y de simulación, analizando el comportamiento en los diferentes escenarios. Adicionalmente, se describirán algunos de los problemas surgidos del análisis de los resultados preliminares, y las propuestas de mejora que se implementaron.

Conclusiones: en el capítulo final, se analizarán y discutirán los resultados del proyecto, la consecución de los objetivos, y las propuestas para mejorar en futuros trabajos.

2 Planteamiento Teórico

2.1 Introducción al aprendizaje mediante Regresiones Locales

Desde el desarrollo del primer algoritmo capaz de aprender por sí mismo, en 1952 por Arthur Samuel [1], que consistía en un programa capaz de jugar a las damas, y que mejoraba tras cada partida, las técnicas de aprendizaje automático (Machine Learning), y el concepto de Inteligencia Artificial (IA) se han popularizado enormemente, extendiéndose su uso por todo tipo de sectores. Aplicaciones basadas en distintas vertientes de estas técnicas se utilizan desde para obtener modelos climatológicos complejos, hasta para mejorar y anticipar el diagnostico de todo tipo de enfermedades, pasando por acelerar y simplificar la toma de decisiones de los grandes directivos. Y por supuesto, también se están empezando a aplicar para el control y aprendizaje de todo tipo de robots.

Sin embargo, el problema de control de un robot no es trivial, y presenta numerosas dificultades: el ruido de los sensores, la necesidad de una respuesta en tiempo real, el aprendizaje en línea o la limitación del tiempo de entrenamiento disponible son solo algunas de las causas que provocan que abordar esta problemática resulte extremadamente compleja [2].

En general, el problema del aprendizaje de un robot consiste en lograr que este realice una determinada tarea, sin que se haya programado explícitamente para ello. Se puede reducir el objetivo de control de un robot como aquel que resuelve el siguiente ejercicio: partiendo desde una posición inicial x_0, alcanzar un nuevo estado objetivo x, mediante la aplicación de la actuación necesaria u. El problema a resolver será, por tanto, el de lograr predecir adecuadamente el valor de la actuación (variable dependiente), en función del valor del estado inicial, y el estado deseado (variables independientes).

La problemática de resolver la relación que existe entre unas ciertas variables en el espacio de entrada, y otras en el espacio de salida de manera continua en el tiempo, se corresponde con una de las metodologías funda-

mentales del aprendizaje automático: la regresión lineal. Un tipo de aprendizaje supervisado (durante el entrenamiento el algoritmo recibe tanto los datos de entrada, como las salidas esperables), en el espacio continuo.

Si asumimos que la función que queremos caracterizar es de la forma: $f: X \to Y$, donde las variables de salida Y (dependientes), son función de las variables de entrada X (independientes).

Podemos tratar de aproximar esta función mediante regresiones paramétricas, donde asumimos que la función f, está bien representada mediante un modelo lineal parametrizado de la forma $f(x) = a^T x$. Siendo a, los parámetros del modelo lineal que una vez ajustados adecuadamente, permitirán realizar la predicción de la función real $f(x) \sim Y$ [3]. Actualmente existen dos grandes métodos de aproximación a las regresiones lineales: por un lado, los métodos globales, y por otro los locales. Tradicionalmente los más extendidos han sido los métodos globales, por su simple y buen funcionamiento para muchas aplicaciones. Pero a continuación discutiremos porque en aplicaciones de control de robots, pueden no resultar ser los más adecuados.

Una regresión lineal global, buscará obtener un modelo de representación lineal para todo el espacio de entradas/salidas ajustando el vector de parámetros β, de tal manera que minimice el error. Siendo el modelo lineal paramétrico (con x_i vector de entradas):

$$\widehat{y_i} = f(x_i, \beta) \tag{2.1}$$

El error a minimizar se modela mediante la llamada función de coste, que es el sumatorio ponderado de los errores de acuerdo a la ley L que se quiera establecer:

$$C = \sum_i L(\widehat{y_i}, y_i) \tag{2.2}$$

Por ejemplo, tomando el criterio típico de mínimos cuadrados, la función de coste sería:

$$C = \sum_i (f(x_i, \beta) - y_i)^2 \tag{2.3}$$

El problema a la hora de usar los métodos globales, es que puede suceder que, para ciertas zonas del espacio, el modelo lineal no se aproxime bien a la función real. La solución de este problema mediante las técnicas de regresión global, implica un aumento de la complejidad del modelo utilizado, confiando que este sea capaz de ajustarse al sistema real para la mayor cantidad de datos posible. Sin embargo, esto puede no resultar satisfactoriamente, o provocar un problema de *overfitting*. Es decir, puede provocar que el modelo sobreajuste demasiado a los datos de entrenamiento, pero que pierda capacidad de generalización en futuras predicciones.

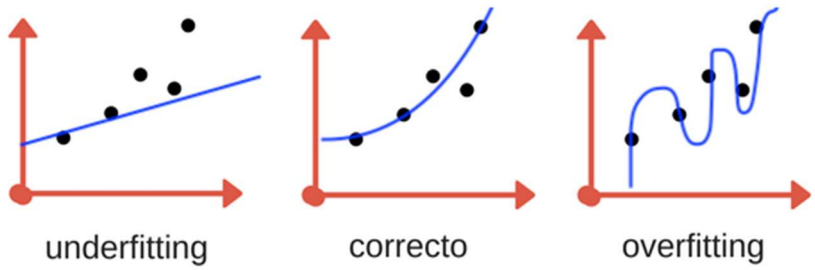


Figura 2.1: El problema de overfitting en ML [4]

Alternativamente, se puede optar por una vía diferente, y tratar de aproximar el problema global, a un subconjunto de problemas locales más sencillos en las distintas regiones de interés, logrando una buena aproximación en todo el espacio sin incurrir en un posible problema de *overfitting*. Este paradigma es el de las regresiones lineales locales. Cuando nos enfrentamos a un problema como el del control de un robot de varios grados de libertad, tratar de resolver el problema mediante técnicas globales no daría buen resultado, pues es necesario un método más flexible, capaz de funcionar bien para todo el conjunto de posibles posiciones del robot, no limitado a un pequeño espacio de trabajo.

Uno de los primeros trabajos en el campo de las regresiones locales, fue el desarrollo del algoritmo LWR (Locally Wighted Regression) [5], y antecesor del algoritmo LWPR. Este modelo, genera un número independiente de regresiones lineales simples a lo largo de toda la muestra de datos, pero con distintas funciones de ponderación, normalmente localizadas en distintas partes del espacio de entrada. De tal manera que ahora el modelo de regresión lineal local será de la forma:

$$\widehat{y_i} = f\big(x_i, \beta(q)\big) K\big(d(x_i, q)\big) \tag{2.4}$$

Donde ahora $f\big(x_i, \beta(q)\big)$ son los modelos locales para cada q espacio local, y $K\big(d(x_i, q)\big)$ es el peso ponderado debido a la distancia entre el vector de datos x_i y cada regresión local q.

Y por tanto ahora la función de coste a minimizar será:

$$C = \sum_i [(f(x_i, \beta) - y_i)^2 K\big(d_i(x_i, q)\big)] \tag{2.5}$$

El resultado de este diseño local, es que ahora el modelo de predicción está constituido como una combinación ponderada de los distintos modelos locales más sencillos. Esta ponderación se puede realizar por diversas metodologías, pero en general tendrá un peso inversamente proporcional a la distancia entre el espacio de entrada de cada dato, y el espacio de cada regresión local.

La aplicación de este nuevo paradigma de regresión local en control, resulto muy satisfactoria, reportando el uso del algoritmo LWR numerosos beneficios [6]:

- Linealiza los modelos de manera autónoma: como hiciese un ingeniero de control, las regresiones locales están tomando modelos no lineales, y linealizándolos en diferentes puntos de funcionamiento para cada espacio de entrada.
- Facilidad para añadir nuevos datos de entrada: permitiendo aumentar el conjunto de datos utilizado fácilmente.
- Capacidad de aprendizaje en pocas pasadas: por lo que no requerirá un entrenamiento continuado sobre el mismo conjunto de datos una y otra vez.
- No sufre del problema de caer en mínimos locales, gracias al uso de regresiones locales ponderadas.
- Evita interferencias de los nuevos datos: este tipo de regresiones no presenta una mayor afinidad por los nuevos datos de entrada. A

diferencia de otras regresiones globales, que tienden a 'olvidar' las experiencias pasadas, y comportarse mejor en las nuevas tareas en las que son entrenadas.

Estas características resultan muy ventajosas para aplicar el algoritmo sobre tareas relacionadas con el control de robots. Sin embargo, el LWR no está exento de algunos problemas:

- En primer lugar, el coste de computación y el espacio en memoria se incrementa con el tamaño de la muestra de datos. Esto se produce porque cuando añadimos datos de entrada en nuevas regiones, se generan nuevas regresiones locales, aumentando el tamaño del modelo, y en consecuencia, el tiempo de computación.
- El algoritmo puede sufrir de la llamada 'maldición de la dimensionalidad', producida por el aumento exponencial de los recursos necesarios conforme aumenta el tamaño y las dimensiones del modelo.
- Además, un buen comportamiento del algoritmo requiere de una buena representación de los datos a procesar. Una buena elección de las entradas, y su correcto escalado, pueden tener un gran impacto en el comportamiento y adecuación de los modelos resultantes.

A partir del algoritmo LWR, y con el fin de solucionar algunos de sus problemas y extender sus aplicaciones, se siguieron desarrollando nuevas técnicas basadas en el nuevo paradigma de las regresiones locales ponderadas. Entre ellas cabe destacar el modelo RFWR (Receptive Field Weighted Regression) [7], una variante del LWR adaptada para trabajar de manera incremental, lo que lo habilitará para el aprendizaje en línea, que es vital para trabajar como controlador de sistemas robotizados. En este caso el propio algoritmo es capaz de agregar los nuevos modelos lineales, y sus funciones de ponderación (llamadas 'receptive fields') automáticamente, cuando aparece algún punto de entrada que no está cubierto por las regresiones almacenadas previamente.

Finalmente, y a partir de este modelo incremental, se desarrolló el algoritmo de regresión local ponderada de proyección (Locally Weighted Projection Regression o LWPR) [8].

2.2 Locally Weighted Projection Regression

Este nuevo algoritmo logra la aproximación de funciones no lineales en espacios de alta dimensión (con dimensiones de entrada que pueden ser redundantes e irrelevantes). Su núcleo se basa en los modelos lineales locales desarrollados en los anteriores algoritmos, proyectados como regresiones univariantes en determinadas direcciones del espacio de entrada.

La aproximación de funciones no lineales con unos datos de entrada de altas dimensiones, detectando las dimensiones de entrada irrelevantes, y manteniendo la complejidad computacional baja, se logra gracias a la proyección de la regresión. La proyección permite hacer frente a estos problemas, descomponiendo las regresiones multivariables en una superposición de regresiones de una sola variable a lo largo de proyecciones particulares del espacio de entrada. La mayor dificultad de las regresiones proyectadas, reside en la manera de seleccionar eficientemente las proyecciones. Combinando la técnica de las proyecciones, con el desarrollo de las regresiones locales, podemos encontrar proyecciones diferentes localmente evitando tener que encontrar las mejores proyecciones a nivel global.

En combinación tentemos un algoritmo que combina diversas técnicas estadísticas con el objetivo de aproximar funciones no lineales cuyos datos se reciben de manera incremental. La aplicación de esta técnica ha demostrado ser muy eficaz en un amplio rango de aplicaciones de control y robótica.

A continuación, se va a desarrollar un breve resumen y descripción del funcionamiento y principios matemáticos de este algoritmo. La hipótesis de partida es que los datos son muestreados a partir de una función no lineal y, dependiente del vector de entradas x, junto con un cierto ruido de distribución típica normal con media cero:

$$y = f(x) + \epsilon \tag{2.6}$$

El objetivo del LPWR es definir un conjunto de modelos lineales que se aproximen localmente a la función no lineal, en una cierta región del espacio de entrada x. Es decir, en cualquier valor de entrada x, que este suficientemente cerca del centro c_k del $k^{ésimo}$ modelo local lineal, la función se aproximará por:

$$y_k = \beta_k^T x + \epsilon \tag{2.7}$$

Siendo β_k el vector de parámetros que define el hiperplano correspondiente a y_k. Ahora bien, el grado en el que la entrada x pertenece a este modelo local, está determinado por la función de ponderación:

$$w_k = \exp(-\frac{1}{2}(x - c_k)^T D_k (x - c_k)) \tag{2.8}$$

Donde D_k corresponde con la distancia que determina el tamaño y la forma de la región de validez del modelo lineal. Estas regiones también son conocidas como 'campos receptivos' (Receptive Fields o RF).

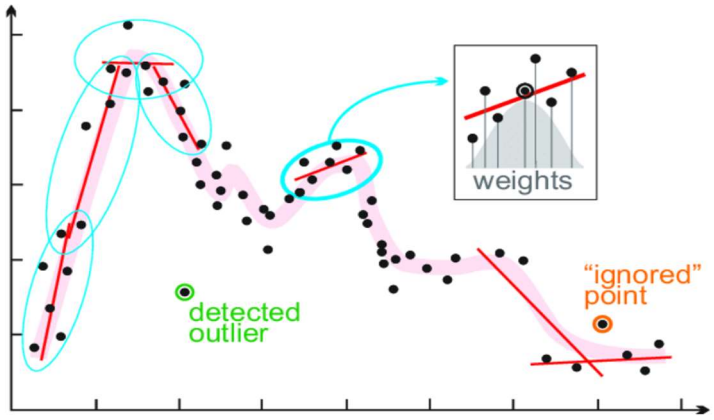


Figura 2.2: Representación de un modelo compuesto de regresiones locales ponderadas [9]

Asumiendo que existen K modelos locales combinados para hacer la predicción. Si para un determinado vector de entradas x, cada modelo lineal calcula una predicción y_k. La salida total del algoritmo será una combinación ponderada de todos los modelos lineales de la forma:

$$\hat{y} = \frac{\sum_{k=1}^{K} w_k y_k}{\sum_{k=1}^{K} w_k} \tag{2.9}$$

Es decir cada muestra x activa todos los K modelos locales, y su efecto en la función de salida $\hat{y}$, depende del valor del peso asociado w_k (función de activación). Aquellos modelos más cercanos al espacio de entrada x, tendrán un mayor peso en su comportamiento, mientras que los modelos más alejados no influirán casi en la función de salida pues su w_k será despreciable.

Como se ha comentado previamente, el algoritmo LWPR nace de la combinación de los avances en las técnicas de regresiones locales, con la reducción de dimensiones ofrecida por la proyección de regresiones. La aplicación de este último elemento, es lo que supuso un gran avance sobre los algoritmos anteriores (RFWR y LWR). Pues estos sufrían fuertemente de la maldición de las dimensiones, y el tiempo de cálculo se volvía absurdo para modelos con más de 10 dimensiones [8].

Sin embargo, aplicando el marco de trabajo de las técnicas de proyección (PLS/CPR), se logra reducir el tiempo de computación de cada región lineal aplicando una secuencia de regresiones monovariables, a lo largo de ciertas direcciones de proyección u_r en el espacio de entrada.

Dado un punto de entrada (x, y):

- Actualizar las medias de entrada y salida:

$$x_0^{n+1} = \frac{\lambda W^n x_0^n + wx}{W^{n+1}}$$
$$\beta_0^{n+1} = \frac{\lambda W^n \beta_0^n + wy}{W^{n+1}}$$

donde $W^{n+1} = \lambda W^n + w$, $x_0^0 = 0$, $u_i^0 = 0$, $\beta_0^0 = 0$ y $W^0 = 0$

- Actualizar el modelo local:
  1. Inicializar $z = x$, $res_1 = y - \beta_0^{n+1}$
  2. Para $i = 1$ hasta r:

$$u_i^{n+1} = \lambda u_i^n + wz\ res_i$$
$$s = z^T + u_i^{n+1}$$
$$SS_i^{n+1} = \lambda SS_i^n + ws^2$$
$$SR_i^{n+1} = \lambda SR_i^n + ws\ res_i$$
$$SZ_i^{n+1} = \lambda SZ_i^n + wzs$$
$$\beta_i^{n+1} = SR_i^{n+1}/SS_i^{n+1}$$
$$P_i^{n+1} = SZ_i^{n+1}/SS_i^{n+1}$$
$$z = z - sP_i^{n+1}$$

$$res_{i+1} = res_i - s\beta_i^{n+1}$$
$$MSE_i^{n+1} = \lambda MSE_i^n + w\ res_{i+1}^2$$

- Predicción de nuevos datos:
  1. Inicializar $y = \beta_0$, $z = x - x_0$
  2. Para $i = 1$ hasta k:

$$s = u_i^T z$$
$$y = y + \beta_i s$$
$$z = z - sP_i^n$$

Figura 2.3: Pesudocódigo del PLS incremental [8]

El algoritmo LWPR utiliza una versión incremental y con pesos ponderados localmente para determinar los parámetros de los diferentes modelos lineales. En la figura 2.3 se resume el pseudocódigo de su implementación, el desarrollo completo puede consultarse en [8].

Se ha descrito la manera en la que el algoritmo es capaz de encontrar las dimensiones de proyección más adecuadas, así como la generación de regresiones locales en cada área. Pero la validez y adecuación de estos modelos locales, depende en gran medida del grado en que se escoge correctamente el tamaño y forma de los RF, lo cual como ya se ha comentado está determinado por la distancia D_k.

Se puede optimizar el valor de D individualmente para cada RF usando el gradiente descendente. La regla de actualización se puede escribir como sigue:

$$D = M^T M \tag{2.10}$$

Siendo M la triangular superior que se actualiza de la forma:

$$M^{n+1} = M^n - \alpha \frac{\delta J}{\delta M} \tag{2.11}$$

Cuya función de coste a minimizar será:

$$J = \frac{1}{W} \sum_{i=1}^{M} \sum_{k=1}^{r} \frac{w_i res_{k+1,i}^2}{\left(1 - w_i \frac{s_{k,i}^2}{s_k^T w_{s_k}}\right)^2} + \gamma \sum_{i,j=1}^{N} D_{ij}^2 \tag{2.12}$$

Siendo γ el termino de regularización.

Finalmente, se resume la rutina de funcionamiento del algoritmo LWPR mediante el siguiente flujo de trabajo:

- Inicialización del LWPR sin ningún RF
- Para cada nuevo dato de entrenamiento (x,y):
    - Para cada k=1 hasta K (cada RF):
        - Calcular la activación w_k
        - Actualizar los parámetros de las regresiones de acuerdo al pseudocógido del PLS incremental y la distancia D_k
    - Si no se ha activado ningún modelo con un valor de w_k mayor que el umbral w_{gen}:
        - Se crea un nuevo RF en $c = x$, $r = 2$ y $D = D_{def}$
- Fin del entrenamiento

Siendo w_{gen} el umbral de activación que determina que hace falta un nuevo RF, r el número inicial de proyecciones que se fija en 2, y D_{def} el valor inicial de la distancia D, usualmente inicializado como una matriz diagonal.

Una activación menor al umbral w_{gen}, indica que se está ante un nuevo espacio de trabajo, pues las funciones de activación son muy bajas, indicando que la distancia entre los espacios ya existentes y el espacio de la nueva muestra es elevada.

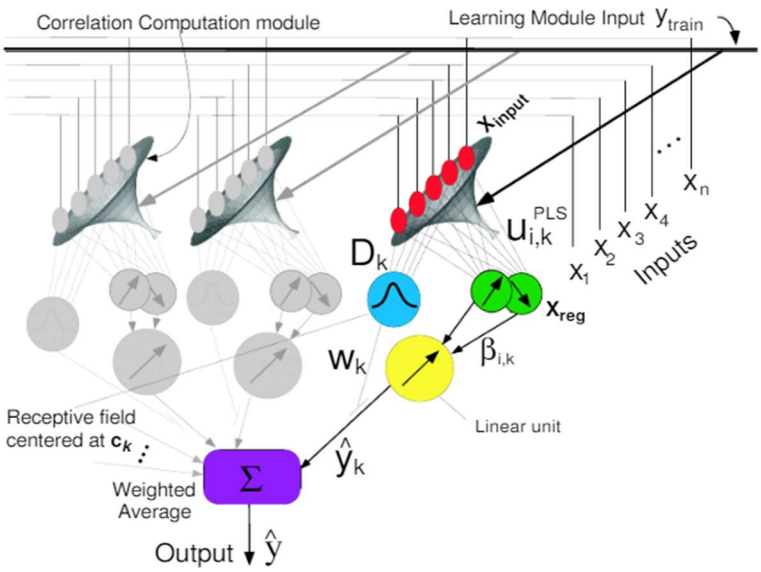


Figura 2.4: Esquema de funcionamiento del procesamiento LWPR [10]

En conclusión, todas estas características bridan al algoritmo LWPR de una gran capacidad de adaptabilidad, actualizando por sí mismo una gran cantidad de parámetros para adaptarse a los nuevos datos de entrada. En el caso de querer representar un espacio de salida multidimensional, se generarán múltiples LWPR mono-salidas que modelan con los mismos datos de entrada x, diferentes funciones objetivo y.

2.3 Algoritmos iterativos: ILC

Cuando se trata de tareas de rehabilitación es muy habitual que los movimientos de recuperación sean cíclicos, repitiendo una y otra vez el movimiento de la articulación o zona dañada. En este supuesto, cada nueva iteración sobre un mismo movimiento genera mucha información relevante sobre el comportamiento del proceso, que no es utilizada por los controladores no lineales clásicos.

El objetivo de los algoritmos iterativos es mejorar el comportamiento del controlador a lo largo de las iteraciones, aprovechando la información del error que sabemos que se va a producir en la siguiente ejecución. Esto permitiría la mejora del comportamiento de sistemas que ejecutan una tarea específica repetitivamente, como sucede en muchas aplicaciones de la robótica, y en este caso en los movimientos de rehabilitación.

Basado en este concepto se empezó a desarrollar y aplicar uno de los algoritmos iterativos más extendidos el Iterative Learning Control (ILC) [11], que ha sido aplicado con éxito a todo tipo de tareas repetitivas en la industria manufacturera, la robótica y la industria de procesos, mejorando el funcionamiento de máquinas de control numérico, inyectores, motores de inducción, trasportadores de cadenas, etc. [12]

La implementación del ILC permite generar un control por prealimentación realice el seguimiento de una trayectoria específica, o corrija el efecto de una perturbación. Mientras que los controles por realimentación reaccionan a las entradas y perturbaciones, implicando siempre un retraso en el seguimiento. El control ILC por prealimentación elimina ese retraso, aunque solo para aquellas señales conocidas o medibles. Por tanto, presentará ciertas limitaciones, dado que corrige los errores basado en la repetición, aquellas perturbaciones de comportamiento errático y el ruido aleatorio no logran ser compensados.

A continuación, se va a introducir el funcionamiento del algoritmo y su esquema de control. Debe tenerse en cuenta que es un tipo control que se implementa en el dominio del tiempo discreto, pues el ILC requiere específicamente almacenar los datos muestreados en iteraciones pasadas. Considerando el sistema SISO invariante en el tiempo:

$$y_j(k) = P(q)\ u_j(k) + d(k) \tag{2.13}$$

Siendo k el número de muestra, y j el índice de la iteración, $P(q)$ la planta del sistema, $u_j(k)$ la señal de control, $d(k)$ una señal exógena y $y_j(k)$ la salida del sistema. Podemos definir la señal de error en cada instante como:

$$e_j(k) = y_d(k) - y_j(k) \tag{2.14}$$

Siendo $y_d(k)$ la referencia del sistema. Se define la señal de control del algoritmo ILC como:

$$u_{j+1}(k) = Q(q)[u_j(k) + L(q)e_j(k+1)] \tag{2.15}$$

Donde $Q(q)$ es un filtro, y $L(q)$ la función de aprendizaje. Es decir que la señal de control se calcula como el valor de la acción de control ILC en ese instante en la iteración anterior junto a la ley de control del error en el instante siguiente (registrada en la iteración anterior).

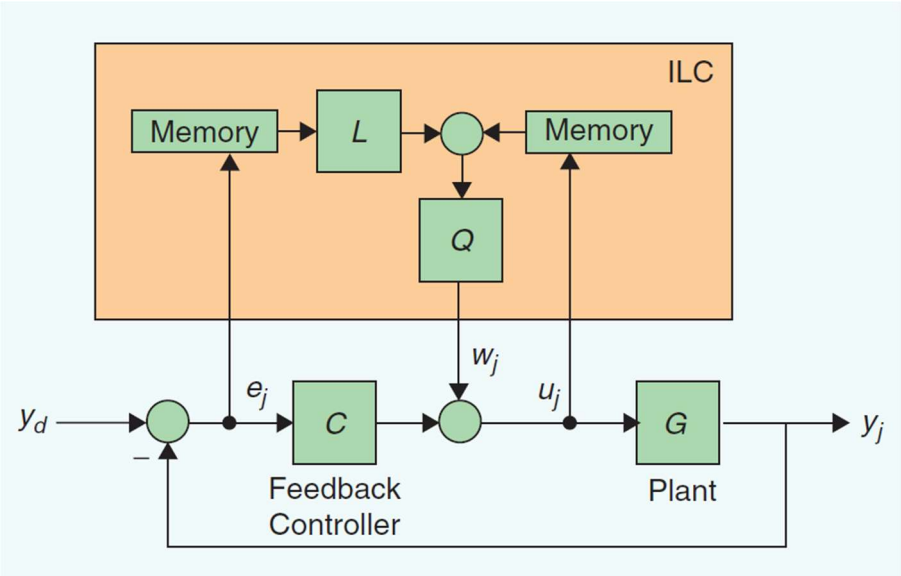


Figura 2.5: Esquema de funcionamiento del algoritmo ILC [12]

El uso del controlador ILC no sustituye el control por realimentación tradicional, sino que lo complementa para mejorar su desempeño.

Existen varias formas de implementar este algoritmo y de obtener el valor de las matrices $Q(q)$ y $L(q)$: mediante inversión de la planta, optimización cuadrática o diseños $\mathcal{H}_\infty$. Pero en general uno de los métodos más simples, eficaces y extendido por ello, es el diseño ILC tipo PD y será el que se utilice en la implementación de este trabajo.

Como su nombre indica, el diseño tipo PD aproxima la señal de aprendizaje mediante una señal proporcional y otra derivativa del error. El ajuste de los parámetros no requiere de implementar un modelo preciso de la planta, basta con ajustar experimentalmente la ganancia proporcional y derivativa del algoritmo para lograr un buen comportamiento. La función de aprendizaje será al siguiente:

$$u_{j+1}\ (k) = u_j(k) + K_p\ e_j(k+1) + \ K_d\ [e_j(k+1) - e_j(k)] \qquad (2.16)$$

Siendo K_p y K_d las ganancias proporcionales y derivativas a ajustar respectivamente. Adicionalmente y para evitar problemas de inestabilidad debido a la realimentación continua de ruido aleatorio, se introduce un término de tasa de aprendizaje (learning rate) α:

$$\alpha = 2^{\frac{A}{j+A-1}} \tag{2.17}$$

Siendo A un parámetro a ajustar que modifica la velocidad de descenso de la tasa de aprendizaje, quedando la ley de control ILC:

$$u_{j+1}\ (k) = u_j(k) + \alpha\ K_p\ e_j(k+1) + \alpha\ K_d\ [e_j(k+1) - e_j(k)] \tag{2.18}$$

La ley de control 2.18 será la que se implemente en la solución real aplicada al robot, y cuyos detalles y resultados se comentarán más adelante.

2.4 El robot paralelo

Desde la aparición de la palabra robótica en 1942 [13], como algo propio de la ciencia ficción, de mano de la obra de Isaac Asimov 'Circulo Vicioso' [14], la fantasía se ha vuelto realidad, habiendo en la actualidad 2.7 millones de robots trabajando a lo largo del mundo, de acuerdo a la federación Internacional de la Robótica (IFR) [15].

Su rápida expansión por los distintos sectores industriales, y su previsible impacto en cada vez más nuevos segmentos fuera de la industria, son sin duda un síntoma de su éxito y eficacia. Y es que los robots permiten realizar tareas de todo tipo gracias a su gran flexibilidad, con una precisión y repetitividad muy superior a la de cualquier humano. Además, de ser capaces de trabajar sin descanso y con cargas de peso muy por encima del umbral de cualquier persona.

En general, se puede elaborar la clasificación de los robots atendiendo a diversos criterios, tales como el número de grados de libertad que poseen, su tamaño o su capacidad de carga. Pero sin duda, una de las clasificaciones más habituales, es la clasificación en base a su estructura cinemática. De

esta manera se distinguen dos tipos principalmente: robots con una configuración cinemática abierta, llamados habitualmente robots serie, o robots con una cadena cinemática cerrada, también conocidos como robots paralelos.

Los robots serie están constituidos por una base fija, seguida de una serie de eslabones consecutivos, que terminan en un extremo libre y móvil. Estos son los robots más habituales en el sector industrial. Suelen poseer una estructura mecánica antropomórfica, constituida por una serie de eslabones rígidos unidos por juntas (normalmente de revolución), que forman la cadena cadera-hombro-codo-muñeca.

Las mayores ventajas de los robots series son: su gran área de trabajo, respecto a su propio volumen, y el poco volumen de suelo que ocupan. Por el contrario, este tipo de cadena cinemática presenta algunos inconvenientes: principalmente su falta de rigidez, propia de su estructura cinemática abierta, y la acumulación de errores entre eslabones consecutivamente, provocando que el mayor grado de error se dé justo en el extremo, cuya posición queremos controlar [16].

Figura 2.6: Izda: configuración serie de 6 juntas de ABB [17], Dcha: configuración SCARA de OMROM [18]

Habitualmente, los robots serie suele presentar una estructura de seis juntas, que les permiten manipular objetos en cualquier posición y orientación que deseen. Aunque, son también habituales configuraciones de cuatro grados de libertad, para aplicaciones más simples de 'pick&place' (las más habituales para robots serie), en las que no hace falta reorientar los objetos manipulados en las tres direcciones del espacio. Este último tipo de configuración suele recibir el sobrenombre de robots SCARA.

Por su parte, en los robots paralelos todas las cadenas cinemáticas confluyen en un mismo elemento. Suelen consistir en una base fija, conectada por la plataforma móvil, por un cierto número de cadenas cinemáticas independientes (extremidades o patas). Estas extremidades suelen consistir habitualmente en juntas prismáticas, unidas con la plataforma mediante juntas esféricas pasivas. Lo que confiere a los robots paralelos de una estructura mucho más rígida, puesto que la plataforma móvil esta soportada por varias extremidades al mismo tiempo.

Esta robustez es su mayor virtud frente a los robots en configuración serie. Sin embargo, presentan otro tipo de inconvenientes: en primer lugar, la limitación de su espacio de trabajo, pues las extremidades pueden colisionar, y los actuadores tienen sus propios límites físicos. Otro problema que suelen presentar, es la pérdida de robustez cuando se alcanza una posición singular, provocando que aumente sus grados de libertad, y volviéndose incontrolable.

Uno de los primeros (y más famosos) robots paralelos es la conocida como plataforma de Stewart (o plataforma de Stewart-Gough) [19]. Constituida por tres extremidades, cada una de las cuales, con dos actuadores, permitía mover y orientas la plataforma con seis grados de libertad, pues su objetivo era el de ser usado para los simuladores de vuelo. Los trabajos e informes de D. Stewart tendrían un gran impacto, siendo uno de los precursores en el campo de los robots paralelos.

Figura 2.7. Izda: Plataforma de Stewart-Gough [16], Dcha: Robot delta de FANUC [20]

Actualmente, la configuración de robot paralelo más extendido es la del robot delta, consistente en tres extremidades conectadas a juntas universales en la base. La clave de este diseño es el uso de paralelogramos en los brazos,

lo que permite mantener la orientación del extremo. Son habitualmente utilizados para aplicaciones de 'pick&place', debido a la gran velocidad de funcionamiento que pueden alcanzar.

El objetivo del robot usado en este trabajo es el de realizar tareas de rehabilitación del tren inferior, mayormente las articulaciones tobillo y rodilla, que no requerirán de grandes rangos de movimiento, pero sí que será necesario que el robot tenga suficiente rigidez y estabilidad estructural para soportar las fuerzas ejercidas por el paciente durante las tareas de rehabilitación. Por ello, la configuración en paralelo resulta la más idónea para estas tareas.

3 Modelo matemático del robot 3UPE-RPU

3.1 Introducción

Para poder realizar el control de un robot, es necesario disponer de un modelo matemático suficientemente bueno que describa el comportamiento del robot y que permita: por un lado, determinar la posición del robot en cada instante conocidos los valores de sus actuadores, y por otro, ser capaces de determinar la configuración adecuada para alcanzar una determinada posición del extremo del robot. Esta cuestión responde al problema directo e inverso respectivamente del control de robots.

Se puede aproximar este problema, a través del estudio cinemático del sistema mecánico, es decir establecer la relación entre las posiciones, velocidades y aceleraciones de sus eslabones, sin tener en cuenta las fuerzas que intervienen en dicho movimiento. El problema cinemático directo determinará la posición y orientación del elemento terminal del robot, en función del valor de sus juntas activas. Mientras que el problema cinemático inverso trata de calcular el valor necesario de las juntas activas para alcanzar una determinada posición y orientación del elemento terminal.

La otra aproximación, sería a través del estudio dinámico del sistema robotizado, es decir realizar un análisis de la posición y orientación, teniendo en cuenta las fuerzas y pares que actúan en el movimiento, tales como términos inerciales, fuerzas de coriolis, el peso del propio robot, etc. Este análisis presenta una complejidad inherente en los robots paralelos, debido a su estructura en bucle cerrado, y las restricciones cinemáticas, pero resulta de vital importancia para su control, para lograr un posicionamiento preciso y un correcto funcionamiento dinámico bajo grandes cargas.

3.2 Modelado

El siguiente desarrollo se presenta como un resumen del trabajo realizado por el Dr. Vicente Mata y adaptada a la última versión del robot por el doctorando José L. Pulloquinga. Para una revisión más pormenorizada consultar la documentación completa [21].

3.2.1 Modelo Cinemático

Para realizar el análisis cinemático, se aplicará la notación de Denavit-Hartenberg (DH) bajo la convención de Paul, a cada una de las paras del robot.

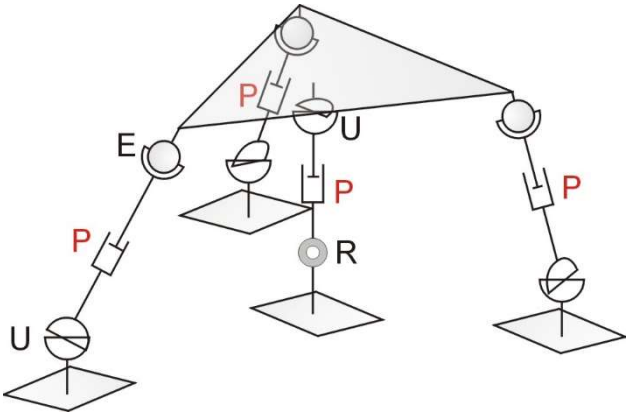


Figura 3.1: Figura esquemática del robot 3UPE-RPU [21]

Se trata de un robot paralelo de cuatro extremidades en dos configuraciones distintas: por un lado, contará con tres patas UPE, con pares cinemáticos universal - prismático - esférico, por otro lado, una pata central de tipo RPU, pares rotación - prismático - universal. Las patas sustentarán la plataforma superior móvil, con sistema de referencia móvil $\{O_M - X_M Y_M Z_M\}$. Y todo el conjunto estará sobre una plataforma fija, a la que estará ligado el sistema de referencia fijo $\{O_F - X_F Y_F Z_F\}$.

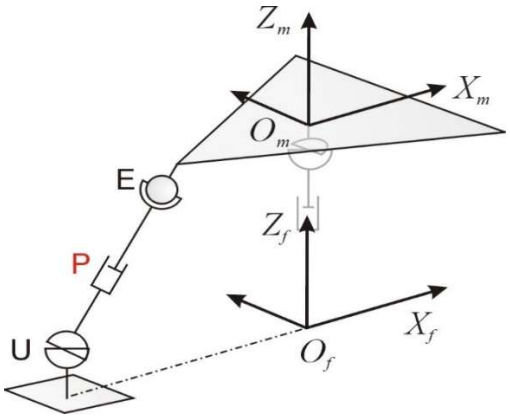


Figura 3.2: Sistemas de referencia fijo y móvil [21]

Los pares cinemáticos universales pueden modelarse como dos pares de rotación perpendiculares, y por su lado los esféricos al habilitar la rotación en las tres direcciones del espacio, se modelan como tres pares de rotación.

Por tanto, las patas UPE estarán compuestas por 'seis eslabones virtuales', mientras que la pata RPU constará de solo cuatro. En las siguientes tablas se recogen sus parámetros de Denavit-Hartenberg:

Barra i-ésima	α_i	a_i	d_i	θ_i
1	$-\pi/2$	0	0	q_1
2	$\pi/2$	0	0	q_2
3	0	0	q_3	0
4	$\pi/2$	0	0	q_4
5	$\pi/2$	0	0	q_5
6	$\pi/2$	0	0	q_6

Tabla 3.1: Parámetros Denavit-Hartenberg de pata UPE

Barra i-ésima	α_i	a_i	d_i	θ_i
1	$-\pi/2$	0	0	q_1
2	$-\pi/2$	0	q_2	π
3	$\pi/2$	0	0	q_3
4	0	0	0	q_4

Tabla 3.2: Parámetros Denavit-Hartenberg de pata RPU

Cinemática inversa

Dada la orientación y posición del elemento terminal $\{x_m, y_m, z_m, \phi, \theta, \psi\}$, se trata de determinar el valor de las juntas activas (actuadores), que serán los valores de las juntas prismáticas correspondientes a q_3 en las patas UPE, y q_2 en la pata RPU.

Será necesario adicionalmente establecer la relación entre el sistema de referencia fijo del robot, y el fijo a cada una de las patas de este, que definiremos en cada caso mediante su matriz de rotación ${}^{f}R_{i0}$, y vector de traslación :

$$ {}^{f}R_{i0} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \tag{3.1} $$

La matriz de rotación es idéntica para todas las patas, cambiando el vector de traslación:

$$
\begin{aligned}
\textit{Pata 1:} \quad & \vec{r}_{O_f A} = \begin{bmatrix} -R \\ 0 \\ 0 \end{bmatrix} \\
\textit{Pata 2:} \quad & \vec{r}_{O_f B} = \begin{bmatrix} R\ \cos(\beta) \\ R\ sen(\beta) \\ 0 \end{bmatrix} \\
\textit{Pata 3:} \quad & \vec{r}_{O_f C} = \begin{bmatrix} R\ \cos(\beta) \\ -R\ sen(\beta) \\ 0 \end{bmatrix} \\
\textit{Pata 4:} \quad & \vec{r}_{O_f O_f} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}
\end{aligned}
\tag{3.2}
$$

Resolviendo los lazo cinemáticos, y teniendo en cuenta que dada la configuración de la pata central, no se permiten ni desplazamientos en Y_f, ni giros respecto a X_f, esto es que $y_m = \ \phi = 0$, se plantea el sistema de ecuaciones igualando las dos formas de llegar a las posiciones de cada pata.

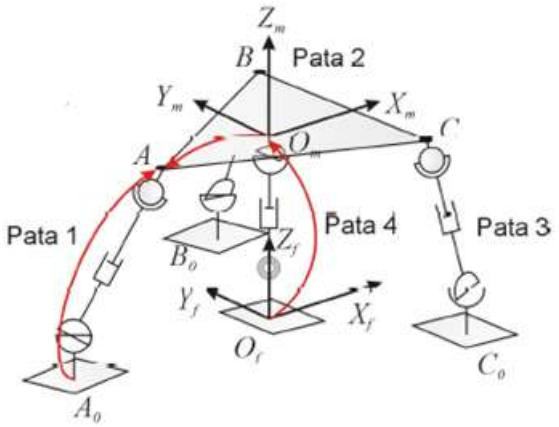


Figura 3.3: Cierre de la cadena cinemática para las coordenadas de los puntos A,B y C [21]

Las coordenadas del punto A por ambos caminos vendrá dada por:

$$
\begin{bmatrix} x_m \\ y_m \\ z_m \end{bmatrix} + {}^{F}R_M \begin{bmatrix} -Rm \\ 0 \\ 0 \end{bmatrix} = [H_{FP1} \cdot H_{01}(q) \cdot H_{12}(q_2) \cdot H_{23}(q_3)] \tag{3.3}
$$

Para el punto B:

$$
\begin{bmatrix} x_m \\ y_m \\ z_m \end{bmatrix} + {}^{F}R_M \begin{bmatrix} R_m\ \cos(\beta) \\ R_m\ sen(\beta) \\ 0 \end{bmatrix} = [H_{FP2} \cdot H_{01}(q) \cdot H_{12}(q_2) \cdot H_{23}(q_3)] \tag{3.4}
$$

Para el punto C:

$$\begin{bmatrix} x_m \\ y_m \\ z_m \end{bmatrix} + {}^{F}R_M \begin{bmatrix} R_m \cos(\beta) \\ -R_m \ sen(\beta) \\ 0 \end{bmatrix} = [H_{FP3} \cdot H_{01}(q) \cdot H_{12}(q_2) \cdot H_{23}(q_3)] \tag{3.5}$$

Y para la pata central:

$$\begin{bmatrix} x_m \\ y_m \\ z_m \end{bmatrix} = [H_{FP4} \cdot H_{01}(q) \cdot H_{12}(q_2)] \tag{3.6}$$

Siendo H_{FPi} la matriz de transformación homogénea desde el origen fijo del robot hasta el origen fijo de cada pata.

A partir de estas ecuaciones y de las restricciones al movimiento impuestas por la pata central, es posible resolver el problema cinemático inverso para obtener el valor de cada una de las articulaciones. El desarrollo completo de la solución puede consultarse en bibliografía [21].

Cinemática directa

El problema cinemático directo consistirá en dadas los valores de las coordenadas generalizadas de cada actuador, determina la posición y orientación de la plataforma móvil, junto con el de las restantes coordenadas generalizas de juntas pasivas.

Considerando las siguientes condiciones de cierre para obtener las ecuaciones de restricción:

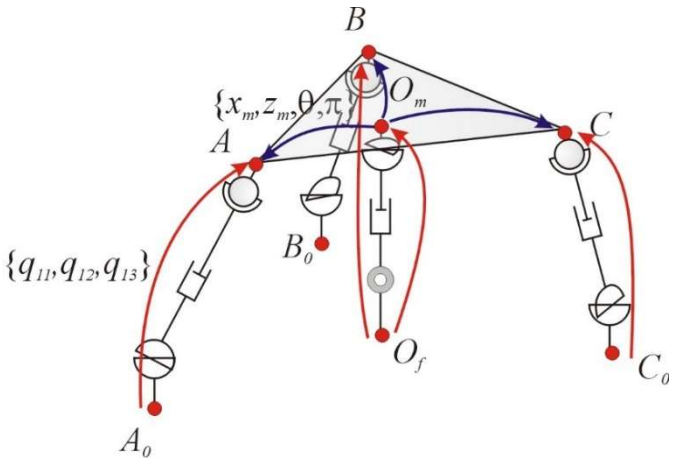


Figura 3.4: Planteamiento de las ecuaciones de restricción [21]

Se plantea para cada pata la condición vectorial de cierre:

$$\vec{r}_{A_0A}(q_{11}, q_{12}, q_{13}) = \begin{bmatrix} x_m \\ 0 \\ z_m \end{bmatrix} + {}^fR_m(\theta, \psi) \begin{bmatrix} -R \\ 0 \\ 0 \end{bmatrix} \tag{3.7}$$

$$\vec{r}_{B_0B}(q_{21}, q_{22}, q_{23}) = \begin{bmatrix} x_m \\ 0 \\ z_m \end{bmatrix} + {}^fR_m(\theta, \psi) \begin{bmatrix} R\cos(\beta) \\ R\ sen(\beta) \\ 0 \end{bmatrix} \tag{3.8}$$

$$\vec{r}_{C_0C}(q_{31}, q_{32}, q_{33}) = \begin{bmatrix} x_m \\ 0 \\ z_m \end{bmatrix} + {}^fR_m(\theta, \psi) \begin{bmatrix} R\cos(\beta) \\ -R\ sen(\beta) \\ 0 \end{bmatrix} \tag{3.9}$$

$$\vec{r}_{O_fO_m}(q_{41}, q_{42}) = \begin{bmatrix} x_m \\ 0 \\ z_m \end{bmatrix} \tag{3.10}$$

Desarrollando estas expresiones llegamos a un sistema de 11 ecuaciones no lineales, con 11 incógnitas distintas:

$$\Phi_i(q_{11}, q_{12}, q_{21}, q_{22}, q_{31}, q_{32}, q_{41}, x_m, z_m, \phi, \psi) = 0, \quad i = 1,2,...\,11 \tag{3.11}$$

Por lo tanto, podríamos resolver el sistema mediante diversas metodologías como la de Newton-Raphson.

3.2.2 Modelo Dinámico

En el modelo dinámico se realizará el análisis de fuerzas y momentos que afectan a los distintos componentes del robot a fin de poder determinar las fuerzas o acciones de control necesarias para lograr el desplazamiento.

Para ello se va utilizar la metodología del análisis sólido rígido simplificado, donde solo consideraremos los sólidos de la plataforma superior y las cuatro patas, cada pata a su vez estará compuesta por dos eslabones: un cilindro y un émbolo.

Fuerzas inerciales

La primera de las fuerzas que intervendrá en la dinámica del robot, son las fuerzas inerciales derivadas de la velocidad y aceleración del robot durante su desplazamiento. Se desarrolla a continuación cómo se modela la pata 1:

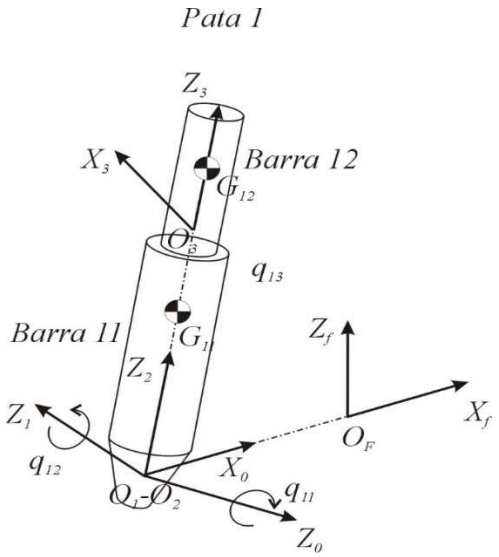


Figura 3.5: Modelado dinámico de la pata 1 [21]

Todas las magnitudes están expresadas atendiendo al sistema de referencia fijo del robot, salvo que se indique lo contrario. De acuerdo al esquema de la Figura 3.5 podemos calcular las velocidades angulares como:

$$\begin{aligned} \vec{\omega}_{11} &= \dot{q}_1 \cdot \vec{u}_{z_0} + \dot{q}_{12} \cdot \vec{u}_{z_1} \\ \vec{\omega}_{12} &= \vec{\omega}_{11} \end{aligned} \tag{3.12}$$

Siendo:

$$\begin{aligned} \vec{u}_{z_0} &= [0 \quad -1 \quad 0]^T \\ \vec{u}_{z_1} &= [-sen(q_1) \quad 0 \quad 0]^T \end{aligned} \tag{3.13}$$

Y las aceleraciones angulares de las dos barras móviles que constituyen la pata 1 serán:

$$\vec{\alpha}_{11} = \vec{\alpha}_{12} = \ddot{q}_{11} \cdot \vec{u}_{z_0} + \ddot{q}_{12} \cdot \vec{u}_{z_1} + \dot{q}_{12} \cdot (\tilde{w}_1 \cdot {}^f R_1 \cdot {}^1\vec{u}_{z_1}) \tag{3.14}$$

Conocidas las velocidades y aceleraciones angulares, podemos calcular la velocidad y aceleración en el centro de gravedad de cada barra:

$$\begin{aligned} \vec{v}_{G_{11}} &= \vec{\omega}_{11} \times \vec{r}_{O_2 G_{11}} \\ \vec{v}_{G_{12}} &= \vec{v}_{O_2} + \vec{\omega}_{11} \times \vec{r}_{O_2 G_{12}} + \dot{q}_{13} \cdot \vec{u}_{z_2} \end{aligned} \tag{3.15}$$

$$\begin{aligned} \vec{a}_{G_{11}} &= \vec{\omega}_{11} \times (\vec{\omega}_{11} \times \vec{r}_{O_2 G_{11}}) + \vec{\alpha}_{11} \times \vec{r}_{O_2 G_{11}} \\ \vec{a}_{G_{12}} &= \vec{v}_{O_2} + \vec{\omega}_{11} \times (\vec{\omega}_{11} \times \vec{r}_{O_2 G_{12}}) + \vec{\alpha}_{11} \times \vec{r}_{O_2 G_{12}} + \ddot{q}_{13} \cdot \vec{u}_{z_2} + 2 \cdot (\vec{\omega}_{11} \times (\dot{q}_{13} \cdot \vec{u}_{z_2})) \end{aligned} \tag{3.16}$$

Y por tanto las acciones inerciales serán, conocidas las masas m y los tensores incerciales locales I_G, de la siguiente forma:

$$\begin{aligned} \vec{F}_{in_{11}} &= -m_{11} \cdot \vec{a}_{G_{11}} \\ \vec{F}_{in_{12}} &= -m_{12} \cdot \vec{a}_{G_{12}} \end{aligned} \tag{3.17}$$

$$\begin{aligned} \vec{T}_{in_{11}} &= -(I_{G_{11}} \cdot \vec{\alpha}_{11} + \vec{\omega}_{11}(I_{G_{11}} \cdot \vec{\omega}_{11})) \\ \vec{T}_{in_{12}} &= -(I_{G_{12}} \cdot \vec{\alpha}_{12} + \vec{\omega}_{12}(I_{G_{12}} \cdot \vec{\omega}_{12})) \end{aligned} \tag{3.18}$$

Para el caso de las patas 2 y 3, el procedimiento y resultado es análogo, y para la pata central 4 y la plataforma móvil se sigue el mismo procedimiento, de acuerdo al modelo dinámico de la Figura 3.6. El desarrollo completo para estos sólidos, no se incluye en esta memoria.

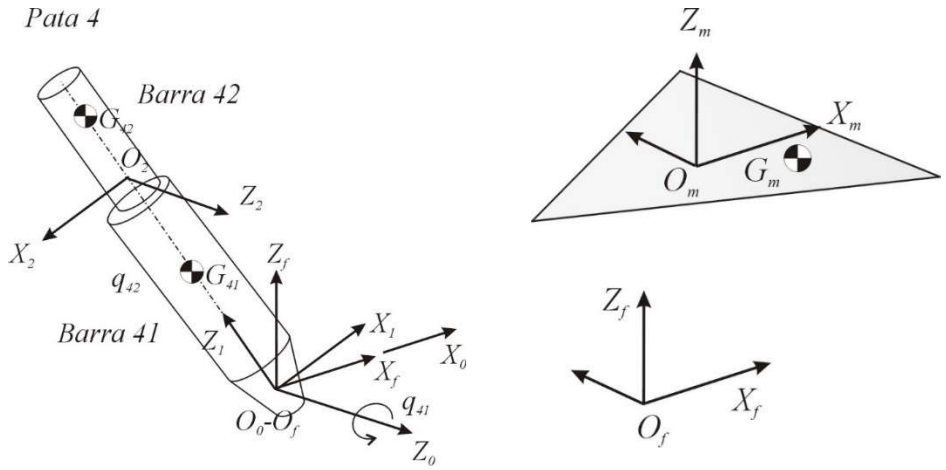


Figura 3.6. Izda: modelo dinámico de la pata 4, Dcha: modelo dinámico de la plataforma móvil [21]

En resumen, quedaría la ecuación generalizada de las fuerzas de inercia, tras incorporar los términos de cada una de los eslabones de cada pata y de la plataforma, de la siguiente forma:

$$\vec{Q}_{in} = \vec{F}_{in_{11}} \frac{\partial \vec{v}_{G_{11}}}{\partial \dot{\vec{q}}} + \vec{T}_{in_{11}} \frac{\partial \vec{\omega}_{11}}{\partial \dot{\vec{q}}} + \cdots + \vec{F}_{in_m} \frac{\partial \vec{v}_{G_m}}{\partial \dot{\vec{q}}} + \vec{T}_{in_m} \frac{\partial \vec{\omega}_m}{\partial \dot{\vec{q}}} \quad (3.19)$$

Pesos

También debemos considerar la dinámica que introducen los propios pesos de las barras móviles del mecanismo, cuya expresión generalizada será:

$$\vec{Q}_{grav} = \vec{P}_{11} \frac{\partial \vec{v}_{G_{11}}}{\partial \dot{\vec{q}}} + \vec{P}_{12} \frac{\partial \vec{v}_{G_{12}}}{\partial \dot{\vec{q}}} + \cdots + \vec{P}_m \frac{\partial \vec{v}_{G_m}}{\partial \dot{\vec{q}}} \quad (3.20)$$

Fuerzas externas activas

Como acciones externas activas se cuenta con las fuerzas introducidas por cada uno de los cuatro actuadores. Estas coincidirán en la dirección de actuación con las cuatro coordenadas generalizadas independientes.

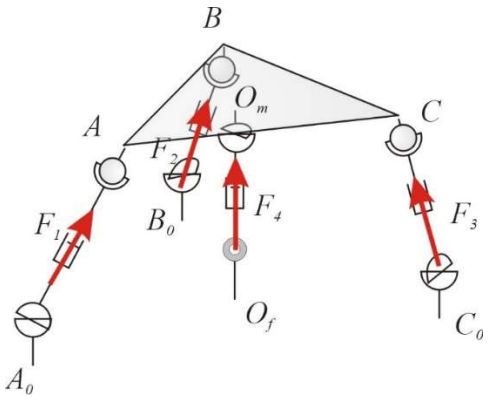


Figura 3.7: Acciones externas activas [21]

Quedando la ecuación de fuerzas activas generalizadas:

$$\vec{Q}_{ex} = \vec{F}_1 \frac{\partial \vec{v}_{G_{12}}}{\partial \dot{\vec{q}}} + \vec{F}_2 \frac{\partial \vec{v}_{G_{22}}}{\partial \dot{\vec{q}}} + \vec{F}_3 \frac{\partial \vec{v}_{G_{32}}}{\partial \dot{\vec{q}}} + \vec{F}_4 \frac{\partial \vec{v}_{G_{42}}}{\partial \dot{\vec{q}}} \quad (3.21)$$

Fuerzas activas aplicadas por el entorno

El último tipo de fuerzas que contribuyen a la dinámica del robot, es la fuerza externa que puede ejercer el paciente durante la rehabilitación. Estas fuerzas se presuponen conocidas, y estarán compuestas por una fuerza y un par definidos respecto al sistema de referencia asociado a la plataforma móvil.

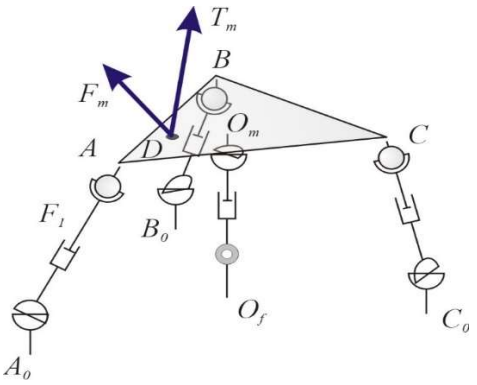


Figura 3.8: Acciones externas aplicadas por el entorno [21]

Su correspondiente ecuación generalizada es:

$$\vec{Q}_{ent} = \vec{F}_m \frac{\partial \vec{v}_D}{\partial \vec{q}} + \vec{T}_m \frac{\partial \vec{\omega}_m}{\partial \vec{q}} \tag{3.22}$$

Ecuación del movimiento en forma aumentada

Queda por tanto definida la dinámica del robot mediante la siguiente expresión:

$$\vec{Q}_{in} + \vec{Q}_{grav} + \vec{Q}_{ex} + \vec{Q}_{ent} - [\Phi_q]^T \cdot \vec{\lambda} = 0 \tag{3.23}$$

Donde $[\Phi_q]^T$ se corresponde con el jacobiano de las restricciones, y $\vec{\lambda}$ el vector de multiplicadores indeterminados de Lagrange.

Esto nos dará un sistema con 15 ecuaciones escalares, y 25 incógnitas (las aceleraciones (15) más los multiplicadores (10)), faltando entonces 10 ecuaciones adicionales, que se obtienen derivando respecto al tiempo dos veces, las ecuaciones de restricción. Obteniendo la denominada formulación aumentada, que se puede expresar de forma matricial (agrupando los términos que incluyen incógnitas a la izquierda):

$$\begin{bmatrix} M_{15\times15} & [\Phi_q]^T_{15\times11} \\ [\Phi_q]_{11\times15} & 0_{11\times11} \end{bmatrix} \cdot \begin{bmatrix} \vec{\ddot{q}}_{15\times1} \\ \vec{\lambda}_{11\times1} \end{bmatrix} = \begin{bmatrix} \vec{Q}_{cyc} + \vec{Q}_{grav} + \vec{Q}_{ex} + \vec{Q}_{ent} \\ \vec{b} \end{bmatrix} \tag{3.24}$$

Donde $\vec{Q}_{cyc}$ son las fuerzas generalizadas que dependen del cuadrado de las velocidades generalizadas (términos centrífugos y de Coriolis).

4 Materiales e Implementación

4.1 Hardware y Software

4.1.1 Robot y sistema de captura

Todo el material e instalaciones utilizadas durante este proyecto han sido facilitadas por el laboratorio de Mebiomec de la UPV [22].

El robot con el que se ha trabajado es un robot paralelo con cuatro grados de libertad, cuya configuración 3UPE- RPU y modelo ya se ha descrito anteriormente.

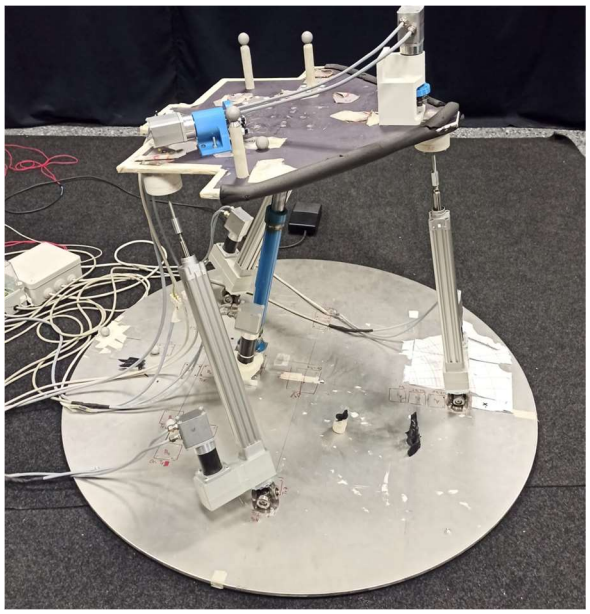


Figura 4.1. Robot paralelo de 4 grados de libertad 3UPE-RPU

Para realizar el seguimiento del robot, y conocer su posición (y en consecuencia su velocidad y aceleración) se utilizan marcadores que permiten triangular la posición tanto de la base fija del robot, como de la plataforma móvil mediante un sistema de cámaras instaladas en el laboratorio con ese fin.

4.1.2 Software de simulación

La implementación de los modelos y simulaciones previas a las pruebas sobre el robot real, se han llevado a cabo mediante el uso del software de Matlab-Simulink. Este completo software provee al usuario de un lenguaje de programación propio, integrado en un entorno de desarrollo estable e intuitivo, orientado al cálculo numérico, y vectorial.

La capa de simulink es la encargada de realizar la simulación de sistemas modelados en tiempo continuo o discretos, herramienta muy extendida para la simulación de todo tipo de tareas de control. Cuenta además con ciertos paquetes de herramientas adicionales (toolboxes), que expanden las funcionalidades de programa. Para este trabajo se ha necesitado de la 'Control Toolbox' que incorpora funciones específicas para el modelado de sistemas de control, análisis y ajuste de controladores, etc.

Adicionalmente, se ha requerido de la implementación de las librerías para Matlab del algoritmo LWPR [23], que incluirá todas las funciones y parámetros necesarios, para poder crear modelos LWPR, entrenarlos y realizar las predicciones.

4.1.3 Software del robot

La implementación final en el robot real requiere del desarrollo de los algoritmos de control en un lenguaje interpretable por el hardware de control. El sistema de control y comunicaciones se basa en las librerías de ROS (Robot Operation System), implementado en lenguaje C++.

ROS es un marco de trabajo flexible, cuyo objetivo es crear un conjunto de herramientas y bibliotecas que simplifiquen las tareas de desarrollar comportamiento robóticos complejos y robustos para todo tipo de plataformas [24].

Mediante ROS se define un sistema de comunicación entre hardware y software basado en la comunicación entre nodos. Habiendo distintas clases de nodos: de control, de computo, de lectura de hardware, y se debe definir el esquema de comunicación entre todos ellos. Algunos nodos establecerán

la comunicación con los elementos hardware, realizando tareas de lectura o escritura, mientras que otros no tienen relación con el mundo físico y solo ejecutarán tareas de cálculo o de coordinación en las comunicaciones. Además, una de las características imprescindibles de ROS, es la capacidad de establecer comunicación en tiempo real, imprescindible para realizar el control con un periodo de muestreo concreto.

Adicionalmente, se instalaron y utilizaron dos librerías fundamentales para desarrollar el código necesario:

- Librería LWPR para C++: esta librería cuenta con las clases, funciones y parámetros necesarios para poder implementar en cualquier programa el algoritmo LWPR [23].
- Librería Eigen3: librería para cálculo, que permite trabajar con vectores y matrices cómodamente en C++. [25]

4.2 Implementación

Durante el desarrollo de los distintos escenarios de control, y la implementación de los distintos algoritmos, se siguió la siguiente metodología:

En primer lugar, se comenzó por la revisión bibliográfica de las técnicas que se utilizarían, para estudiar cómo adaptar su implementación y parámetros de acuerdo a referencias en trabajos previos.

Una vez estudiado el caso, siempre se comenzó implementando los algoritmos en simulación, para verificar el comportamiento de los controladores desarrollados, y realizar un primer ajuste de los hiperparámetros necesarios. Y así, tener un comportamiento seguro y medianamente robusto a la hora de trabajar sobre el robot real.

A continuación, y una vez validados en simulación se desarrolló la implementación en C++ para integrar los algoritmos en el hardware de control y comunicación. Toda la información relevante durante los experimentos se guarda, para poder revisarla posteriormente.

Finalmente, aprovechando los datos recogidos se realizó un análisis pormenorizado de los distintos datos recogidos, a fin de detectar errores, fuente de mejora, y verificar que el comportamiento real se corresponde con el esperado en simulación.

Con esta metodología se ha desarrollado toda la etapa experimental del proyecto. De acuerdo con los objetivos del proyecto, pueden diferenciarse tres grandes escenarios de implementación diferenciados a desarrollar.

4.2.1 LWPR como controlador del robot

El objetivo principal del trabajo, es aplicar las técnicas de regresión locales en línea, al control del robot de rehabilitación del laboratorio, a fin de lograr el seguimiento de trayectorias predefinidas con un error relativamente bajo. Además, el controlador debe ser capaz de generalizar en distintas trayectorias manteniendo un buen comportamiento.

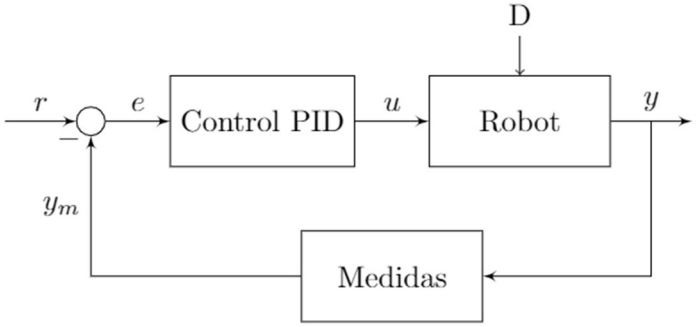


Figura 4.2: Esquema de control clásico mediante realimentación del error y PID

En este caso el algoritmo LWPR recibirá como entradas X: las referencias en posición, velocidad y aceleración para cada una de las juntas, y la salida Y a predecir será el valor de actuación de estas juntas. Puesto que tenemos cuatro juntas prismáticas (una por extremidad), y tal y como se discutió en la sección 2.2 en realidad tendremos que entrenar cuatro modelos distintos, uno por salida a predecir. Todos recibirán las entradas citadas, y cada uno predecirá una de las cuatro juntas.

Sin embargo, para garantizar la estabilidad del sistema es necesario complementar el control 'feedforward' del LWPR, con un control 'feedback' de la señal de error, que compense los errores y mantenga el robot estabilizado. El controlador 'feedback' será un PID clásico, cuyas ganancias ya han sido diseñadas y ajustados en trabajos anteriores con el robot.

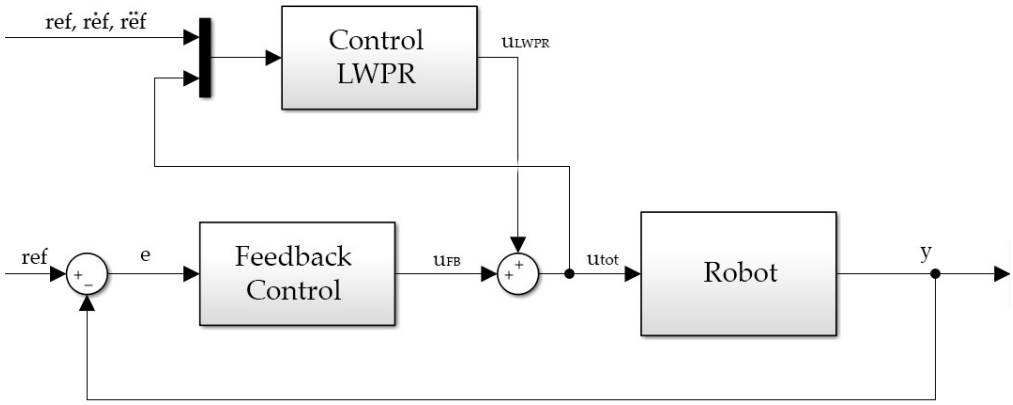


Figura 4.3: Esquema de control mediante un módulo de control LWPR

De esta forma la estructura de control quedará de la siguiente forma, en cada periodo de muestreo el módulo LWPR recibirá la posición, velocidad y aceleración de referencia, y predecirá la señal de control estimada por sus modelos, simultáneamente el controlador PID corregirá el error de posición del robot. La señal de control total (LWPR+PID), será la que utiliza el algoritmo LWPR para realizar el aprendizaje y la actualización de sus modelos.

A fin de garantizar la estabilidad en el comportamiento del robot real, durante las primeras iteraciones y mientras el algoritmo se está entrenando por primera vez, el módulo LWPR estará desacoplado del control, aunque este siga siendo entrenado, sus predicciones no serán tenidas en cuenta. Por lo que podemos diferencias dos modos de funcionamiento:

- Modo entrenamiento: en este, el módulo LWPR no actúan sobre el robot, quedando trabajando el PID en solitario. El algoritmo sí que aprende y actualiza los modelos, y las predicciones si bien se pueden calcular para verificar su comportamiento, no se toman en cuenta desde el punto de vista del control.

- Modo control LWPR: en este el módulo LWPR trabaja solidariamente al PID. El algoritmo sigue aprendiendo y actualizando los modelos si fuese preciso, y deben calcularse las predicciones en cada periodo de muestreo pues estas se agregarán a la señal de control.

La idea principal de esta implementación, es que cuando el algoritmo enfrente a alguna de sus trayectorias por primera vez, dejarlo en modo entrenamiento, y una vez verificado que está prediciendo correctamente, sumarlo al control. Así, se evita que un error en las primeras predicciones, cuando aún no existen modelos creados, pueda dar lugar a una actuación muy agresiva que pueda descontrolar el robot, o romper alguna pieza. El objetivo es que una vez entrenado en varias trayectorias distintas, no sea necesario volver a usar el modo entrenamiento, pues lo modelos sean suficientemente robustos y refinados, como para adaptarse a nuevos espacios de entrada sin producir inestabilidades.

Aunque esté el módulo PID trabajando conjuntamente con el módulo LWPR, este último debería realizar la mayoría del esfuerzo de control, quedando relegado el PID a corregir errores puntuales, vibraciones, ruidos, … Se puede apreciar el paralelismo entre el trabajo que estará realizando el algoritmo y el problema dinámico inverso de la robótica, pues se estará calculando el valor de las actuaciones en función de la posición del robot.

4.2.2 LWPR para predecir la posición

El algoritmo LWPR presenta una gran versatilidad, y no solo se puede utilizar para realizar el control del robot, sino que también se ha utilizado para estimar la posición futura del robot en función de la posición actual, y la acción de control.

Esto podría permitir sustituir el complejo modelo matemático necesario para simular el comportamiento del robot, donde debemos realizar un análisis cinemático y dinámico completo del robot, y todos sus componentes, con la alta complejidad que esto supone.

En este caso las entradas X que alimentan al algoritmo serán la posición y velocidad actual del robot, además de la acción de control realizada por el controlador, mientras que la salida del modelo Y será la predicción de la nueva posición que alcanzará el robot. Nuevamente, será necesario generar cuatro modelos, pues la posición del robot la definiremos por sus cuatro coordenadas prismáticas. Como se aprecia en la figura 4.2, se añadió un filtro para la velocidad, como se justificará en los resultados presentados en la siguiente sección:

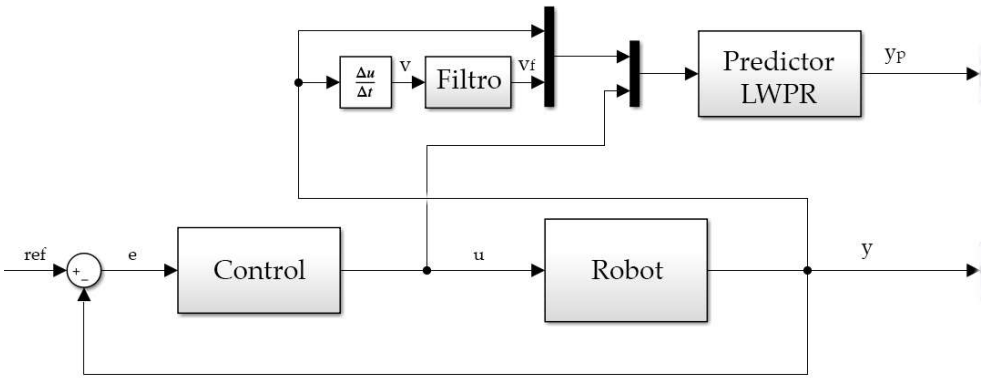


Figura 4.4: Esquema de funcionamiento del LWPR para predecir posiciones futuras

Realmente, el algoritmo LWPR no actúa sobre el esquema de control, por lo que realmente podríamos realizar el entrenamiento de este modelo fuera de línea, utilizando los datos obtenidos de la ejecución de las trayectorias.

En este caso estamos resolviendo un problema similar al del modelo dinámico directo, pues obtenemos la posición del robot en función de la fuerza de los actuadores (y de la posición de partida).

4.2.3 Control LWPR combinado con ILC

Finalmente, y a fin de mejorar aún más el comportamiento del controlador LWPR, se incorporó paralelamente un módulo de control iterativo ILC en su formato PD, que se utilizará exclusivamente para el seguimiento de una misma trayectoria repetidas veces.

El esquema de control es similar al del apartado 4.2.1, pero en este caso la señal de control será suma de a actuación del módulo LWPR, la realimentación mediante el PID, y la actuación del ILC. El módulo ILC estará compuesto por dos partes principales: en primer lugar, el módulo de control que utiliza la información de la iteración anterior para el cálculo de la señal de actuación, y por otro la memoria en la que están almacenados los datos de la iteración anterior, y donde se almacenarán los de esta.

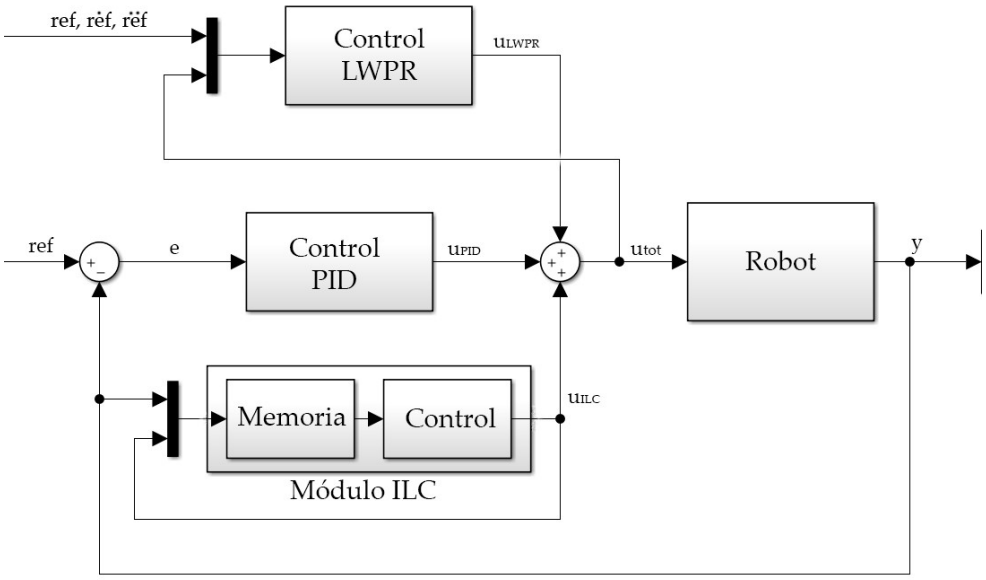


Figura 4.5: Control iterativo implementado junto al control LWPR

De acuerdo con la ley de control (2.18) propuesta con anterioridad, es necesario almacenar en memoria el error en cada instante y la actuación del ILC. En función de estos valores, y las ganancias de control correctamente dimensionadas se realizará el cálculo de la acción de control ILC, que tratará de anticiparse al error (que ya conoce de la iteración anterior) y corregir la actuación antes de que se produzca.

De este modo la implementación debe reducir los errores de posición paulatinamente a lo largo de las iteraciones sobre la misma trayectoria.

5 Resultados

A continuación, se discutirán los resultados más importantes obtenidos en durante el desarrollo de la parte experimental del proyecto.

Tanto para las simulaciones, como para los experimentos reales, se han utilizado una serie de trayectorias diseñadas en trabajos previos, orientadas a realizar movimientos típicos de tareas de rehabilitación para las que está diseñado el robot.

5.1 Experimentos LWPR como controlador

Aplicando la implementación discutida en la sección 4.2, se comprobó el funcionamiento del esquema de control en diversas trayectorias, y ajustando los hiperparámetros del algoritmo hasta lograr el mejor desempeño en los experimentos reales. A continuación, se discutirán algunos de los resultados más importantes, así como algunos detalles importantes de la implementación final respecto a lo visto anteriormente.

En las primeras implementaciones de simulación se detectó un importante problema, que hasta ahora no se ha tratado. Como sucede con la mayoría de técnicas de inteligencia artificial, y en particular en este algoritmo, tener los datos de entrada y salida bien escalados es muy importante para facilitar y acelerar el aprendizaje.

De acuerdo con la documentación del algoritmo [26], este escalado no debe realizarse 'a ciegas', mediante la desviación típica de los datos:

$$x_{norm} = \frac{x - \bar{x}}{\sigma} \tag{5.1}$$

En todo caso, lo recomendable es realizar una normalización en base al rango estimado de las entradas, dividiéndolas por este valor para tener todo en escalas similares entorno al valor unitario. Como puede observarse en la figura 5.1, el rango de trabajo de las distintas variables está en órdenes de magnitud completamente distintos.

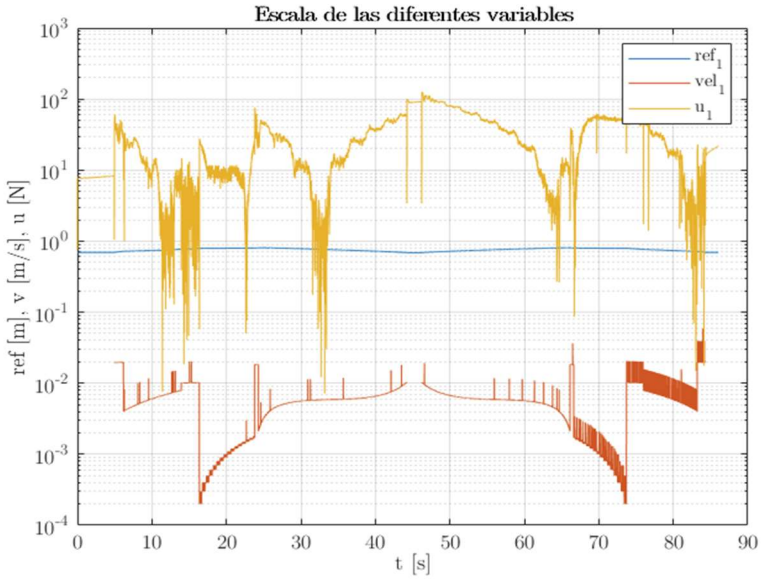
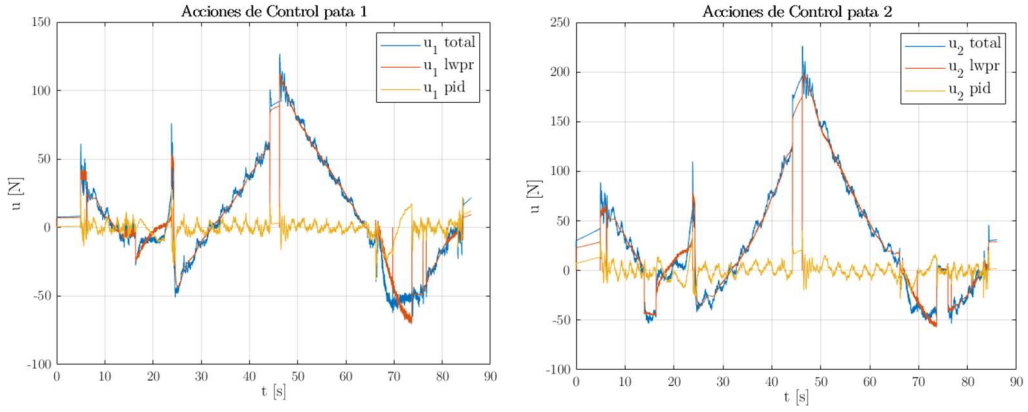


Figura 5.1: Comparativa logarítmica del valor de las variables de entrada/salida al algoritmo

Este escalado también debe hacerse con la salida del algoritmo (fuerzas aplicadas por los actuadores), pues resulta más conveniente predecir también valores entorno a la unidad. Luego estas predicciones habrá que reescalarlas para volver a estar en el rango de actuación.

Una vez subsanado este mal, se logró una implementación satisfactoria en simulación, y se pudo trasladar al robot real. A continuación, se observan las acciones de control en el modo de control LWPR sobre una trayectoria, tras haber realizado previamente una pasada de aprendizaje:



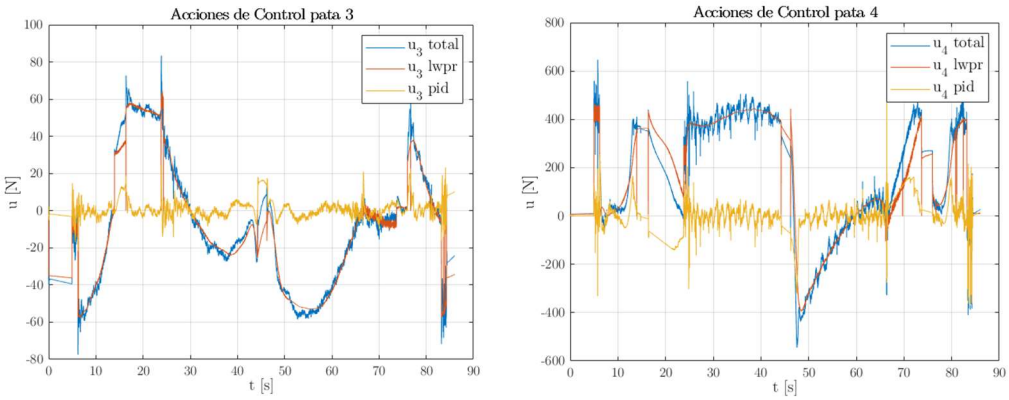
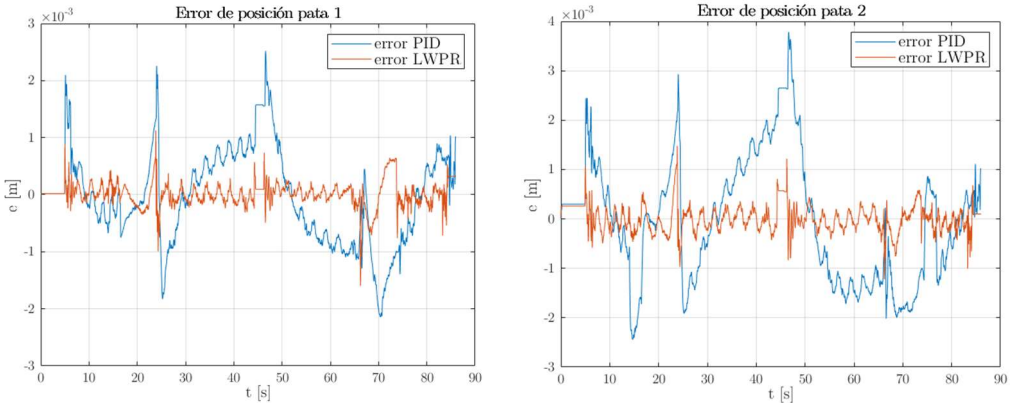


Figura 5.2: Acciones de control para cada una de las patas

Logrando en el modelo de cada una de las patas el comportamiento deseado: el módulo LWPR aprende la dinámica del sistema, y es el encargado de hacer el esfuerzo de control, mientras que la realimentación del error, realiza pequeñas correcciones y garantiza la estabilidad.

Además, el uso del módulo LWPR logra reducir significativamente los errores de posición. Si se comparan los errores producidos cuando esta el módulo de control LWPR activo, con los errores producidos solo por el controlador PID se ha observado lo siguiente:



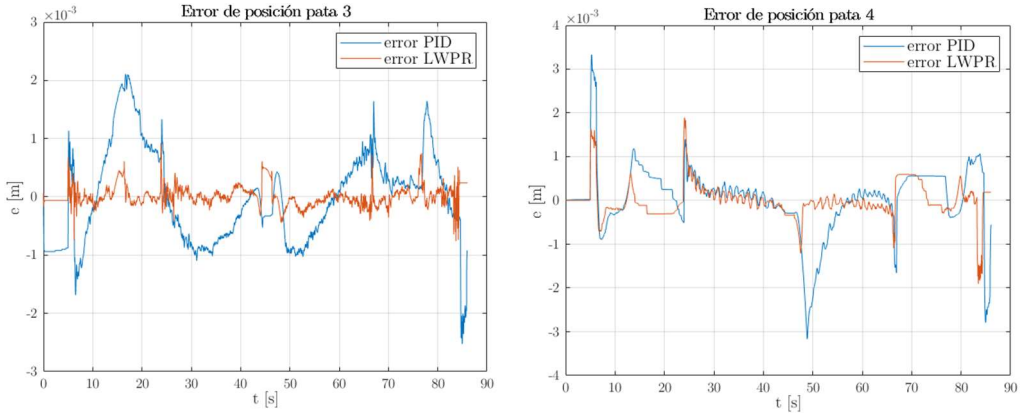


Figura 5.3: comparativa errores absolutos de posición activando el módulo LWPR

El control con la inclusión del algoritmo LWPR logra reducir los errores en casi todas las patas a decimas de milímetro. Por tanto, no solo el algoritmo logra aprender adecuadamente a controlar los actuadores del robot, sino que además resulta en mejoras cuantitativas del desempeño del controlador.

Hasta ahora se han mostrado resultados obtenidos sobre una trayectoria en la que previamente se ha entreno el algoritmo una vez, y posteriormente se repitió la trayectoria en modo control. Sin embargo, el algoritmo debe ser capaz de generalizar sobre nuevas trayectorias, sin perder eficacia en las que ya fue entrenado. Para comprobar su capacidad de generalización se utilizó la siguiente metodología:

En primer lugar, se realizaron sobre varias trayectorias distintas iteraciones consecutivas en modo entrenamiento seguidas de modo control, para asegurar que el algoritmo no sufría de *overfitting* y se adaptaba a nuevos espacios de entrada/salida.

Una vez entrenado en diversas trayectorias, se verificó una de las propiedades con las que a priori cuenta el algoritmo: no olvidar los espacios de trabajo anteriores, en favor de los últimos en los que ha sido entrenado. Para ello, se lanzó directamente en modo control sobre la primera trayectoria en la que fue entrenado, y mantuvo un comportamiento correcto, similar al de la primera iteración.

Finalmente, y para verificar su capacidad de generalización se lanzó el algoritmo directamente controlando sobre trayectorias completamente nuevas, en las que no había sido entrenado previamente. Logrando seguir las referencias perfectamente, y demostrando la velocidad y adaptabilidad del modelo entrenado.

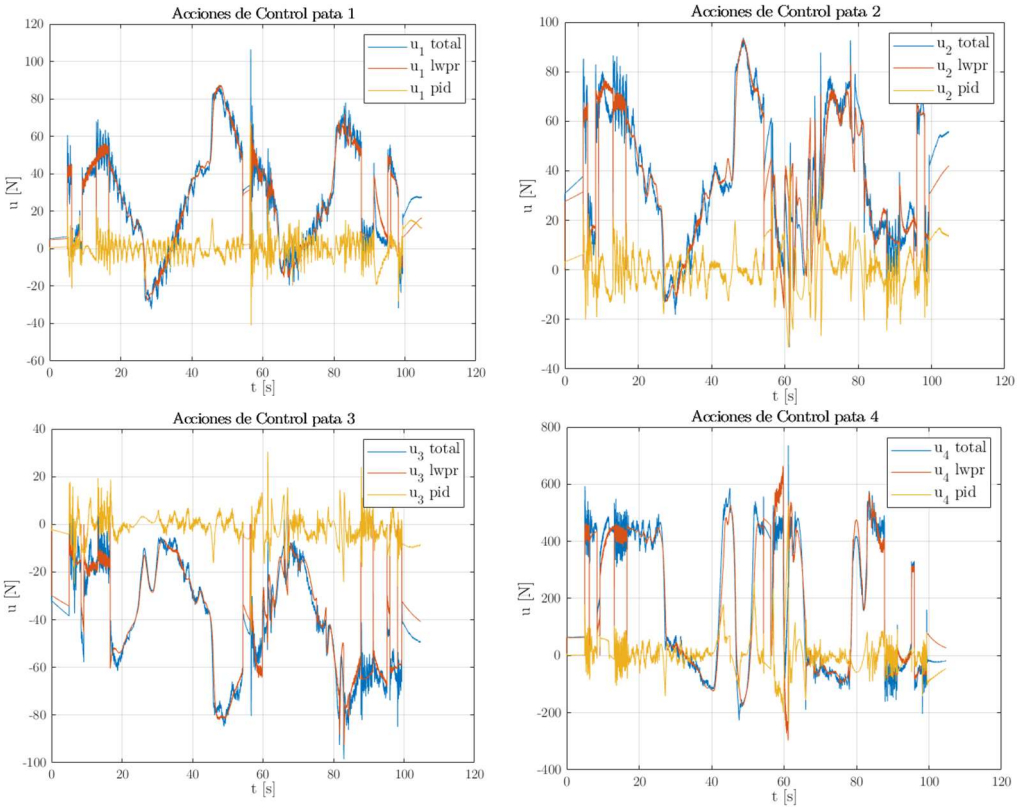


Figura 5.4: Acciones de control sobre una nueva trayectoria no entrenada

En la figura 5.4 se puede observar el comportamiento del controlador en este último caso, en el que debe hacer frente a una nueva trayectoria. El módulo LWPR mantiene un comportamiento eficaz y robusto capaz de generalizar adecuadamente. Aunque aquí las correcciones del PID son más elevadas sobre todo en ciertas zonas de trabajo, donde el algoritmo aún este generando y mejorando sus modelos.

Una de las grandes características fundamentales del algoritmo, que destacábamos en la sección 2.2, es la forma de generar y optimizar sus regresiones locales. Esto le otorga la gran capacidad de generalización que hemos observado, agregando nuevos 'receptive fields' (RF) cuando se encuentra en nuevos espacios locales cuyo valor de activación no supere el umbral w_{gen}.

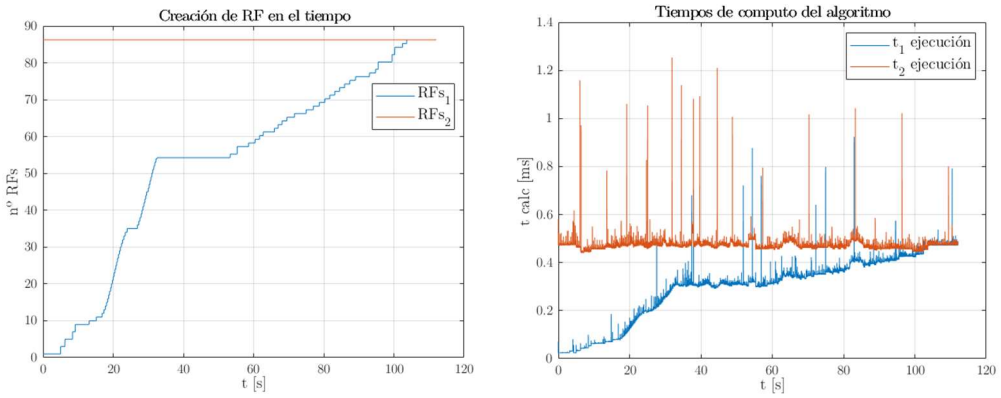


Figura 5.5: Relación entre la generación de nuevos RF y el tiempo de cálculo en dos iteraciones consecutivas sobre la misma trayectoria

En la figura 5.5 izquierda, podemos observamos la generación de RF en la primera iteración del controlador conforme encuentra nuevos espacios de trabajo, y como si posteriormente repetimos la misma trayectoria, no se crearán nuevos RF, tan solo se actualizarán sus parámetros. Conforme el algoritmo crea y almacena nuevas regresiones locales, aumenta su tamaño, y también el coste de cómputo, pues por un lado aumentamos el número de parámetros a optimizar con cada regresión lineal, y por otro la predicción se vuelve combinación de un mayor número de regresiones, aumentando el número de operaciones que debe realizar el algoritmo. Esto se ve fuertemente reflejado en el tiempo de cálculo, si observamos la figura 5.5 derecha, se aprecia la extremada similitud entre el tiempo de cálculo a lo largo del tiempo, y el número de RF. Igualmente, en una segunda pasada como se mantiene el número de RF, los tiempos de cálculo se mantienen estables.

Conforme el algoritmo recibe entrenamiento en nuevas trayectorias y encuentre nuevos espacios de entradas, se generan más RF con su consiguiente aumento del tiempo de cálculo. Mediante el ajuste de la distancia D inicial, podemos limitar el número de RF que se crean (a mayor distancia, aumentamos el número de regresiones locales). Durante la fase experimental, como ya se ha explicado, los modelos fueron entrenados sobre diversas trayectorias hasta que fueron capaces de controlar en cualquier escenario, manteniendo siempre tiempos de cálculo por debajo del milisegundo para este algoritmo, muy por debajo del tiempo de muestreo (10 milisegundos), garantizando un buen funcionamiento del mismo.

5.2 Experimentos LWPR para predecir la posición

El proceso de implementación de este modelo siguió un proceso análogo al del anterior. Sin embargo, uno de los principales elementos diferenciados, es que la predicción de este modelo no es utilizada en ningún caso en el esquema de control, el único objetivo es entrenar al algoritmo para que sea capaz de realizar las predicciones, por lo que es posible realizar el entrenamiento del algoritmo fuera de línea, utilizando los datos recogidos del movimiento y actuación del robot en otros experimentos. Igualmente, en muchas ocasiones se entrenó de manera paralela al módulo LWPR de control, para aprovechar mejor el tiempo en el laboratorio.

Nuevamente, la selección y preprocesado adecuado de las señales con las que se alimenta al módulo de aprendizaje es muy importante para mantener el buen comportamiento del algoritmo. En esta ocasión además del correcto escalado, se ha visto necesario realizar el filtrado de la señal de entrada de la velocidad medida del robot. Esta velocidad es una señal altamente ruidosa, y se filtrará mediante la implementación de un filtro de Kalman.

En este caso se utilizó el diseño de un filtro de Kalman de velocidad y posición, que considera la aceleración nula [27]. Tal y como se aprecia en la figura 5.6 esto es así en casi todas las muestras de la trayectoria:

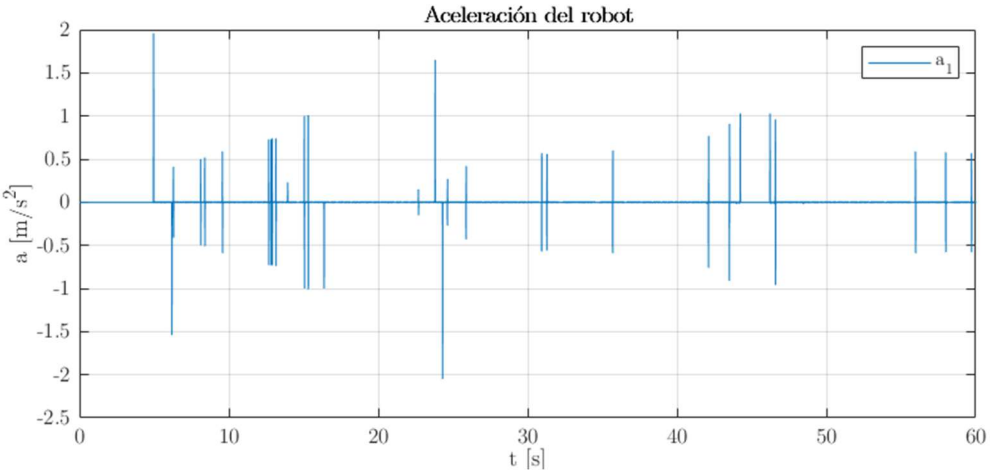


Figura 5.6. Aceleración del robot en la pata 1

Aplicando este modelo de filtro de Kalman a la velocidad, logramos adecuar la señal, y suavizar el ruido de la velocidad medida como observamos en la figura 5.7:

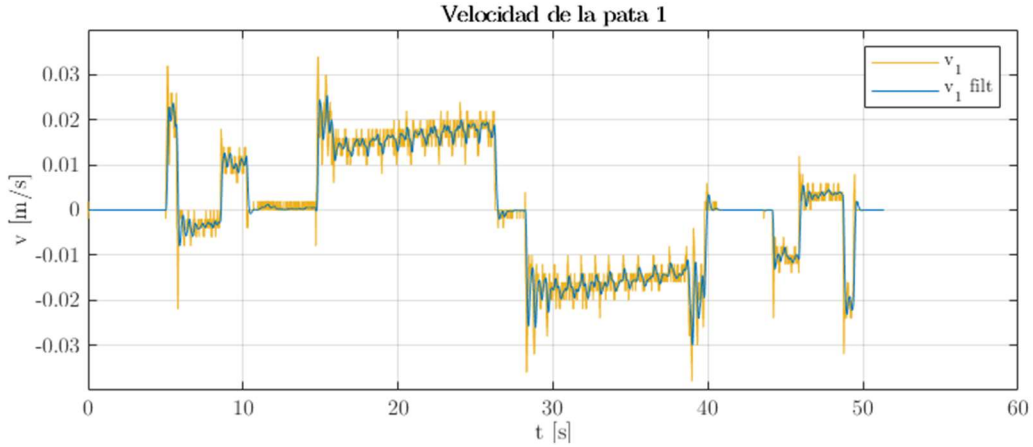


Figura 5.7: Filtrado de la velocidad en la pata 1 mediante el filtro de Kalman

Tras adecuar correctamente las señales de entrada y salida, y tras el ajuste de los parámetros del modelo, el algoritmo funciono perfectamente en simulación y en el robot real.

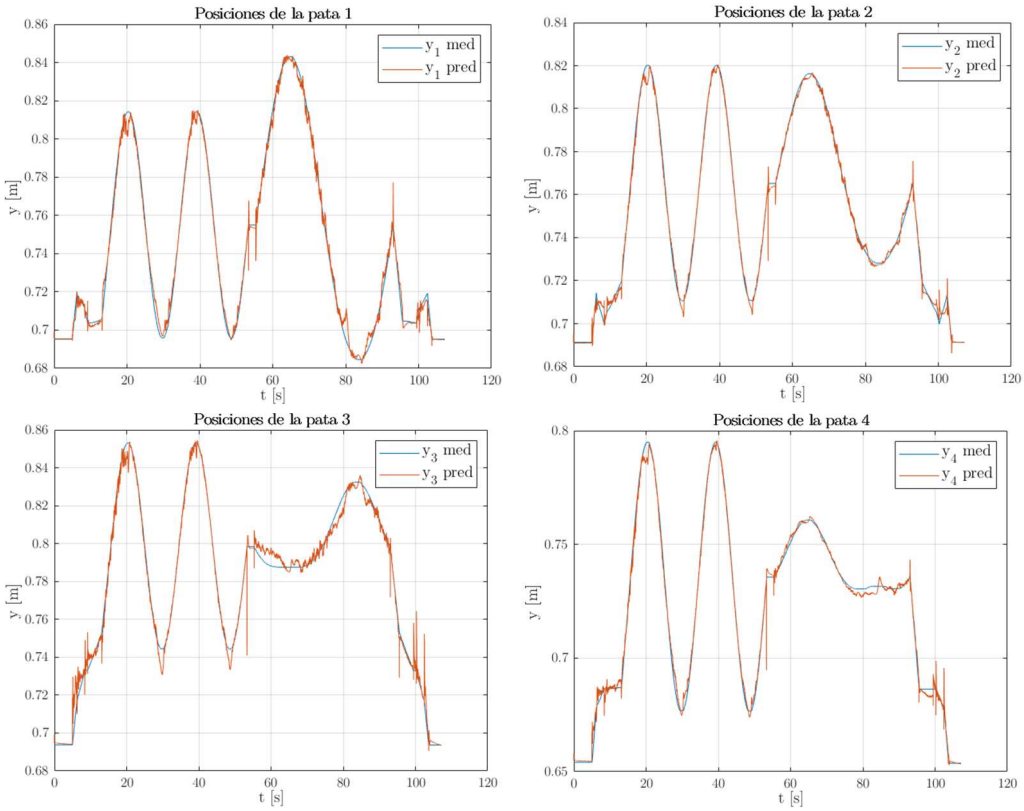


Figura 5.8: Comparativa entre la posición medida y la predicción realizada por el LWPR

En la figura 5.8 se observa el resultado experimental sobre una de las trayectorias, la predicción se ajusta correctamente a la posición real medida por el sistema de captura en general, si bien presenta cierto 'ruido', si lo comparamos con la señal bastante más limpia de la posición medida.

Podemos aplicar un filtrado a la señal de salida mediante el mismo filtro de Kalman visto anteriormente, para reducir este ruido y tener una predicción mucho más limpia.

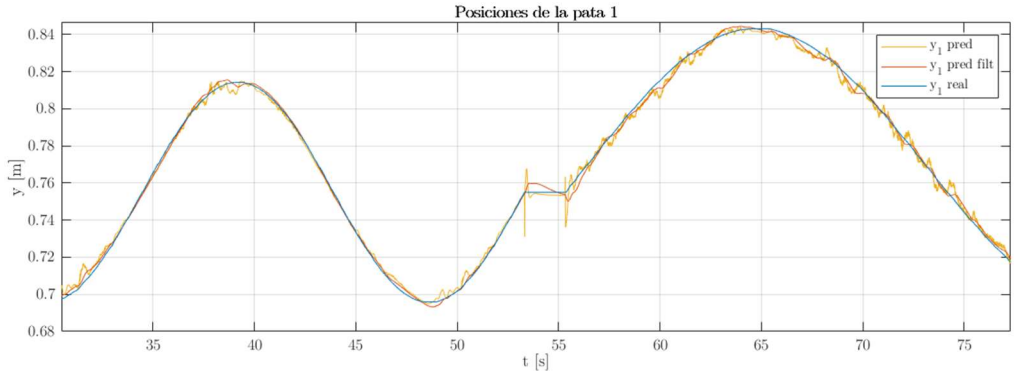


Figura 5.9: Comparativa entre la posición medida, la predicción, y la predicción filtrada en la pata 1

Como se puede comprobar en la figura 5.9, aplicar el filtrado a la salida logra corregir el ruido de la señal, y mejorar el comportamiento del módulo de predicción.

Para evaluar la eficacia de la predicción usaremos como métrica el error cuadrático medio RMSE, que se calcula mediante la siguiente expresión:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \tag{5.2}$$

Donde $\hat{y}_i$ será el valor de la posición predicho, y_i la posición real medida por el robot y n el número de muestras. Quedando los siguientes resultados para cada pata:

	Pata 1	**Pata 2**	**Pata 3**	**Pata 4**
RMSE (m)	2.991×10^{-3}	2.283×10^{-3}	3.115×10^{-3}	2.245×10^{-3}

Tabla 5.1: Valores de error RMSE de la predicción de la posición

El error de predicción en promedio se sitúa en todas las patas entre 2 y 3 mm respecto a la posición real, un valor relativamente bajo, que nos permitiría conocer la posición del robot siempre y cuando no necesitáramos una precisión milimétrica.

Como sucedía con el módulo de control, también tenemos que el número de RF crece durante la primera ejecución en una trayectoria, pero se mantiene en las siguientes pasadas.

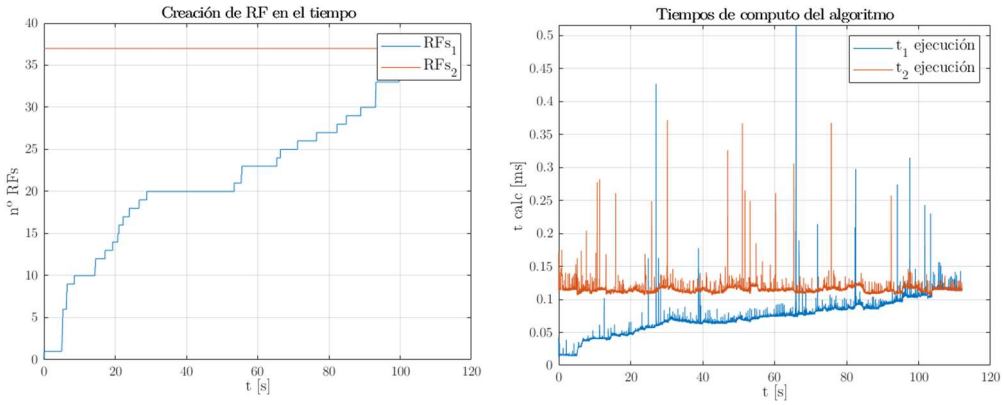


Tabla 5.2: relación entre el tiempo de cálculo y el número de RF

Y sigue existiendo relación directa entre el tiempo de cálculo y el aumento del número de modelos locales. En este caso, fue muy importante ajustar correctamente el valor de D inicial durante la fase experimental, porque con valores demasiado altos, el tiempo de cálculo se disparaba demasiado rápido, sin ofrecer mejoras significativas en el desempeño de la predicción.

5.3 Experimentos con LWPR + ILC

Debido a la naturaleza iterativa de este algoritmo, todos los resultados que se presentan a continuación se realizaron sobre una misma trayectoria, en concreto se realizaron diez iteraciones, pues a partir de ese número ya no es apreciaban mejoras significativas en su comportamiento.

Podemos observar el comportamiento del control ILC sobre cada uno de los actuadores en la siguiente figura:

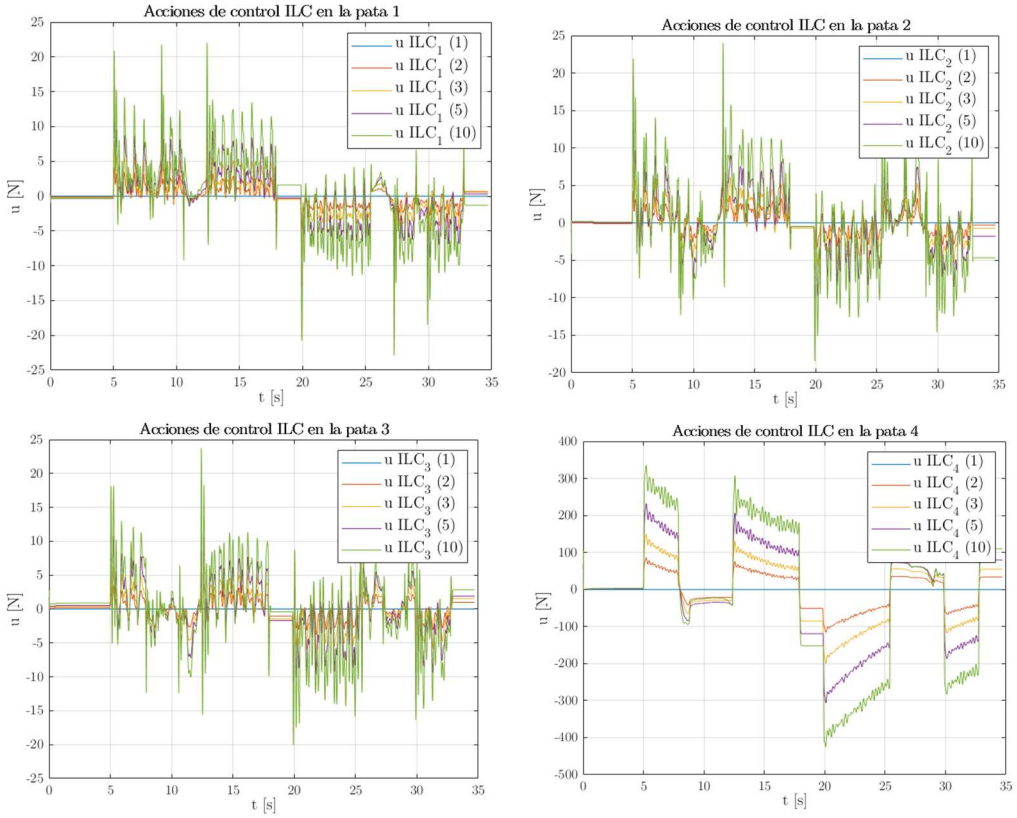


Figura 5.10: Evolución de la acción de control ILC a lo largo de las iteraciones

Se aprecia el aumento del peso de la acción con el paso de las iteraciones, aunque cada vez en menor grado debido a la disminución del error, y el *learning rate* decreciente. En concreto se nota especialmente la actuación sobre la cuarta pata, pues esta es la que soporta gran parte del esfuerzo de actuación debido a su posición central en la plataforma.

El objetivo principal de esta implementación era la reducción de los errores de posición, para evaluarlo se ha utilizado el error medio cuadrático (5.1), y la mejora de este respecto a su valor de referencia (la primera iteración, en la que la actuación del ILC es nula, pues no tiene ningún valor almacenado en la memoria).

	Pata 1		**Pata 2**		**Pata 3**		**Pata 4**	
i	**RMSE (mm)**	**%**	**RMSE (mm)**	**%**	**RMSE (mm)**	**%**	**RMSE (mm)**	**%**
1	0,2359	0,0%	0,244	0,0%	0,2562	0,0%	0,9331	0,0%
2	0,2099	11,0%	0,222	9,2%	0,2272	11,3%	0,7915	15,2%
3	0,1985	15,8%	0,200	18,2%	0,2069	19,2%	0,6863	26,4%
5	0,1640	30,5%	0,164	32,9%	0,1815	29,2%	0,5316	43,0%
10	0,1608	31,8%	0,161	34,0%	0,1799	29,8%	0,3568	61,8%

Tabla 5.3: Errores RMSE en cada pata para las iteraciones del ILC

Como se aprecia en la tabla 5.3, los errores se ven reducidos iteración a iteración, especialmente en la pata 4, siendo también esta en la que de partida se registraba un error más significativo. De hecho, en esa junta se observa la reducción de los errores claramente con el paso de las iteraciones.

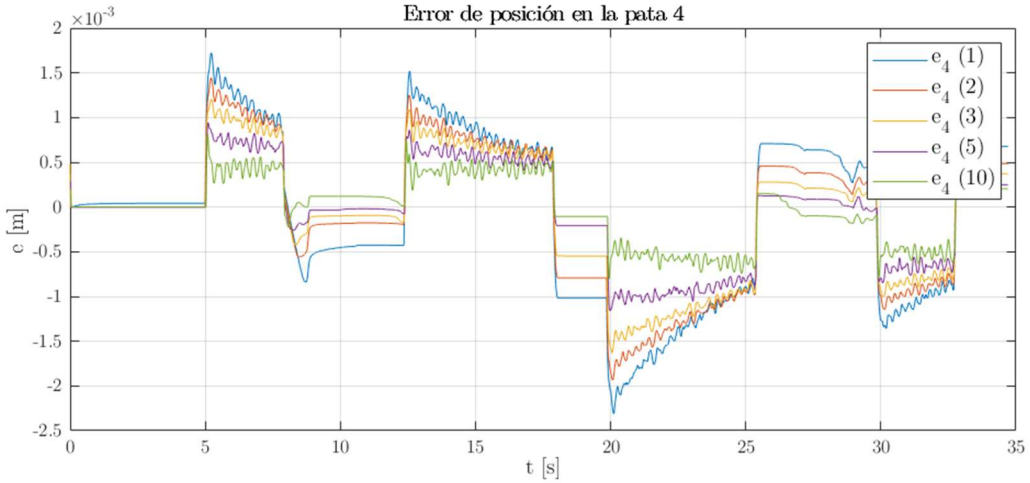


Figura 5.11: Evolución del error de posición a lo largo del tiempo y las iteraciones en la junta 4

En las secciones de la trayectoria donde había un mayor error inicial, el uso del ILC logra reducirlos a menos de la mitad en solo diez iteraciones de esta.

Queda por tanto verificado la ventaja que ofrece el uso del algoritmo ILC en trayectorias repetitivas, logrando una reducción de errores especialmente donde estos eran más abultados. Si bien, donde los errores ya eran bastante bajos, su efecto no es tan contundente, pues ya se parte de una situación muy favorable.

6 Conclusiones y trabajos futuros

6.1 Conclusiones

A lo largo del desarrollo del trabajo previo, la etapa experimental y la confección de la memoria se han ido cumpliendo satisfactoriamente todos los objetivos propuestos inicialmente, y que se recogen en la introducción de este documento.

Inicialmente, la realización de una completa y rigurosa revisión bibliográfica del estado del arte en materia de regresiones locales y otras técnicas de inteligencia artificial, y su aplicación al campo particular de la robótica, permitió conocer cuál debía ser el enfoque más adecuado para cada uno de los algoritmos, y técnicas de control y aprendizaje que se han utilizado en este trabajo.

El entorno de simulación ha respondido satisfactoriamente, permitiendo el desarrollo y prueba de los algoritmos en la primera etapa del trabajo experimental, que posteriormente serían puestos en marcha sobre el propio robot real, gracias a las modificaciones aplicadas en base a la información recogida en simulación.

Posteriormente, se ha logrado implementar un controlador basado en el algoritmo LWPR para ejercer el control de trayectorias satisfactoriamente. Este algoritmo ha resultado ser una técnica idónea para esa aplicación, permitiendo entrenar los modelos, y en pocas iteraciones poder establecer un control suficientemente robusto. Además, no solo se han conseguido los objetivos de control en escenarios pre-entrenados, sino que el algoritmo es capaz de adaptarse en tiempo real a nuevos espacios de trabajo, y generalizar adecuadamente gracias a las virtudes del entrenamiento local. Prueba del éxito de este algoritmo, es que ha sido capaz de batir al control tradicional mediante PID, reduciendo significativamente los errores de posición en todas las medidas.

Adicionalmente, la potencia del LWPR para predecir sistemas complejos no lineales, ha permitido obtener un modelo de predicción futura del robot

que funciona correctamente. Este permite evitar tener que resolver el problema dinámico directo de la robótica con la suficiente precisión como para ser una opción a tener en cuenta para modelar robots, ante la dificultad técnica de desarrollar su modelo matemático completo. La implementación del algoritmo no requiere conocimientos específicos sobre el robot, y su modelo mecánico, tan solo es necesario realizar una definición adecuada de las señales de entrada, y la salida a predecir, y el ajuste de algunos hiperparámetros experimentalmente, lo que permite centrar los esfuerzos, recursos y tiempo en otras áreas de interés. Este modelo, además bebe directamente de las señales ofrecidas por el robot real, optimizándose a lo largo del tiempo, y no presentando un error sistemático debido a fallos de modelado, simplificaciones, incertidumbre, etc.

Finalmente, la incorporación de la técnica interactiva ILC al controlador LWPR permite optimizar su comportamiento, y seguir reduciendo los errores producidos aún más, abriendo la puerta a las posibilidades que ofrecen la diversidad de técnicas de inteligencia artificial disponibles, y como podrían combinarse para obtener controladores mixtos más refinados.

6.2 Trabajo futuro

Con el desarrollo de este proyecto, y a la vista de sus resultados, queda claro que el futuro de las técnicas de aprendizaje automático en el campo de la robótica son prometedoras, y aún queda mucho camino por recorrer.

El desarrollo del control LWPR (o el combinado con el ILC) permiten realizar un control del robot paralelo más robusto y preciso para el seguimiento de trayectorias. Los modelos generados durante el desarrollo de este proyecto pueden utilizarse para controlar el robot en futuros proyectos, y gracias a su aprendizaje en línea seguirán optimizándose y adaptándose a nuevos espacios de trabajo.

Además, el éxito en la implementación del algoritmo en el robot paralelo invita a en un futuro seguir implementándolo en otros robots de esta u otra

tipología, como robots serie antropomórficos, para verificar su buen desempeño en todo tipo de escenarios, y ser una alternativa a futuro de otros controladores tradicionales.

Por su parte, los algoritmos iterativos como el ILC presentado en este proyecto, resultan otra alternativa interesante para optimizar el control en tareas repetitivas, no solo en combinación con el LWPR, sino en combinación con otras tipologías de controladores, o acompañados únicamente a los PIDs.

Finalmente, la aplicación de la inteligencia artificial a la robótica no tiene por qué quedar constreñida a las regresiones locales y algoritmos iterativos aquí introducidos, existen un sinfín de algoritmos y paradigmas distintas desarrollados en los últimos tiempos, que pueden implementarse en distintas aplicaciones y presentar mejoras interesantes.

Bibliografía

[1] B. Copeland, «Encyclopedia Britannica,» 11 Agosto 2020. [En línea]. Available: https://www.britannica.com/technology/artificial-intelligence.

[2] S. Mahadevan, *Machine Learning for Robots: A Comparison of Different Paradigms,* Tampa, Florida.

[3] O. S. Freek Stulp, *Many regression algorithms, one unified model - A review,* Elsevier, 2015, pp. 60-79.

[4] Na8, «Aprende Machine Learning,» 12 Diciembre 2017. [En línea]. Available: https://www.aprendemachinelearning.com/que-es-overfitting-y-underfitting-y-como-solucionarlo/.

[5] A. W. M. S. S. Christopher G. Atkeson, *Locally Weighted Learning,* Kluwer Academic Publishers, 1997.

[6] A. W. M. S. S. Christopher G. Atkeson, *Locally Weighted Learning for Control,* Atlanta, 1996.

[7] S. S. Christopher G. Atkeson, *Receptive Field Weighted Regression,* Atlanta, 1997.

[8] S. s. Sethu Vijayakumar, *Locally Weighted Projection Regression An On Algorithm for Incremental Real Time Learning in High Dimensional Space,* Los Ángeles, 2000.

[9] J. M. R. M. N. M. F. G.-C. Anna Derkacheva, *Data Reduction Using Statistical and RegressionApproaches for Ice Velocity Derived by Landsat-8,Sentinel-1 and Sentinel-2,* Grenoble, 2020.

[10] S. Vijayakumar, *LWPR: A scalable method for incremental online learning in high dimensions,* Los Ángeles, 2005.

[11] S. K. S. M. F. Arimoto, «Bettering Operation of Robots by Learning,» *Journal of Robotic Systems,* vol. 1, pp. 123-140, 1984.

[12] M. T. A. G. A. Douglas A. Bristow, «A Survey of Iterative Learning Control,» *IEEE Control Systems Magazine,* vol. Junio, pp. 96-114, 2006.

[13] H. Peter, *Robot,* 2021.

[14] I. Asimov, «Runaround,» *Astounding Science-Fiction,* Marzo 1942.

[15] «Global Population Of Industrial Robots At Record High, Says IFR Report,» *Industry Europe,* 25 Septiembre 2020.

[16] H. D. TAghirad, *Parallel Robots: Mechanics and Control,* cRC Press, 2013.

[17] A. I. 2. Documentation, «abb.com,» [En línea]. Available: https://new.abb.com/products/robotics/industrial-robots/irb-2600/irb-2600-data.

[18] O. eCobra, «industrial.omrom.es,» [En línea]. Available: https://industrial.omron.es/es/products/ecobra.

[19] D. Stewart, «A Platform with Six Degrees of Freedom,» *Proceedings of the Institution of Mechanical Engineer,* vol. 180, nº 15, pp. 371-386, 1965.

[20] FANUC, «fanuc.eu,» [En línea]. Available: https://www.fanuc.eu/es/es/robots/página-filtro-robots/delta-robots/serie-m3/m-3ia-6a.

[21] V. Mata, *Robot 3UPE-RPU,* Valencia, 2018.

[22] «MEBIOMEC,» 2013. [En línea]. Available: https://mebiomec.ai2.upv.es.

[23] A. D. S. S. Sethu Vijayakumar, «LWPR Library Documentation,» 9 febrero 2012. [En línea]. Available: http://lwpr.sourceforge.net/index.html.

[24] «ROS.org,» [En línea]. Available: https://www.ros.org.

[25] B. J. G. Guennebaud, «eigen.tuxfamily.org,» [En línea]. Available: https://eigen.tuxfamily.org/index.php?title=Main_Page.

[26] S. V. Stefan Klanke, *A Library for Locally Weighted Projection Regression - Supplementary Documentation,* 2008.

[27] M. M. Y. P. Joel Maridor, *Kalman filter to measure position and speed of a linear actuator,* 2011.

Sección II. Presupuesto

i. Partidas del presupuesto

A continuación, se realizará el análisis económico del presente proyecto, de acuerdo a las recomendaciones del Centro de apoyo a la innovación, la investigación y la transferencia de tecnología (CTT) de la Universidad Politécnica de Valencia. Podemos dividir este presupuesto en dos partes bien diferenciadas:

En primer lugar, los gastos de personal, tienen en cuenta el coste de las horas de ingeniería invertidas en el proyecto. Como coste horario se tomará el recomendado por el CTT para ingenieros superiores contratados como personal eventual. La partida de presupuesto de gastos de personal se ha asignado atendiendo a las tareas en las que se ha dividido el trabajo, haciendo un total de 240 horas acorde a la asignación de los trabajos fin de master (12 ECTS, a 20 h/ECT).

Tarea	€/Hora	nº Horas	Coste(€)
Revisión bibliográfica	24,99	25	624,75
Implementación en Simulación	24,99	40	999,60
Implementación en c++	24,99	20	499,80
Fase experimental	24,99	80	1.999,20
Análisis de resultados y mejoras	24,99	30	749,70
Redacción de la memoria	24,99	45	1.124,55
Total		240	**5.997,60 €**

Tabla i.1: Gastos de personal

En segundo lugar, tenemos los distintos gastos materiales asociados al desarrollo de las actividades del proyecto. Se ha tenido en cuenta: el equipo informático personal (ordenador, monitor, ratón…) utilizado para el desarrollo de la mayor parte de las fases del proyecto, el coste de la licencia de software de simulación Matlab (el IDE utilizado para la compilación en c++ era de libre distribución) y otras licencias software como la del sistema operativo o las herramientas ofimáticas (Windows y Microsoft Office respectivamente). Además, del coste del robot, las cámaras de visión y el equipo informático utilizado para el control.

De acuerdo a las recomendaciones del CTT se ha tomado un periodo de amortización de 6 años para todo el equipo, por lo que se tendrá en cuenta la parte proporcional del gasto hecho en el tiempo de uso.

Partida	Precio (€)	Uso (meses)	Amort. (años)	Importe (€)
Equipo Informático personal	750,00	6	6	62,50
Licencia Matlab	3.350,00	6	6	279,17
Otras licencias software	328,00	6	6	27,33
Robot	13.055,33	4	6	725,30
Equipo informático control	950,00	4	6	52,78
Total				1.147,07

Tabla i.2: Gastos materiales

ii. Presupuesto total del proyecto

Finalmente se desarrolla el coste final del presupuesto de licitación teniendo en cuenta otros gastos indirectos, así como el IVA.

Categoría	Subtotal(€)	Importe(€)
Gastos materiales	1.147,07	
Gastos de personal	5.997,60	
Presupuesto de ejecución material (€)		7.144,67
Gastos generales (13%) (€)		928,81
Beneficio Industrial (6%) (€)		428,68
Presupuesto de ejecución por contrata (€)		8.502.16
IVA (21%) (€)		1.785,45
Presupuesto base de licitación (€)		10.287,62

Tabla ii.1: Presupuesto total

El presupuesto total del proyecto asciende a la cantidad de: "DIEZ MIL DOSCIENTOS OCHENTA Y SIETE EUROS Y SESENTA Y DOS CÉNTIMOS".