UNIVERSITAT POLITÈCNICA DE VALÈNCIA

# Diseño e implementación de un sistema de control de robots mediante la ingeniería del software basada en componentes. Aplicación a un robot paralelo de 3DOF

Tesis Doctoral del Programa de Doctorado en Automática, Robótica e Informática Industrial


Autor: **D. José Ignacio Cazalilla Morenas**

Dirigida por: **Dra. Dña. Marina Vallés Miquel**

**Dr. D. Ángel Valera Fernández**

Intituto Universitario de Automática e Informática Industrial (AI2)

Departamento de Ingeniería de Sistemas y Automática (ISA)

Universitat Politècnica de València, España


Mayo 2017


# Agradecimientos

Bueno, pues llegó el momento de dedicar unas líneas a todos aquellos que han hecho posible que todo esto esté pasando.

En primer lugar, me gustaría dar las gracias a Marina y Ángel por la confianza depositada en mí desde el primer día que nos conocimos. Por haberme permitido aprender tanto de vosotros. Por demostrarme que ser investigador y docente de éxito no está reñido con ser una excelente persona. Por darme tantos y tantos consejos a lo largo de la última década. Es un lujo y un placer recorrer todo este camino junto a vosotros. Sois una pareja envidiable en todos los sentidos. Os admiro.

A los profesores Vicente Mata (Dpto. Ingeniería Mecánica, UPV), Miguel Díaz (Dpto. Tecnología y Diseño, Univ. Los Andes) y Álvaro Page (Instituto de Biomecánica de Valencia, IBV) por cómo estuvísteis involucrados en este proyecto de investigación desde el minuto uno. A Pedro Albertos, porque sin su figura, nada de esto estaría pasando. Mi rotunda admiración a tal excelente carrera.

A los amigos del pueblo, del instituto y de la carrera, así como los compañeros del laboratorio. Por todo lo que me habéis aportado para crecer como investigador y persona. Os deseo lo mejor.

*To my second family, my Irish friends from Galway. Thanks a million for all the unforgettable moments we have spent together. Thanks for making me feel like home and for the immersion into your nice culture. Thanks for showing me what the craic is. Thanks for the evenings in the pub having a few sociable ones. George, Mike, Stephen, Ehsan, Bryan, Will, Huanquing, Tere, Germán and Talia, thanks for being there whenever I needed anything. Big big special thanks to Dr. Siogkas who never let me give up. I am pretty sure that without your help, guys, I would had never made it. I will always remember this period of my life with a huge smile.*

A Irene, mi *Doctora* favorita. Por ser la mejor compañera de viaje que nunca habría podido imaginar. Por saber como nadie la fórmula para sacarme siempre una sonrisa en esos días de dudas. Por todo tu apoyo, tus consejos, tu serenidad, tu tranquilidad. Por haber hecho de mí una mejor persona. Por ser mi mejor amiga, y porque este viaje juntos no ha hecho nada más que empezar. Muchísimas gracias por ser la culpable de toda esta travesía. Te quiero.

A toda mi familia, especialmente a mis padres y mi hermana, los cuáles han sido con los que he compartido mis alegrías y tristezas más de cerca. Muchas gracias por la educación que me habéis dado. Es un gran orgullo para mí el poder presumir de semejante familia. *Mamá, Papá, siempre habíais soñado con tener un «Doctor» en casa, ¿verdad? Pues esto cada día está más cerca, aunque nunca podré curaros un simple resfriado. Espero que no os importe, puesto que la Medicina nunca fue lo mío.*

A muchísimos más que no entran en esta página, pero que se seguro se darán por aludidos. A todas y todos, GRACIAS.

# Resumen

En la presente Tesis se emplean técnicas de la ingeniería del software basada en componentes, con el objetivo de desarrollar controladores avanzados para un robot paralelo de 3 grados de libertad, así como crear una novedosa aplicación partiendo de todo lo anterior. Inicialmente, se realiza un estado del arte tanto en el desarrollo de componentes software como en controladores avanzados para manipuladores paralelos. Posteriormente, se presenta la metodología utilizada para el desarrollo de controladores dinámicos, así como controladores más avanzados como los adaptativos o híbridos. Finalmente, se desarrolla una aplicación completa para la rehabilitación de miembros inferiores.

Mediante las anteriores metodologías propuestas, se han podido tratar problemas referentes a la implementación de controladores para sistemas robóticos, siendo las siguientes las principales aportaciones de la Tesis:

- **Desarrollo de controladores modulares**. Mediante un apropiado diseño de los controladores, se exhibe la posibilidad de implementarlos de forma modular, siguiendo una serie de pautas relacionadas con el desarrollo de software basado en componentes. Gracias a ello,

se demuestran todas las ventajas que ello conlleva (en un caso de uso real) tales como la reusabilidad, robustez, dinamismo y coste.

- **Diseño e implementación de controladores dinámicos avanzados**. Basándose en el desarrollo de controladores modulares, se han diseñado, implementado y comprobado experimentalmente diversos controladores en los que la dinámica del sistema afecta al bucle de control principal. Además, se ha implementado un novedoso control adaptativo, capaz de aproximar parámetros que afectan a la dinámica del sistema (los cuales son desconocidos o variantes en el tiempo) de forma dinámica.

- **Diseño e implementación de controladores híbridos**. Mediante la integración de un sensor de fuerza en el sistema robótico, se ha conseguido desarrollar un novedoso controlador híbrido fuerza/posición. Este controlador permite la modificación, en tiempo de ejecución, de la referencia de posición, mediante la inclusión del sensor de fuerza en el bucle de control.

- **Aplicación de las tecnologías desarrolladas para la creación de una aplicación novedosa completa**. A partir de los numerosos controladores desarrollados y la integración entre diferentes *frameworks*, se presenta una aplicación capaz de realizar numerosos ejercicios de rehabilitación, optando a la posibilidad de realizar los mismos de una manera controlada de forma teleoperada.

# Resum

En la present Tesi s'empren tècniques de l'enginyeria del *software* basada en components, amb l'objectiu de desenvolupar controladors avançats per a un robot paral·lel de 3 graus de llibertat, així com la creació d'una nova aplicació partint de l'anterior. Inicialment, es realitza un estat de l'art tant en el desenvolupament de components programari com a controladors avançats per a manipuladors paral·lels. Posteriorment, es presenta la metodologia utilitzada per al desenvolupament de controladors dinàmics, així com controladors més avançats com els adaptatius o híbrids. Finalment, es desenvolupa una aplicació completa per a la rehabilitació de membres inferiors.

Mitjançant les anteriors metodologies proposades, s'han pogut tractar problemes referents a la implementació de controladors per a sistemes robòtics, sent les següent les principals aportacions de la Tesi:

• **Desenvolupament de controladors modulars**. Mitjançant un apropiat disseny dels controladors, s'exhibeix la possibilitat d'implementar-de forma modular, seguint una sèrie de pautes relacionades amb el desenvolupament de *software* basat en components. Gràcies a això, es demostren

tots els avantatges que això comporta (en un cas d'ús real) com ara la reusabilitat, robustesa, dinamisme i cost.

• **Disseny i implementació de controladors dinàmics avançats**. Basant-se en el desenvolupament de controladors modulars, s'han dissenyat, implementat i comprovat experimentalment, diversos controladors en què la dinàmica del sistema afecta el bucle de control principal. A més, s'ha implementat un nou control adaptatiu, capaç d'aproximar paràmetres que afecten la dinàmica del sistema (els quals són desconeguts o variants en el temps) de manera dinàmica.

• **Disseny i implementació de controladors híbrids**. Mitjançant la integració d'un sensor de força en el sistema robòtic, s'ha aconseguit desenvolupar un nou controlador híbrid força / posició. Aquest controlador permet la modificació, en temps d'execució, de la referència, mitjançant la inclusió del sensor de força en el bucle de control.

• **Aplicació de les tecnologies desenvolupades per a la creació d'una aplicació innovadora completa**. A partir dels nombrosos controladors desenvolupats i la integració entre diferents *frameworks*, es presenta una aplicació capaç de realitzar nombrosos exercicis de rehabilitació, optant a la possibilitat de realitzar els mateixos d'una manera controlada de forma teleoperada.

# Abstract

In this thesis, using techniques related to component-based software engineering, a variety of advanced controllers for a parallel robot of 3 degrees of freedom have been developed in a modular way. A novel and a complete application based on all of the above has been implemented as well. Initially, a state of the art is developed both in the software component development and advanced control for parallel manipulators. Subsequently, a methodology for the development of dynamic controllers is presented, as well as more advanced controllers such as adaptive or hybrid. Finally, a complete application for the rehabilitation of lower limbs is developed.

Through the previous proposed methodologies, it has been able to discuss problems regarding the implementation of controllers for robotic systems, being the main contributions of the Thesis:

• **Development of modular controllers.** Through an appropriate design of the controllers, the possibility of implementing them in a modular way, following a series of guidelines related to the development of software based on components, is exhibited. Thanks to this, all the advantages

that this entails (in real use case) such as reusability, robustness, dynamism and cost are demonstrated.

• **Design and implementation of advanced dynamic controllers.** Based on the development of modular controllers, several controllers have been designed, implemented and verified experimentally in which the dynamics of the system affects the main control loop. In addition, a novel adaptive control has been implemented, able to approximate parameters that affect the dynamics of the system (which are unknown or variants in time) dynamically.

• **Design and implementation of hybrid controllers.** By integrating a force sensor into the robotic system, a new hybrid force / position controller has been developed. This controller allows the modification, at run time, of the position reference, by including the force sensor in the control loop.

• **Application of the technologies developed for the creation of a complete and new application.** From the numerous controllers developed and the integration between different frameworks, an application to perform numerous rehabilitation exercises is presented. Also, by means of the frameworks integration, the application can be done in remotely in a controlled way.

# Índice general

# Índice de figuras

# Índice de tablas

# Parte I

## Introducción

# Capítulo 1

## Introducción

COMO podemos apreciar, cada vez más, los robots están empezando a ocupar espacios en nuestra sociedad en los que, unas décadas atrás, nunca nos lo podríamos haber imaginado. En la actualidad, en el campo doméstico, no nos extrañamos al contar con la presencia de un robot limpiador, así como determinados sensores capaces de advertir cuando encender o apagar de la forma más eficientemente posible las luces de una casa, por ejemplo. Si estos cambios han sido notables en este ámbito, la excepcional irrupción de la robótica en la industria ha jugado (sigue jugando y jugará) un papel determinante en todos los aspectos de nuestra sociedad, en los que los tiempos de entrega, la calidad y, sobretodo, el precio final del producto son aspectos críticos para éxito o el fracaso.

Debido a la voraz competencia que existe en la industria, con los continuos cambios en elementos hardware y software, no es suficiente con tener únicamente optimizada la tarea en sí del robot, sino también un

nuevo enfoque a la programación de los mismos. Pese a que nos podemos encontrar con robots de infinitos tipos, siempre hay tres partes muy diferenciadas que son comunes a todos:

1. Sensores
2. Sistema de control
3. Actuadores

Es por ello por lo que la idea del desarrollo de software basado en componentes cobra una gran importancia. Este concepto de componentes software (que se comentará en detalle en la siguiente sección) consiste, simplemente, en la creación de unidades básicas e independientes, con servicios perfectamente definidos y con una funcionalidad completa. El hecho de aplicar metodologías consistentes en la ingeniería del software basada en componentes va a permitir el poder reutilizar módulos previamente desarrollados, aumentando la calidad del producto y, a su vez, reduciendo el coste total. Más concretamente, en cuanto a los robots de configuración paralela, a los que se les exige la realización de tareas con gran precisión y velocidad, son necesarias estrategias avanzadas de control para cumplir este requisito.

Como se discutirá en los siguientes apartados, la descomposición de estos sistemas control complejos en pequeños módulos independientes entre sí, va a proporcionar unas ventajas determinantes, permitiendo así diseñar y desarrollar nuevos controladores más avanzados de una manera más eficiente.

## 1.1. Desarrollo de software basado en componentes

Cuando se habla sobre Ingeniería del Software, es muy complicado adivinar el momento de sus orígenes. El motivo es la definición en sí del concepto del software ya que, dependiendo de ello, nos podríamos remontar a finales del siglo XVIII [1], pasando por la Segunda Guerra Mundial y Alan Turing [2] o, como la gran mayoría de literatura señala, alrededor de los años 50 cuando el matemático Tom Kilburn almacenó en un dispositivo electrónico una pieza de código y fue ejecutada exitosamente [3]. No obstante, no fue sino hasta la primera conferencia de Ingeniería del Software en 1968 cuando se introdujo el concepto de componentes software [4].

Pese a que el concepto de desarrollo de software basado en componentes, *CBSD* (*Component-Based Software Development*) o *CBSE (Component-Based Software Engineering)* no es nada nuevo y hay un buen número de trabajos sobre esta temática, nunca se ha llegado a posicionar como la metodología más usada, pese a demostrarse algunos de los puntos fuertes (que posteriormente se abordarán) que se pueden conseguir, como son:

- Reutilización
- Coste (económico y desarrollo)
- Calidad
- Integración

Si bien ha habido un cierto crecimiento en el uso de esta metodología, éste no se ha visto plasmado en el campo de la robótica, siendo este el campo en el que nos vamos centrar y analizar a continuación.

En primer lugar, para poder comprender qué es el CBSD, hay que definir qué es un componente. No es una tarea sencilla puesto que, dependiendo de los autores, la definición varía. La definición más extendida es la formulada en [5], en el que se define un componente software como *una unidad funcional y básica de composición con interfaces y un contexto bien definidos.* Otros autores definen un componente como *una unidad que encapsula una funcionalidad, restringiendo el acceso a esa funcionalidad interfaces explícitamente definidas* [6] o, más orientado al Software, lo definen como *una caja negra que proporciona y requiere servicios a través de interfaces bien definidas* [7]. Como se puede ver, dependiendo de la temática de los trabajos, las definiciones que se pueden encontrar son muy variadas. Pese a que las mismas no son exactamente iguales, prácticamente todos los autores coinciden en las dos partes estructurales que debe presentar un componente software [7–9].

- **Puertos**. La principal función de un puerto es la de permitir la comunicación entre los distintos componentes. Tal y como se comenta en [7], *en software, un puerto es lo equivalente a un conector en hardware.* Únicamente mediante los mismos se pueden crear conexiones, algo elemental en la ingeniería del software basada en componentes. En cuanto al tipo de puertos y conexiones, estos serán comentados en siguientes apartados.

- **Interfaces:** Las interfaces son las partes visibles (públicas) de los componentes y, mediante estas, los componentes pueden interactuar entre ellos. Un buen diseño de la interfaz es crítico a la hora de explotar al máximo el concepto de modularidad. Como muy bien dicen los autores en [8], *diseñar componentes reusables consiste en encontrar un equilibrio entre ser demasiado específico (menos reusable) y demasiado genérico (menos calidad).*

Del mismo modo que con la definición de componente, no se puede encontrar en la literatura una única definición de «Desarrollo o Ingeniería del Software basado en Componentes (CBSD o CBSE)». Una de las definiciones más extendidas es la que se puede encontrar en [8], en la que se expone que el principio del *CBSD consiste en la creación de sistemas a partir de componentes reusables, manteniendo separado el desarrollo del componente en sí, con el desarrollo del sistema*. Otros autores lo definen como *el desarrollo de sistemas distribuidos a partir componentes software reusables, independientes, y debidamente testeados* [10]. Como vemos, no hay un consenso claro a esta definición, por lo que algunos autores [11] se atreven a decir el *CBSE es una parte coherente de la ingeniería, pero que aún no está muy claro qué es exactamente.*

Habiendo explicado los conceptos básicos, en los siguientes apartados nos centraremos en el desarrollo de software basado en componentes en el campo de la robótica, así como sus principales características y costes. Finalmente, se comentarán los principales *middlewares* orientados a componentes más influyentes del momento.

### 1.1.1. El desarrollo de software basado en componentes (CBSD) orientado a la robótica

Los requisitos software para las aplicaciones robóticas son exactamente iguales a aquellos requisitos de sistemas software en otros ámbitos como puede ser la empresa automovilística, automática o, incluso, telecomunicaciones. En todos y cada uno de estos ámbitos, el objetivo es usar la ingeniería del software para desarrollar un nuevo producto con una mayor calidad, pero con un menor coste (en todos los sentidos). Una opción para poder conseguir esto es aprovechar la soluciones existentes para crear nuevos sistemas, siendo ello la esencia de la ingeniería del software basado en componentes. Algunos de los beneficios que podemos encontrar en sistemas basados en componentes son [12]:

**Modularidad:** Realmente, el hecho de usar componentes implica usar los principios básicos de la modularidad. Una de las principales ventajas que se pueden encontrar gracias a la modularidad es la flexibilidad que proporciona. Al tratarse de entidades independientes, estas son candidatas perfectas en un sistema distribuido, teniendo un coste de integración muy pequeño. Pese a que el concepto de modularidad no es algo novedoso y las mejoras son evidentes, la realidad es que la gran mayoría de soluciones software no utilizan este principio.

**Reusabilidad:** Esta es otra de las motivaciones del CBSD ya que, reusando componentes, implicaría reducir costes pero sin dejar de

lado la calidad del producto. La procedencia de estos componentes que pueden ser reusados no tienen que venir necesariamente de dentro de nuestra empresa o laboratorio, sino que también pueden venir de organizaciones externas. Es muy importante recalcar que reusar no es «copiar y pegar» alguna función o clase [13]. Idealmente, cuando una nueva versión de un componente es lanzada al mercado, el proceso de integración no sería más que reemplazar el nuevo por el viejo componente. Además, el usuario final no debe conocer necesariamente el código fuente del nuevo componente, sino las funcionalidades que aporta y/o posibles cambios incompatibles que implique cambios en componentes que dependan de él. Estas características se pueden ver en la interfaz pública del nuevo componente.

Pese a que los principios del CBSD son bastante apropiados para la robótica, no ha tenido una gran acogida. Sin embargo, en [12] se describe una serie de características que son comunes a los sistemas robóticos, los cuales son idóneas para aplicar el concepto de CBSE:

**Complejidad:** Un aplicación robótica, por simple que pueda parecer, es algo realmente complejo. Como mínimo, el sistema debe ser capaz de interactuar mediante sensores, actuadores y algoritmo de control. Esta comunicación (que puede ser síncrona, asíncrona e, incluso, en hilos de ejecución distintos) es una tarea compleja desde el punto de vista de la ingeniería del software.

**Flexibilidad:** Un sistema robótico tiene que ser capaz de ser flexible,

por ejemplo, en cuanto a la realización de experimentos. Los investigadores esperan ser capaces de generar resultados, centrándose únicamente en una parte del algoritmo (por ejemplo, visión), dejando intacto el resto del sistema. Del mismo modo, también se espera que el sistema sea lo suficientemente flexible como para comparar algún aspecto del mismo, reemplazándolo por otro componente compatible (por ejemplo, en control, comprobar la estabilidad del sistema usando un controlador proporcional (P), proporcional-derivativo (PD) o proporcional-integral-derivativo (PID)).

**Distribución:** Los sistemas robóticos son, generalmente, distribuidos, y cada vez es más común que un único robot esté compuesto por más de un procesador. Es muy importante que el código no cambie cuando en caso de integrar distintas funcionalidades de un procesador a otro.

**Heterogeneidad:** Los sistemas robóticos distribuidos suelen requerir hardware de diferentes fabricantes, así como más de un único sistema operativo.

Por otro lado, el desarrollo de componentes no deja de ser ingeniería y que, como se comenta en [14], los componentes software no se han desarrollado *de forma casual* de modo que se pueden utilizar en un esquema modular, sino que ha habido un estudio previo del diseño para poder conseguir que se utilicen de esta manera. Es por ello que en [15] se pueden observar cuatro conceptos clave, «las cuatro C's» que, años más tarde pasaron a ser «las cinco C's», en un artículo en el que se exponen estos con-

ceptos, con el objetivo de poder desarrollar componentes reusables en el campo de la robótica [16].

**Computación:** La computación consiste en el procesado de datos por parte del algoritmo que se necesita en la aplicación [17]. Generalmente, esto consiste típicamente en la lectura y escritura de datos por parte del componentes, así como la sincronización de las actividades computacionales entre componentes.

**Configuración:** Puesto que el objetivo de la CBSE es la creación de nuevas aplicaciones reusando componentes previamente desarrollados, además de las conexiones entre ellos, en ocasiones es necesario ajustar ciertos parámetros a través de las interfaces que ofrece el componente (configurando los mismos). Parámetros configurables son tales como ganancias, política de comunicaciones, direcciones de memoria de actuadores y sensores, etc.

**Comunicación:** La comunicación consiste en el intercambio de información entre los componentes, así como los mecanismos que intervienen en ella. Aspectos como ancho de banda, latencia o prioridad son determinantes en el aspecto de la comunicación.

**Coordinación:** En cuanto a la coordinación, esto consiste en cómo los componentes interaccionan entre ellos en un mismo sistema. Concretamente, *la coordinación provee el comportamiento discreto de un componente o sistema.*

**Composición:** Este concepto (que fue añadido en [16], extendiendo las

4 C's [15]) consiste en la relación o acoplamiento, que siempre es necesario, de cada uno de los componentes independientes.

Sin embargo, no son todo ventajas en la ingeniería del software basada en componentes, y se puede encontrar literatura en la que exponen, también, puntos en contra de esta metodología. En [12] indican que los componentes serán útiles siempre y cuando estos hayan sido desarrollados usando un mismo estándar. Por otro lado, hay otros autores que son incluso más críticos, afirmando que una vez se comienza a desarrollar software para un *framework* (o *middleware*) concreto, siempre existirá una fuerte dependencia, pese a que la esencia (basado en componentes) sea la misma [18]. En [7] también se afirma que en muchas ocasiones, al no encontrar un *framework* que satisfaga completamente todas la necesidades, la opción que se adopta es *implementar otro nuevo framework desde cero, tendiendo a reinventar la rueda*. Evidentemente, esta opción no es la más adecuada, por lo que el mismo autor propone una solución muy interesante (la cual se comentará con más detalle en las siguientes secciones), que consiste en fomentar la comunicación entre distintos *middlewares* que sigan un mismo concepto basado en la CBSE [7].

### 1.1.2. El coste del CBSD

Desde la «crisis del software» [19], hace más de medio siglo, se ha experimentado un cambio significativo en cuanto al coste de desarrollo del software (SW), respecto al coste de desarrollo de hardware (HW). Mientras que inicialmente el coste del HW era el factor determinante en el precio

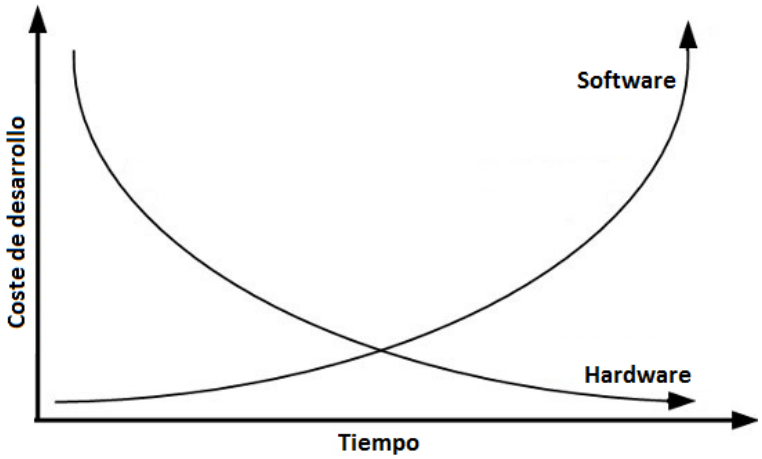



**Figura 1.1: Coste SW vs HW**

final del producto (siendo totalmente insignificante el coste del desarrollo de SW), en la actualidad esto ha cambiado completamente. Más específicamente, con el paso del tiempo, la tendencia ha consistido en la reducción del coste del HW e incrementar el del desarrollo del SW, llegando a invertirse la situación completamente ( Figura 1.1).

Es por ello por lo que en nuestros días, el software ha llegado a ser el componente con el coste más elevado en un sistema industrial, y esto es debido a que éste tiene que ser desarrollado de forma manual por ingenieros (y no máquinas). Es por ello por lo que hay diversos estudios en los que se centran en la estimación del coste del software, basándose en diferentes heurísticas [20, 21].

Basado en lo anterior, se puede deducir que la reducción en el coste del desarrollo de software es la clave del éxito en, por ejemplo, aplicaciones robóticas, las cuales tienen un ciclo de vida largo en el que ciertas funcionalidades deben ser actualizadas o reemplazadas varias veces. Pese a que uno de los conceptos elementales para conseguir este objetivo es «reutili-

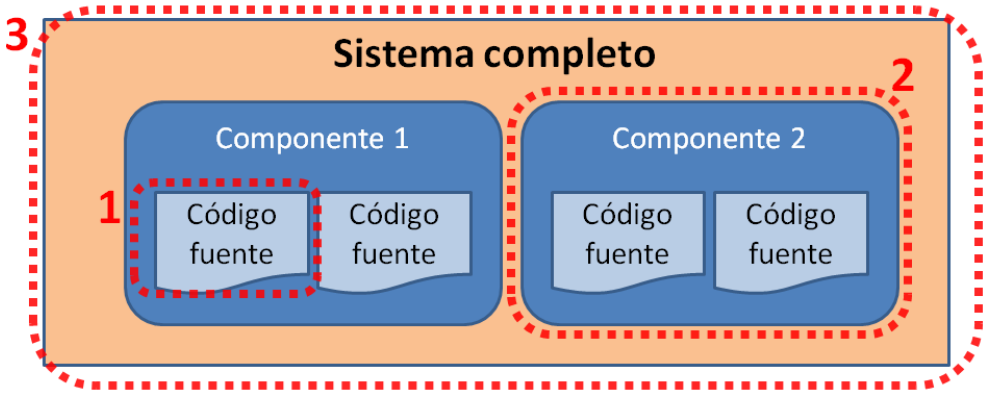



**Figura 1.2: Diferentes niveles de reutilización del software**

zar», ello implica que hay que emplear más esfuerzo (por lo tanto, dinero) en una fase inicial como puede ser la de diseño, obteniendo un beneficio a medio-largo plazo [22]. Este esfuerzo en la fase de diseño es del orden de 3 a 4 veces más [23] que en unas condiciones normales. En cuanto al concepto de reusar, pese a que se ha mencionado en la sección 1.1.1, cabe destacar que se pueden encontrar varios niveles de reutilización.

En la figura 1.2 se refleja los diferentes niveles de reutilización, tal y como se puede apreciar en [22]. En primer lugar, el nivel más bajo consistiría en la reutilización de alguna parte de código fuente, como podría ser una función o una clase (recuadro 1 en la figura 1.2), siendo esto poco recomendado en el CBSD. El siguiente nivel de reutilización estaría relacionado con el aprovechamiento de pequeños componentes, aunque con una funcionalidad definida y completa (recuadro 2 en la figura 1.2) como, por ejemplo, un controlador para un actuador. Finalmente, el nivel más alto de reutilización consistiría en el aprovechamiento de diferentes componentes conectados entre sí, formando un sistema con una funcionalidad completa (recuadro 3 en la figura 1.2), lo cual podría ser un sistema de

control para un robot. Sin embargo, el hecho de la reutilización pone de manifiesto una serie de retos (o dificultades) como:

**Evolución de los requisitos funcionales**. Pese a que, idealmente, en el desarrollo de componentes reusables debería realizarse en base a unos requisitos funcionales estables, en la realidad no es así. Realmente, los requisitos provenientes de cada uno de los productos son, ocasionalmente, iguales. Sin embargo, gran parte de ellos, son completamente nuevos y, puesto que el componentes debe cumplir con las necesidades de todos los productos, nuevas funcionalidades deben ser implementadas. Generalmente, el número de cambios en el componente es mayor en su fase temprana, en la que no tiene una funcionalidad generalizada y, nuevos requisitos implican añadir nueva funcionalidad. Sin embargo, en etapas posteriores en las que el componente es más genérico (debido a la funcionalidad previamente implementada), nuevos requisitos pueden ya estar cubiertos. Es en esta etapa en la que se aprecian las mayores ventajas en cuanto a la resuabilidad. Finalmente, en la última etapa de la vida del componente, el componente se queda desactualizado y, a la postre, obsoleto, debido a aspectos tales como nuevos estándares, nuevas plataformas o nuevos procedimientos. La figura 1.3 [22] demuestra todo el proceso comentado anteriormente.

**Migración entre diferentes arquitecturas.** Debido a las nuevas arquitecturas que van apareciendo, es lógico que un producto previamente desarrollado para una arquitectura, tenga que ser portado

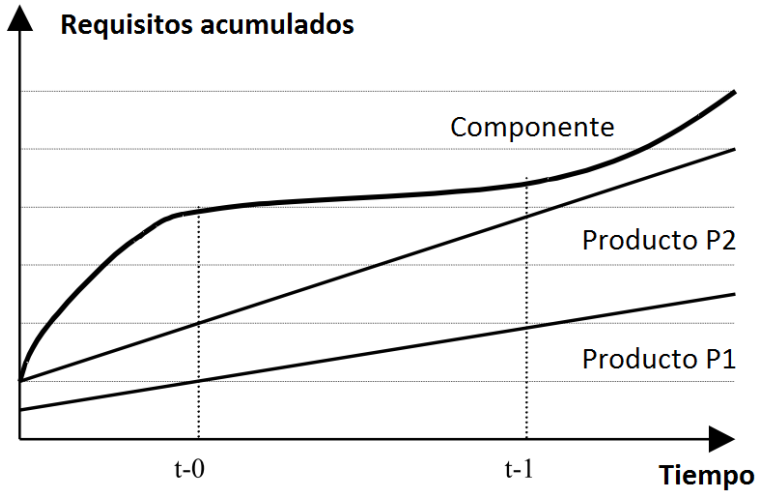



**Figura 1.3: Requisitos de un componente software**

a otra arquitectura. Este hecho implica que haya que mantener y desarrollar el mismo producto para cada una de las diferentes variantes. Puesto que un concepto clave para la reutilización es que los componentes no cambien en su capa más externa (interfaz), una opción es encapsular aspectos de bajo nivel (específicos a la plataforma) en la implementación. Esta idea funciona, en teoría, ya que lo único que se debería realizar es recompilar el componente para cada una de las plataformas, asumiendo que se usa un lenguaje de programación estándar como C/C++, Python o Java.

**Compatibilidad entre versiones**. Otro de los aspectos más importantes para maximizar la reutilización de los componentes está relacionado con la compatibilidad entre las diferentes versiones. Como también comentan otros autores, si dos componentes son compatibles, ellos deben ser *plug&play.* Generalmente, siempre (en la medida de lo posible) se debe asegurar la compatibilidad hacia atrás entre distintas versiones. Por ejemplo, un cambio incompatible podría

ser debido a que la declaración de una función pública ha cambiado de una versión a otra. Sin embargo, un cambio compatible podría ser la adición de una nueva función a ese componente. Mucho más complejo es cuando la plataforma en la que se va a realizar la integración aún no ha sido escogida y, pequeños cambios, por ejemplo, en el protocolo de comunicación hacen que se pierda la compatibilidad con versiones previas.

**Acoplamiento entre componentes**. Como se ha comentado anteriormente, los componentes deben ser entidades independientes con funcionalidad completa. Pese a las numerosas ventajas que ello aporta, también introduce una serie dificultades. Uno de los problemas está relacionado con el testeo del componente, que suele ser bastante complicado puesto que se necesita un entorno con todas las dependencias del componente en cuestión (en ocasiones, ese conjunto de dependencias pueden ser el producto completo). Otra de las dificultades es asegurar la consistencia entre las dependencias entre los componentes. Por ejemplo, si un componente A depende de un componente B y una librería D, mientras que el componente B depende de una nueva versión de la librería D (D'), las cuales no son compatibles, generaría una inconsistencia en cuanto a las las dependencias del componente A (figura 1.4).

**Mantenimiento.** En lo que se refiere al mantenimiento es realmente complicado cuando se tiene más de un cliente. Uno de los mayores retos es no crear un producto específico para cada uno de los

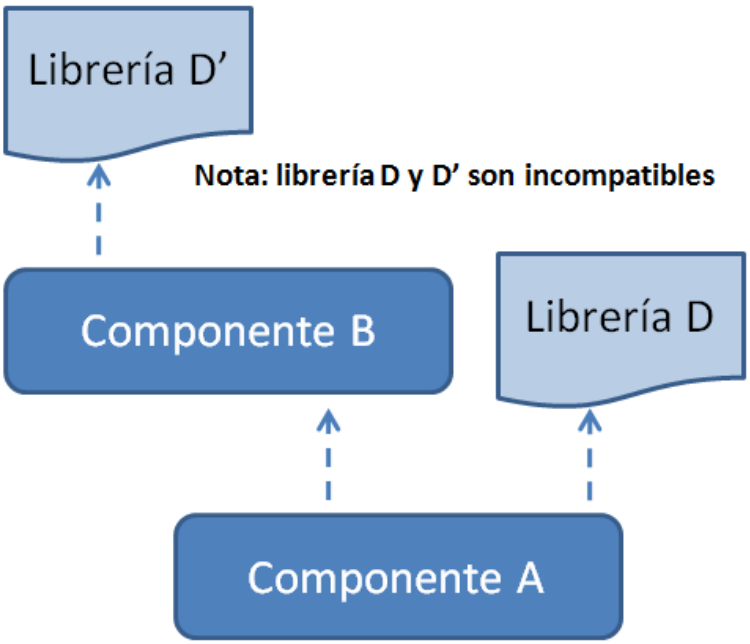



**Figura 1.4: Consistencia entre versiones**

clientes, añadiendo cada nueva funcionalidad que van requiriendo. En este caso, cuando los clientes proponen un cambio, éste debe ser analizado en su conjunto, puesto que puede afectar a más de un producto. Únicamente cuando se ha hecho un análisis sobre el impacto en otros productos, una nueva versión (compatible o incompatible, dependiendo de los cambios realizados) es lanzada al mercado.

Es por que ello que el coste del software (y no tanto el hardware) es el factor determinante por el cual existe una brecha tecnológica entre grandes y, pequeñas y medianas empresas (PYMES) en cuanto al campo de la robótica industrial se refiere. Son muchos los autores que afirman que, pese a ser estas PYMES las que más podrían beneficiarse en automatizar los procesos, no logran conseguirlo debido al corto ciclo de vida y el desarrollo de código específico para una aplicación (con el coste que ello

conlleva). Otro de los problemas es la inversión inicial que hay que hacer. Por ejemplo, el coste temporal de desarrollar una aplicación automática de soldado de un vehículo es, aproximadamente, 360 veces superior al coste de realizar la tarea en sí [24].

Debido a la falta de componentes reusables o genéricos en el ámbito de la robótica industrial, la única opción que tienen las PYMES para beneficiarse de la robótica (pudiendo asumir los costes) es usar la programación *online* [25]. La programación *online* de un robot consiste en especificar, manualmente, los movimientos que debe realizar el robot. Pese a que el concepto es bastante simple, tiene una serie de limitaciones, tales como:

- Sólo es válido para trabajar en tareas muy sencillas.
- La calidad va a ser dependiente de cómo se haya *entrenado* al robot, por lo que ésta va a ser dependiente del operario que haya generado la trayectoria.

Sin embargo, pese a las limitaciones que tiene la programación *online*, esta es la única opción que, hasta el momento, pueden permitirse las PYMES. La otra opción que existe es la programación *offline*, que consiste en desarrollar el software que va a ser ejecutado en el del robot, sin tener que estar en contacto directo con el mismo [26]. Para ello es necesario un buen modelado del robot, el cual permita realizar una simulación lo más precisa posible [27]. Obviamente, la calidad de esta tarea es ampliamente superior a la programación online (aunque el coste es, también, mayor).

Es cada vez más evidente que, para poder reducir los costes del producto final, y a su vez mantener (o incluso incrementar) la calidad, el concepto

de componentes o *piezas* intercambiables va cobrando cada vez más fuerza. En [28] van más allá y hablan de componentes *plug-and-play*, es decir, componentes los cuales no requieren de esfuerzo extra para su integración (evidentemente, estos componentes *plug-and-play* deben proporcionar una funcionalidad similar a los anteriores). Según estos los autores, si los robots son cada vez más modulares, el mantenimiento y soporte necesario será cada vez mayor. Sin embargo, el coste de desarrollar una nueva funcionalidad se produce una única vez, puesto que puede ser reusado por otro sistema robótico.

Por último, pese a que el desarrollo de controladores robóticos basado en componentes reduciría los costes, permitiendo el acceso a la automatización a ciertas empresas, hay otro concepto muy relevante que es la *adaptabilidad* [28]. Este control adaptativo (el cual se comentará en capítulos posteriores) permitiría la optimización de parámetros de forma dinámica, mejorando el rendimiento del sistema.

### 1.1.3. Middlewares basados en componentes

Otro de los elementos clave en la ingeniería del software basada en componentes son los *middlewares* o *frameworks*, concretamente los que son orientados a componentes. Un *middleware* no es más que una capa de abstracción que se sitúa entre el sistema operativo y la capa de aplicación, siendo el principal objetivo de éste, abstraer al desarrollador de tareas complejas tales como la gestión de información que se transmite por la red, gestión de recursos compartidos, notificación de eventos, gestión de

prioridades, etc. Pese a que el objetivo principal del CBSD es la reutilización de componentes, no existe una tendencia en lo que se refiere a reusar, también, los *frameworks*. El hecho es que, durante los últimos 20 años, un nuevo *middleware* orientado a componentes (específicamente en el campo de la robótica) se ha hecho popular más o menos cada dos años [7]. Algunos como Orocos [29] se han centrado más en aspectos relacionados con el tiempo real, mientras que otros como Player [30], Orca [31], GenoM [32], SmartSoft [33, 34], Miro [35], YARP [36], OpenRTM [37] o ROS [38], se han centrado en aspectos más generales.

Por otro lado, tampoco es de extrañar que, debido a la diversidad de aplicaciones robóticas que van apareciendo, se vayan requiriendo nuevos requisitos funcionales en los *middlewares* que no se habían previsto anteriormente. Es por ello que, cuando se crea un nuevo robot, los desarrolladores se enfrentan a la decisión de qué *framework* escoger, ya que que no existe una opción general y unificada para el campo de la robótica. Esta decisión es bastante complicada, ya que ello implica adoptar una serie de restricciones propias del *framework* durante todo el ciclo de vida del producto. En muchas ocasiones, la opción elegida suele ser la de «reinventar la rueda» [18], implementando un nuevo *framework* desde cero que, generalmente, no cumple con los requisitos esperados, se retrasa en el tiempo de implementación, no es maduro y, además, carece de la robustez que es necesaria en este tipo de aplicaciones.

Otros autores [39] prefieren construir una capa de abstracción por encima del *framework*, basándose en el principio del desarrollo de software dirigido por modelos (MDSD). Es el *Object Management Group* [40] el

organismo que está estandarizando la arquitectura dirigida a modelos, definiendo los 4 niveles de abstracción, siendo M3 el nivel más abstracto (o a un nivel muy alto) y M0 el nivel más específico (implica la propia implementación en un *framework* concreto, con librerías concretas). Usando este concepto de MDSD [41], es posible representar un sistema a partir de modelos, con un nivel de abstracción muy alto, pudiendo separar las unidades software de la plataforma de ejecución. Además, puesto que en esta fase aún no se ha escogido el *framework*, el diseñador no tiene que preocuparse siquiera del lenguaje de programación que desea utilizar, ya que eso vendría en una etapa posterior (de hecho, los autores en [39] hablan de generación automática de código). Pese a que el concepto es realmente bueno, es evidente que el *framework* a utilizar debe tener, al menos, unos mismos principios (de otro modo, el diseño del modelo sería una tarea francamente complicada), como puede ser orientado a servicios, orientado a objetos, etc.

Por otro lado, hay que destacar la influencia que, poco a poco, va teniendo el MDSD fuera del ámbito de la ingeniería del software pura, como puede ser en la automática industrial. En [42, 43] se presenta la metodología MeiA (*Methodology for industrial Automation Systems*), cuyo *framework* permite el desarrollo y modelado de sistemas industriales. La principal novedad que presentan los autores es la facilidad de uso (no es necesario saber UML, siendo ello uno de los principales hándicaps con los que se enfrenta la industria) y la encapsulación de herramientas bien conocidas en el ámbito de la automática como *GRAFCET* [44] o *GEMMA* [45]. Todo esto unido a una capa de abstracción sobre el usuario final, hace posible

que todas las fases, desde requisitos, diseño e integración, hasta la documentación del producto, quedan cubiertas con una única herramienta.

Por todo lo visto anteriormente, se hace patente que el MDSD cobra fuerza en todos los campos de la Ingeniería (no sólo en la del Software). En [46], los autores van un paso más adelante y proponen tanto la transformación entre modelos (M2M), como la generación automática de código también, o modelo a texto (M2T). Concretamente, para la transformación entre modelos, se parte de un modelado en UML [47], que se exporta automáticamente a formato XMI [48] que no es más que un fichero XML, pero con metadata integrado, y después a una notación específica que los autores desarrollan. En cuanto a la generación automática de código a partir del modelo específico que presentan los autores es, evidentemente, una tarea completamente dependiente del *framework* escogido. Es decir, deberá existir tantas transformaciones M2T como *frameworks* (y lenguajes de programación) para los cuales se quiera generar código. Más concretamente, en [49] se expone una herramienta, así como las reglas de transformación, la cuál es capaz de realizar la transformación de M2T para los middlewares ROS y Orocos. Pese a la gran utilidad que ello proporciona, en *middlewares* que no sean lo suficientemente maduros, quizá podría ser demasiado costoso el tener que mantener la generación M2T en cada una de las versiones oficiales del nuevo *framework*.

Con el objetivo de unificar pautas a seguir a la hora de desarrollar software basado en componentes para aplicaciones robóticas, nació BRICS [16]. Del mismo modo que se expone en [46], los autores proponen un uso extendido de la ingeniería dirigida por modelos (MDE) indicando que es la

única manera de dar soporte a sistemas complejos (como un sistema robótico), de forma que primero se modela la aplicación, después se analiza y verifica y, en última instancia, se genera el código fuente. En lo que se refiere al modelado, los autores presentan el metamodelo *component-port-connector (CPC)*, del cual se parte para la generación automática de código (esto demuestra que es completamente independiente al *framework*). Este metamodelo desarrollado por los autores es realmente simple y se conforma de las siguientes partes:

**Primitivas:** Un sistema está formado por primitivas básicas como son los componentes y conectores. Los componentes definen la funcionalidad y son configurados mediante *propiedades.* Por otro lado, los *puertos* permiten el acceso a una entidad externa comunicarse con el componente.

**Composición:** Las conexiones representan la interacción entre dos módulos (siempre mediante sus puertos)

**Restricciones:** No todas las reglas de composición son válidas (por ejemplo, un puerto de entrada, el cuál tiene conectado más de un puerto de salida), es por ello por lo que existen una serie de restricciones tales como:

- Un componente contiene cero o más puertos.
- Un puerto pertenece a un único componente.
- Una conexión se produce siempre entre dos puertos del mismo tipo.

Continuando con el modelado de aplicaciones siguiendo los principios expuestos en *BRICS* y con el objetivo de facilitar la tarea del modelado a los ingenieros, en [50] se presenta un entorno de desarrollo basado en Eclipse, BRIDE (*BRICS IDE*). Una de las principales características de *BRIDE* es que, además de permitir la generación de modelos (favoreciendo el principio de MDSD) basado en los principios de *BRICS*, permite la generación automática de código tanto para ROS [38] como Orocos [51], a partir de los modelos generados.

En cuanto a la generación automática de código, no deja de ser más que la creación del esqueleto o estructura básica del componente. Tal y como se aprecia en [8], si el modelo estuviese completamente vacío, únicamente sería necesario generar aquellas interfaces obligadas. Por el contrario, una de las principales ventajas del modelado es que se pueden añadir, además de las interfaces obligadas, interfaces funcionales o propiedades especiales en a un alto nivel de abstracción. Esto permite que en la transformación de modelo a texto (M2T), el código generado contenga todos los elementos. Evidentemente, este hecho sirve para evitar errores humanos, al menos, en la estructura básica (puesto que la funcionalidad debe ser desarrollada independientemente a la herramienta de modelado), siempre y cuando la transformación M2T sea lo suficientemente madura.

## 1.2. Manipuladores paralelos

Con el paso de los años, el campo de la robótica está creciendo a un ritmo frenético en el cual, si nos ponemos a pensar, podemos encontrar robots

en prácticamente todos los ámbitos de nuestra vida. Pese a que, en la actualidad, las grandes industrias hacen uso de la tecnología robótica, es un hecho innegable la repercusión que ésta ha tenido en otros distintos ámbitos como puede ser el hogar.

Centrándonos en el ámbito de la industria, se pueden discriminar dos categorías principales de robots, las cuales están basadas en la estructura de los mismos:

**Antropomórficos.** A este tipo de robots se le llaman antropomórficos debido a la similitud que tienen con el ser humano, especialmente en la parte del hombro, brazo y muñeca.

**Paralelos.** Su acepción paralela se refiere a que el extremo final del miembro del robot está conectado a su base por una serie de (por lo general tres o seis) miembros independientes, con la particularidad de que trabajan en paralelo. Por ello, *paralelo* se utiliza en el sentido topológico, y no geométrico, puesto que los miembros pueden trabajar juntos, y no tienen por qué estar alineados en forma de líneas paralelas.

En cuanto a los robots paralelos, del mismo modo que ocurre en el capítulo 1.1, no se puede precisar con exactitud cuándo apareció el primero como tal. En [52] se ordenan cronológicamente los diferentes manipuladores paralelos que han ido apareciendo, tales como la plataforma de Gwinnett [53], el robot paralelo para el pintado de Pollard [54], la plataforma de Gough para el test de neumáticos [55], o la más que conocida plataforma de Stewart [56] de 6 grados de libertad, pensada para aplicaciones

relacionadas con la simulación de vuelo. Si bien no se tiene la certeza de saber qué plataforma sigue más el principio de robot paralelo, es cierto que en la últimas décadas, el interés por los manipuladores paralelos ha ido creciendo exponencialmente (tal y como se demuestra con el número de publicaciones que tratan sobre la materia). Este interés viene dado por las principales ventajas que proporcionan, tales como:

- Alta rigidez y robustez, debido a la configuración cerrada que presenta su estructura.
- Movimientos de gran velocidad con gran precisión.
- Capacidad de manejar cargas pesadas, puesto que, generalmente, esta carga se reparte entre todos los actuadores y estos sueles estar fijos a la base del mismo.

Evidentemente, en los manipuladores paralelos no todo son ventajas, puesto que entre algunos puntos débiles está el limitado espacio de trabajo o la complejidad en la implementación de controladores, siendo uno de los mayores inconvenientes para este tipo de robots. Es por ello por lo que existen numerosos estudios que abordan el diseño y desarrollo de controladores para manipuladores paralelos, los cuales serán analizados en la sección 1.2.1.

### 1.2.1. Sistemas de control en los robots paralelos

Como se ha comentado anteriormente, los manipuladores paralelos tienen un número de ventajas respecto a los manipuladores en serie (antro-

pomórficos), siendo éstas las detonantes para que la comunidad científica se haya mostrado cada vez más interesada por este tipo de robots [57,58].

Pese a estas ventajas sobre los robots en serie, han sido numerosos los estudios en los que han señalado posibles problemas relacionados con la precisión y la rigidez. Concretamente, en [59] se presentan una serie de fenómenos, los cuales ponen en compromiso los conceptos comentados anteriormente:

- Puesto que el robot paralelo contiene numerosas articulaciones (tanto activas como pasivas), ello implica que la precisión en la elaboración del modelo cinemático contenga errores debido, por ejemplo, a la separación entre los componentes o a defectos de ensamblado [60].
- Debido a la compleja cinemática de los manipuladores paralelos, se tiende a simplificar el modelo, por lo que induce a obviar aspectos relevantes del robot, que pueden influir en el resultado final. Una de las posibles soluciones consiste en una correcta identificación de parámetros, permitiendo un modelado del robot bastante preciso en simulación y mundo real [61]. Numerosos autores han trabajado en diversas técnicas para la identificación de parámetros, las cuales se comentarán posteriormente.
- Debido a la configuración del robot paralelo, los actuadores no actúan directamente sobre el elemento final del robot, sino que este esfuerzo de torsión se propaga a través de las dos o más cadenas cinemáticas (eso no pasa en robots serie, ya que se aplica directamente sobre el elemento final) [62]. Todo ello implica una pérdida

en la rigidez, y por tanto, una pérdida en precisión.

Como se puede observar, no es tan trivial y evidente la mejora en precisión, comparando con otro tipo de robots y, es por ello, por lo que la robótica paralela es aún un campo con numerosas posibilidades que aún no han sido exploradas.

Una de las soluciones para poder solventar algunas de las limitaciones relacionadas anteriormente, consiste en la identificación de parámetros, un área bastante explorada recientemente [63]. Como podemos ver en [64], la identificación de parámetros dinámicos consiste en ajustar la respuesta del modelo dinámico a datos experimentales medidos durante el movimiento del robot. En cuanto a las fuerzas aplicadas, generalmente se obtienen experimentalmente relacionándolas directamente con la la corriente aplicada. Por lo que respecta a la posición del sistema, se puede medir de forma precisa mediante el uso de encóders.

Estudios como el presentado en [65] proponen una metodología para la identificación de parámetros dinámicos en robots paralelos en función de parámetros relevantes. Básicamente, esta metodología se basa en comenzar con el modelo dinámico completo y, a partir de ahí, se empieza a reducir el modelo (gracias a las características geométricas que tienen los robots paralelos). Tras la simplificación del modelo completo, se obtiene un modelo reducido, en el que únicamente están presentes los *parámetros relevantes*.

Otros trabajos, con el objetivo también de explotar al máximo los puntos fuertes de los robots paralelos, proponen métodos más novedosos [66]. En

este artículo, los autores proponen añadir sensores extra en las articulaciones pasivas (no actuadas), con el objetivo de introducir información extra en el esquema de control, y así mejorar la precisión y velocidad en el robot. También se indica que, debido a la complejidad de la identificación de parámetros, en ocasiones ello implica que el modelo presente ciertas incertidumbres. Es por ello que, al tener información extra de sensores en articulaciones pasivas, esas incertidumbres se reducen, pudiendo incrementar el rendimiento del robot de manera sustancial (tal y como se puede apreciar en los resultados).

### 1.2.2. Aplicaciones de los manipuladores paralelos

Históricamente, cuando se ha hablado de robótica, siempre se ha asociado este concepto a la robótica humanoide. Posteriormente, debido a la industrialización, es mucho más común encontrarse con robots de tipo antropomórfico en la industria automovilística realizando tareas tales como pintado o soldado. Sin embargo, los robots paralelos han sido, probablemente, los más olvidados, sin prestar atención a la importancia que cobran en la actualidad.

Son numerosos los campos en los que se pueden encontrar los robots paralelos, siendo los más destacados los siguientes.

**Industria** Dentro del ámbito de la industria, los robots paralelos se comenzaron a utilizar de forma experimental para el testeo de neumáticos (usando la plataforma de Gough) [55] o, Stewart con su plataforma de 6

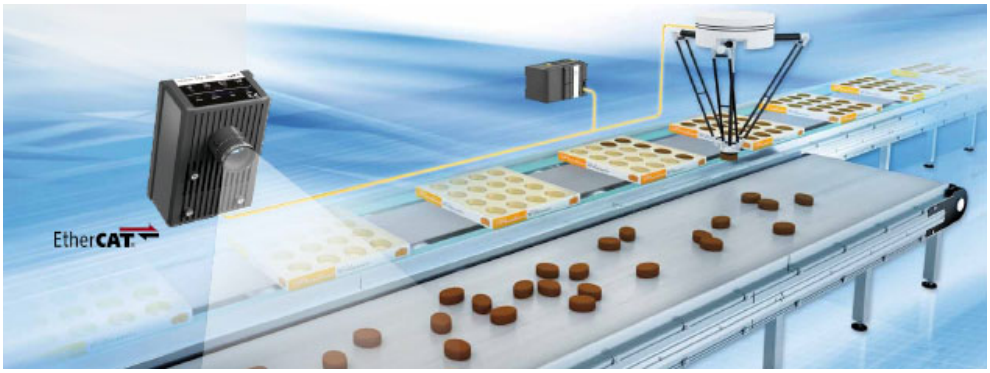



**Figura 1.5: Robots delta en el sector de la alimentación (*Omron*)**

grados de libertad en aplicaciones tales como la simulación de vuelo [56] (beneficiándose de las características de estos robots, tales como grandes aceleraciones, siendo muy apropiado para estas tareas). Como se ha visto anteriormente, otra de las características de los manipuladores paralelos es la alta velocidad y precisión y, es por ello, por lo que los robots Delta han cobrado una gran importancia [67]. Estos robots tienen genuinamente 3 grados de libertad (las tres traslaciones en el plano XYZ) aunque, se puede dotar de otros 3 grados más de libertad (DOF), añadiendo las 3 rotaciones adicionales al elemento final del robot. Sin embargo, los más populares son los de 4 DOF, como se puede ver en la figura 1.5 (tres traslaciones y una rotación), estando presentes en aplicaciones tales como *pick-and-place* de objetos de una masa de hasta 1 kg, las cuales deben ser realizadas a alta velocidad y precisión. Generalmente, este tipo de robots suelen estar integrados en sistemas de visión por computador en sectores como el alimentario [68], el electrónico o el farmacéutico.

**Sector aeroespacial** En cuanto al sector aeroespacial, los manipuladores paralelos han cobrado gran importancia, habiéndose desarrollado

numerosos simuladores vuelo en las últimas décadas. Instituciones tan prestigiosas como la NASA (*National Aeronautics and Space Administration*) han apostado por este tipo de robots para diseñar sus prototipos para este tipo de aplicaciones. Es, de hecho, en el ARC (*Ames Research Centre*), propiedad de la NASA, donde se encuentra el simulador de vuelo más realista del mundo: el VMS (*Vertical Motion Simulator*) [69]. Este robot paralelo de 6 DOF ofrece un rango de movimientos excepcionales (a la vez de inusuales para este tipo de manipuladores), alcanzando más de 18 metros en el eje vertical y, alrededor de 12 metros en el eje horizontal, con aceleraciones superiores a los 13 $m/s^2$. Por ello permite la recreación de maniobras críticas como despegue y aterrizaje de naves con un alto nivel de realismo, tal y como se puede apreciar en la encuesta realizada por la NASA [69], en la que se preguntó sobre el realismo del simulador a los pilotos que habían realizado una maniobra real previamente. Como concluye la NASA en cuanto al uso del VMS, *se ha conseguido reducir los riesgos, se ha maximizado la seguridad, se han reducido los tiempos de aprendizaje, por lo que, finalmente, ha permitido reducir los costes*.

A pesar de ser los simuladores de vuelo las aplicaciones que marcan la tendencia en el sector aeroespacial, también existen aplicaciones menos comunes tales como las relacionadas con el seguimiento de satélites, como se demuestra en [70], donde los autores presentan un robot de 2DOF en el que montan una antena en el efector final del robot, pudiendo realizar este seguimiento de una forma muy precisa.

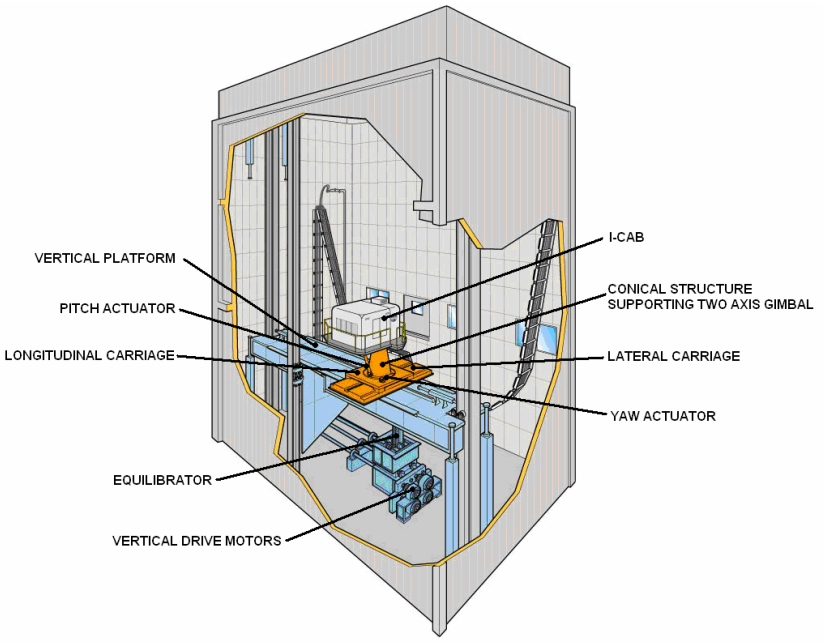



**Figura 1.6: El *Vertical Motion Simulator* (*NASA*)**

**Sector médico** El sector médico es, sin duda, la gran tendencia en estos momentos para aplicaciones en las que los robots paralelos están involucrados. La variedad de las aplicaciones es inmensa aunque todas ellas se benefician de uno de los puntos fuertes del robot paralelo: la gran precisión. En [71] se presenta la estructura de un robot hexápodo de 6 DOF usado para operaciones neurológicas, siendo su área de trabajo 100 x 100 x 50 mm con 15º de rotación. Este dispositivo proporciona una precisión de 20 micrómetros (figura 1.7). Otra aplicación distinta se puede encontrar en [72], en la que un robot en configuración delta tiene en su extremo final un potente microscopio (de 20 Kg), como herramienta. Por último, aplicaciones que están siendo tendencia son las relacionadas con la fisioterapia [73–75]. En ellas, se centran en el modelado y diseño de robots paralelos para la rehabilitación de tobillos (aspecto el cual se investigará en detalle en esta Tesis).

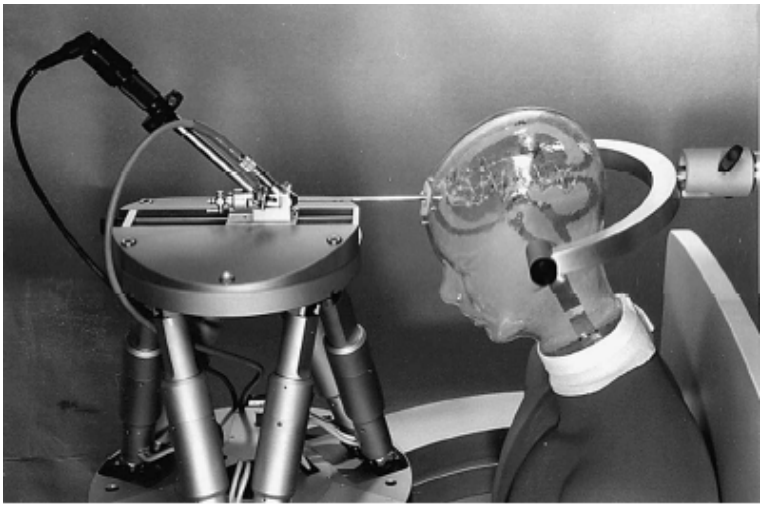


**Figura 1.7: Aplicación de la plataforma de Stewart en la neurocirugía**

Como se ha podido observar, el salto de calidad en cuanto a los manipuladores paralelos en el sector de la medicina ha venido determinado por la gran precisión en aplicaciones realmente complejas en las que el ser humano no es capaz de alcanzarla, las cuales son, generalmente, de un alto riesgo para el paciente.

**Otros sectores** Por último, la comunidad científica ha tratado de dar un giro a lo convencional y contribuir en áreas en las que la robótica aún estaba obsoleta. Un claro ejemplo de esto son los robots trepadores, dispositivos basados en la plataforma de Gough-Stewart, los cuales están dotados de dos anillos unidos por 6 actuadores lineales a través de articulaciones esféricas y universales. En [76] se pueden encontrar las principales caracterísitcias de este novedoso tipo de robots, así como las ventajas que éstos tienen sobre robots de configuración en serie.

Más concretamente, en [77] se exponen una serie de diseños de robots

**Figura 1.8: Robot trepador «TREPA»**

trepadores capaces de desplazarse a lo largo de estructuras tubulares, incluso también a través de las mismas. Tal y como comentan los autores, la idea de emplear los robots paralelos para la realización de estas tareas está enfocada a la minimización de situaciones de riesgo para trabajadores, los cuales deben trepar a este tipo de estructuras. Una aplicación realmente novedosa es el robot trepador de palmeras «TREPA» (figura 1.8), un prototipo realizado en la Universidad Politécina de Madrid y la Universidad Miguel Hernández de Elche, diseñado para la poda y fumigación de palmeras de forma autónoma.

## 1.3. Motivación y objetivos

Han sido numerosas las motivaciones que se han encontrado durante la realización de este trabajo, las cuales han desembocado en la realización

de esta Tesis, aportando un cierto número de novedades al estado de arte actual. Principalmente, estas motivaciones han desencadenado en la definición de una serie de objetivos a resolver durante este trabajo de investigación. Puesto que el mismo está relacionado tanto con la Ingeniería de Software e Ingeniería de Control, las motivaciones se van a separar en diferentes grupos.

### Desarrollo de software basado en componentes

Aunque el desarrollo de software basado en componentes no es un concepto novedoso dentro de la Ingeniería del Software, son numerosos los autores, los cuáles, están tratando de dar a conocer este concepto en el ámbito de la robótica [78]. El hecho de continuar usando viejas técnicas tiene como consecuencia que este campo siga ajeno a las ventajas que aportan las nuevas metodologías. Éste es el principal motivo por el que la aplicación de la ingeniería del software basada en componentes para el desarrollo de aplicaciones robóticas podría resolver una serie de problemas tales como:

- **Desarrollo desde cero**. Como se ha podido apreciar, el coste de desarrollar una pieza Software desde cero en su totalidad requiere de un gran esfuerzo tanto temporal como de recursos. Desgraciadamente, en la actualidad, ésta es una práctica muy común en la comunidad investigadora llegando, en muchas ocasiones, a redesarrollar algo ya existente previamente [18]. Debido a esto, los tiempos

se alargan considerablemente hasta que un software alcanza su madurez, siendo un hándicap importante en la comunidad científica.

- **Poca reusabilidad**. Relacionado con el punto anterior, a medida que el software alcanza un grado de madurez más elevado, mayores son las opciones para explotarlo experimentalmente. El hecho de no reutilizar software desarrollado previamente genera, de nuevo, un coste más elevado, con todo lo que ello implica. Esta carencia de reusabilidad puede ser apreciada, incluso, dentro de una misma aplicación.

- **Baja calidad**. Puesto que el ciclo de vida del software utilizado (no sólo la aplicación final, sino el entorno también) es muy corto, la calidad suele ser, generalmente, baja. Ello implica que el número de dificultades que se encuentran, incluso en las etapas de desarrollo, sean críticas para el éxito del proyecto.

- **Alto coste**. Debido a lo previamente comentado, el software para aplicaciones robóticas tiene un coste realmente elevado. Este hecho, por ejemplo, evita que pequeñas y medianas empresas (o, incluso, pequeños laboratorios científicos) puedan beneficiarse de las ventajas que aporta la robótica.

### Implementación de controladores dinámicos

Por otro lado, un objetivo común en el mundo de la robótica cosiste en el desarrollo de controladores que presenten una gran precisión. Como se ha

visto en apartados anteriores, existen robots paralelos muy precisos aunque, generalmente, la dinámica del sistema no es considerada dentro del bucle de control. En ocasiones, incluso usando controladores dinámicos (como se presentarán en los siguientes capítulos de esta Tesis) no es suficiente para controlar forma precisa un sistema en el que hay una serie de parámetros los cuales son dinámicos el tiempo. Una de las posibles soluciones reside en los controladores adaptativos, los cuales se comentarán en profundidad en posteriores capítulos de este documento.

**Desarrollo de aplicaciones integrando diferentes *middlewares***

En última instancia, cuando se desarrollan aplicaciones robóticas, normalmente, éstas están fuertemente acopladas al sistema (*framework* o *middleware*) para el que han sido desarrolladas (en ocasiones, esta dependencia llega a nivel de sistema operativo). Evidentemente, esto es una gran limitación en cuanto a la reusabilidad se refiere. Es por ello por lo que una de las claves para poder aprovechar todo el potencial de cada uno de los diferentes *middlewares* es la integración entre los mismos, y no el desarrollo por duplicado de cada aplicación para cada uno de los *middlewares*. Gracias a la integración de éstos, se pueden conseguir desarrollar aplicaciones finales de una mayor calidad, explotando los puntos fuertes de cada uno de los mismos.

Por todo lo anterior, los principales objetivos de esta Tesis son:

1. Diseñar controladores para un robot paralelo de tres grados de libertad aplicando los principios de la ingeniería del software basado

en componentes.

2. Implementar controladores dinámicos complejos y adaptativos, de forma modular, permitiendo su ejecución de forma distribuida.

3. Integrar controladores de fuerza y posición para la creación de un controlador híbrido.

4. Integrar los controladores implementados en los *middlewares* Orocos y ROS.

5. Diseñar e implementar una aplicación de bajo coste para la rehabilitación de miembros inferiores (simulación y experimental).

## 1.4. Organización y estructura de la memoria

El presente manuscrito se puede estructurar en tres partes diferenciadas:

En el primer capítulo de esta primera parte se ha realizado un estado el arte, con el objetivo de describir y analizar las investigaciones más recientes y actuales de la temática del trabajo, así como para agrupar elaboraciones similares dentro de un mismo marco conceptual. Puesto que la presente Tesis se centra tanto en el concepto del desarrollo del software basado en componentes, como en la implementación de controladores para un robot paralelo, este primer capítulo se ha dividido, a su vez, en dos secciones:

1. **Desarrollo de software basado en componentes**. En esta sección, además de analizar los últimos trabajos relacionados con la ingeniería del software basada en componentes enfocado al campo de la robótica, también se ha realizado un estudio sobre los costes que ello conlleva (así como sus implicaciones). Finalmente, se ha descrito el concepto de *middleware*, así como el desarrollo de software basado en modelos, proveyendo de un nivel extra de abstracción a la hora de diseñar software.

2. **Manipuladores paralelos.** En la primera parte de esta sección se han descrito las características principales que poseen los manipuladores paralelos, así como las ventajas que éstos tienen sobre otras configuraciones. Posteriormente, se ha hecho un análisis sobre cómo han evolucionado los controladores para este tipo de controladores, además de las principales dificultades y retos que tiene el diseño de controladores para este tipo de robots. Finalmente, se han expuesto una serie de aplicaciones, dividiéndolas por sectores, las cuales son las más comunes realizadas por robots paralelos.

La segunda parte está formada por tres capítulos, cada uno de los cuales corresponde a la principal publicación científica que se ha realizado. En esta segunda parte se expondrán los controladores dinámicos y adaptativos implementados de forma modular para un robot paralelo de tres grados de libertad, así como la creación de un controlador híbrido de fuerza y posición (usando un sensor de fuerza acoplado en el sistema). Todo ello, integrado mediante módulos en el *middleware* orientado a componentes, Orocos.

Concretamente, el capítulo 2 está basado en el siguiente artículo de revista:

- M. VALLÉS, J. CAZALILLA, A. VALERA, V. MATA, and A. PAGE, "Dynamic controllers implementation based on the OROCOS *middleware* | Implementación basada en el *middleware* OROCOS de controladores dinámicos pasivos para un robot paralelo," *RIAI - Revista Iberoamericana de Automática e Informática Industrial*, vol. 10, no. 1, 2013

Este primer artículo fue publicado en la *Revista Iberoamericana de Automática e Informática Industrial*, una revista con un IF (*Impact Factor*) de 0.475 puntos según *Thomson Reuters Journal Citation Report,*y situada en el cuartil 4. En él se hace una introducción al *middleware* Orocos, así como las ventajas que se consiguen siguiendo las pautas de la ingeniería del software basado en componentes. Además se presenta el robot de 3DOF y se implementan una serie de controladores dinámicos básicos, obteniendo unos resultados iniciales realmente satisfactorios.

En cuanto al capítulo 3, este se corresponde con el siguiente artículo de revista:

- J. CAZALILLA, M. VALLÉS, V. MATA, M. DÍAZ-RODRÍGUEZ, and A. VALERA, "Adaptive control of a 3-DOF parallel manipulator considering payload handling and relevant parameter models," *Robotics and Computer-Integrated Manufacturing*, vol. 30, pp. 468–477, oct 2014.

En este artículo, publicado en la prestigiosa revista *Robotics and Computer-Integrated Manufacturing*, con un factor de impacto de 2.077 puntos (que la sitúa entre las cinco revistas más importantes en este campo) y perteneciendo al cuartil 1 (Q1) se va mas allá y, además de implementar controladores dinámicos de mayor complejidad que en el capítulo anterior, se consigue diseñar e implementar un controlador adaptativo. Lo ciertamente novedoso de este artículo es que el modelo dinámico del robot se divide, independientemente, en parámetros del cuerpo rígido, parámetros de fricción y dinámica de los actuadores. De este modo, en el controlador adaptativo se pueden estudiar por separado o haciendo cualquier combinación entre ellos, todas las incertidumbres que puede presentar el sistema. Los resultados, tanto en simulación como de manera experimental con el manipulador paralelo, confirman lo esperado.

En cuanto al capítulo 4, éste trata sobre la implementación de un control híbrido de fuerza/posición, basándose en el siguiente artículo de revista:

- J. CAZALILLA, M. VALLÉS, Á. VALERA, V. MATA, and M. DÍAZ-RODRÍGUEZ, "Hybrid force/position control for a 3-DOF 1T2R parallel robot: Implementation, simulations and experiments," *Mechanics Based Design of Structures and Machines*, vol. 44, pp. 16–31, apr 2016.

Este artículo es una versión extendida de un trabajo que fue publicado en el *International Symposium on Multibody Systems and Mechatronics,* el cual fue galardonado como mejor paper por el *Scientific Committee for*

*MUSME,* finalmente, formando parte (la versión extendida) de un volumen especial en la revista internacional *Mechanics Based Design Of Structures And Machines*, la cual cuenta con un factor de impacto de 1.508, perteneciendo al cuartil 2 (Q2). En este artículo, usando los controladores dinámicos previamente implementados y la integración de un sensor de fuerza de 6DOF, se consigue diseñar un controlador híbrido de fuerza-posición, en el cual, las medidas provenientes del sensor de carga, realimentan en bucle cerrado al algoritmo de control, modificando la posición de referencia deseada. Además de intervenir en la posición de referencia deseada, también se ha conseguido que el robot sea capaz de seguir un patrón determinado de fuerza con una gran precisión.

La tercera parte está formada por el capítulo 5, en el que presenta una aplicación completa para la rehabilitación de miembros inferiores. Ésta, está basada en una publicación de la revista científica *Robotica (Cambridge University)*, con un factor de impacto de 0.824 (Q3) y posicionada la número 17 entre las revistas indexadas en la catergoría de robótica:

- M. VALLÉS, J. CAZALILLA, Á. VALERA, V. MATA, Á. PAGE, and M. DÍAZ-RODRÍGUEZ, "A 3-PRS parallel manipulator for ankle rehabilitation: towards a low-cost robotic rehabilitation," *Robotica*, pp. 1–19, mar 2015.

El objetivo de este artículo es, con todo lo desarrollado anteriormente, crear una aplicación real mediante la integración entre *middlewares*. En este caso, la aplicación desarrollada consiste en diseñar diversos ejercicios con el robot paralelo, con el objetivo de rehabilitar miembros inferio-

res. Estos ejercicios están basados tanto en el controlador de posición puro, como en el híbrido fuerza/posición. Finalmente, se presenta y demuestra la integración Orocos-ROS mediante un ejercicio de tele-operación en el que el personal sanitario no está físicamente en la misma sala que el robot pero que, a su vez, son capaces de realizar todos los movimientos del mismo con un dispositivo Bluetooth y visualizarlo en tiempo real en un modelo CAD.

Finalmente, en la cuarta parte de esta Tesis se exponen las conclusiones generales, además de los posibles trabajos futuros dentro de este mismo marco.

## 1.5. Aportaciones de esta Tesis

Durante el periodo de investigación, se ha contribuido con la comunidad científica tanto con la publicación de diversos artículos en Revistas Científicas, como con la ponencia de nuestra actividad investigadora en Congresos Internacionales. Además, también se ha co-dirigido un Proyecto Final de Carrera y una Tesina Final de Máster. A continuación, se pueden apreciar las aportaciones realizadas, así como el factor de impacto y el cuartil de la Revista.

### Artículos de revista (Indexados en JCR)[1])

**J. CAZALILLA, M. VALLÉS, V. MATA, M. DÍAZ-RODRÍGUEZ, and A. VALERA**, Adaptive control of a 3-DOF parallel manipulator considering payload handling and relevant parameter models, *Robotics and Computer-Integrated Manufacturing*, vol. 30, pp. 468477, oct 2014. Factor de impacto basado en *JCR* de 2.077 y perteneciendo al cuartil 1 (Q1).

**J. CAZALILLA, M. VALLÉS, Á. VALERA, V. MATA, and M. DÍAZ-RODRÍGUEZ**, Hybrid force/position control for a 3-DOF 1T2R parallel robot: Implementation, simulations and experiments, *Mechanics Based Design of Structures and Machines*, vol. 44, pp. 1631, apr 2016. Factor de impacto basado en *JCR* de 1.508 y perteneciendo al cuartil 2 (Q2).

[1] *Journal Citation Reports* es una publicación anual que realiza el *Institute for Scientific Information*, que actualmente es parte de la empresa Thomson Scientific, evaluando el impacto y relevancia de las principales revistas científicas

**M. VALLÉS, J. CAZALILLA, Á. VALERA, V. MATA, Á. PAGE, and M. DÍAZ-RODRÍGUEZ**, A 3-PRS parallel manipulator for ankle rehabilitation: towards a low-cost robotic rehabilitation, *Robotica*, pp. 119, mar 2015. Factor de impacto basado en *JCR* de 0.824 y perteneciendo al cuartil 3 (Q3).

**M. VALLÉS, J. CAZALILLA, A. VALERA, V. MATA, and A. PAGE**, Dynamic controllers implementation based on the OROCOS middleware | Implementación basada en el *middleware* OROCOS de controladores dinámicos pasivos para un robot paralelo, *RIAI - Revista Iberoamericana de Automática e Informática Industrial*, vol. 10, no. 1, 2013. Factor de impacto basado en *JCR* de 0.475 y perteneciendo al cuartil 4 (Q4).

## Artículos de revista (no indexado en JCR)

**J. CAZALILLA, M. VALLÉS, V. MATA, M. DÍAZ-RODRÍGUEZ, and A. VALERA**, Implementation of a 3-DOF parallel robot adaptive motion controller, *International Journal of Mechanics and Control*, vol. 15, no. 1, 2014. Factor de impacto basado en *SCR* [2] de 0.117 y perteneciendo al cuartil 4 (Q4).

[2]The SCImago Journal & Country Rank. http://www.scimagojr.com

## Artículos de congreso

**J. CAZALILLA, M. VALLES, M. DIAZ-RODRIGUEZ, V. MATA, A. SORIANO, and A. VALERA**, Implementation of dynamic controllers using real-time middleware for a low-cost parallel robot, *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 47, pp. 20842084, IEEE, may 2014. Catalogada como la Mejor Conferencia sobre Robótica del Mundo, por la *Microsoft Academic* y la *Excellence in Research in Australia* (ERA).

**J. CAZALILLA, M. VALLÉS, A. VALERA, V. MATA, AND M. DÍAZ-RODRÍGUEZ**, Implementation of Force and Position Controllers for a 3DOF Parallel Manipulator, in *Mechanisms and Machine Science*, vol. 25, pp. 359369, 2015. Factor de impacto basado en *SCR* de 0.133.

**J. CAZALILLA, M. VALLÉS MIQUEL, V. MATA AMELA, M. DÍAZ-RODRÍGUEZ, and Á. VALERA FERNÁNDEZ**, Implementation of a 3-DOF Parallel Robot Adaptive Motion Controller, in *International Workshop on Robotics in Alpe-Adria-Danube Region (RAAD)*, (Portoroz), pp. 303 310, Jozef Stefan Institute, 2013. Factor de impacto basado en *SCR* de 0.152.

**E. BRAU, J. CAZALILLA, M. VALLÉS, A. BESA, A. VALERA, V. MATA, and A. PAGE**, Dynamic calibration of force platforms by means of a parallel robot, in *BIODEVICES 2013 - Proceedings of the International Conference on Biomedical Electronics and Devices*, 2013. Factor de impacto basado en *SCR* de 0.11.

**A. SORIANO, L. MARÍN, R. JUAN, J. CAZALILLA, A. VALERA, M. VALLÉS**., Plataforma Robótica De Bajo Coste Y Recursos Limitados Basada En Arduino Y Dispositivos Móviles, en *Jornadas de Automática 2014*, p. 9

## Supervisión de TFG/TFM

**CORONADO VALLÉS**, **JORGE**. Tesina de Máster de Automática en Informática Industrial. *Desarrollo de aplicaciones embebidas de control en robots móviles*. Departamento de Ingeniería de Sistemas y Automática. Universitat Politècnica de Valencia. Septiembre 2013

**PONZ CAMPS, FRANCISCO DE BORJA.** Trabajo Final de Grado. *Desarrollo de algoritmos de navegación automática para robots móviles y vehículos ligeros mediantes sensores láser de rango.* Escuela Técnica Superior de Ingeniería del Diseño. Universitat Politècnica de Valencia. Septiembre 2014

# Parte II

## Desarrollo de controladores de posición y/o de fuerza de robots basado en componentes

# Capítulo 2

## Implementación basada en el middleware OROCOS de controladores dinámicos pasivos para un robot paralelo

ESTE capítulo está basado en las primeras publicaciones del autor [79] y [80]. En estos artículos, se presenta una metodología con el objetivo de desarrollar software (concretamente, controladores) usando el principio del CBSD, usando el *middleware* de control Orocos. Además de esto, se han implementado una serie de controladores dinámicos para un robot paralelo de tres grados de libertad.

**Resumen**

La complejidad actual de los sistemas robotizados y de las aplicaciones que éstos deben realizar requiere que los robots dispongan de un control automático que permita la ejecución de las distintas tareas que forman parte del algoritmo de control y que tenga en cuenta cuestiones relacionadas por ejemplo con la periodicidad, el modo de ejecución, el hardware que se utilizará, etc. Para el desarrollo de este tipo de aplicaciones de control en los últimos años se tiende a la programación basada en componentes puesto que ésta permite obtener código reusable. Así mismo también se está incrementando la utilización de *middlewares* que permiten la abstracción de los sistemas operativos, el soporte de tiempo real y la infraestructura de comunicaciones. En el presente artículo se propone la utilización de un *middleware* orientado especialmente a la robótica: OROCOS. Así se describe cómo haciendo uso de una de sus librerías, Orocos Toolchain, se han desarrollado una serie de componentes correspondientes a distintos algoritmos para el control dinámico de robots, aplicándose a un robot paralelo de 3 grados de libertad (DOF).

## 2.1. Introducción

El control automático de los sistemas robotizados actuales involucra cada vez más la implementación de distintas tareas a realizar por el robot con distinto grado de complejidad, de naturaleza periódica o aperiódica, ejecutadas de manera local o distribuidas a través de una red de comunicacio-

nes y para lo cual es necesario manejar hardware de distinta naturaleza. Abordar la implementación de controladores para una nueva plataforma robotizada supondría volver a desarrollar código adecuado para el nuevo hardware o en el mejor de los casos adaptar el ya existente. En la última década ha habido un interés creciente en los sistemas software basados en componentes que faciliten el trabajo en situaciones como las anteriormente descritas a partir del desarrollo de componentes reusables y que puedan interoperar fácilmente entre ellos. Según [81] plantear el desarrollo de código bajo esta filosofía es posible debido a la situación de evolución o maduración en la que se encuentra actualmente la arquitectura del software en donde la programación orientada a objetos se encuentra totalmente establecida y en la que un componente podría entenderse como un objeto, destacando su interoperatividad y reusabilidad. Así, en [39] se comenta que el desarrollo de software basado en componentes (CBSD) es una alternativa de diseño que favorece la reutilización de elementos software y facilita el desarrollo de sistemas a partir de elementos preexistentes, siendo un componente software una unidad de composición con interfaces bien definidas y un contexto de uso explícito. Como complemento a esta definición, en [82] se expone que el CBSD se basa en la descomposición de un software en componentes lógicos o funcionales con interfaces bien definidas, con el fin de facilitar el tiempo de desarrollo y mejorar el mantenimiento del sistema. Analizar el coste de un CBSD, sin embargo, es un tema que ha evolucionado poco dentro de la ingeniería del software y en su estimación resulta importante considerar, no sólo en el coste de la creación, sino también la eficiencia y productividad. A la hora de desarrollar un CBSD es conveniente contar con un entorno de trabajo

que dé soporte a este tipo de filosofía de programación del software. Dentro de la robótica existen algunos *middleware* que facilitan dicha labor. Los *middleware* funcionan como una capa de abstracción de software distribuida, que se sitúa entre las capas de aplicaciones y las capas inferiores (sistema operativo y red). Así el *middleware* abstrae de la complejidad y heterogeneidad de las redes de comunicaciones subyacentes, así como de los sistemas operativos y lenguajes de programación, proporcionando una API para la fácil programación y manejo de aplicaciones distribuidas [83]. En la actualidad existen diversos *middlewares*, muchos de ellos pensados para aplicaciones robóticas como son Player [30, 84], Orca [31], Miro [35], OpenRDK , Marie [85], Smartsoft [33], Orocos [51] o ROS [38].

Para el presente trabajo se escogió el *middleware* OROCOS por estar especialmente pensado para robots industriales y por la librería de tiempo real que ofrece, la Orocos Toolchain, la cual, lo hace especialmente indicado para aplicaciones en las que se requiera manejo de tiempo real, como es el caso de la plataforma robótica escogida, un robot paralelo.

Un Manipulador Paralelo (MP) es un mecanismo en donde el movimiento del elemento final, generalmente una plataforma móvil, está conectado a la base fija por medio de, al menos dos cadenas cinemáticas [86]. Los MPs han supuesto un campo de investigación muy activo durante los últimos años. En la actualidad la aplicación de estos resultados de investigación se ha transferido a la industria [87], principalmente debido a las ventajas que los MPs presentan frente a los robots serie (o de cadena cinemática abierta) ya que, al dividir la carga entre varias articulaciones, los MPs proporcionan alta velocidad y precisión de posicionado, capacidad de car-

ga y rigidez. Debido a estas ventajas podemos encontrar diversas aplicaciones de los MPs, como por ejemplo el popular robot Delta [58], que con sólo 3 DOF prismáticos se puede aplicar satisfactoriamente a tareas de pick-and-place [88], aplicaciones médicas, como equipos de reanimación cardio-pulmonar [89], máquinas-herramienta [87], simuladores de vuelo o conducción, máquinas de pruebas, etc. [55, 56, 58, 90].

Además de los conocidos y habituales esquemas de control PD y PID, se pueden encontrar una gran variedad de esquemas de control utilizados en los MPs, como por ejemplo, control robusto no lineal [91], control robusto [92], control borroso [93] y control por modos deslizantes [94] entre otros.

Sin embargo, para controlar de forma efectiva un manipulador consiguiendo una respuesta rápida y precisa es necesario el diseño de esquemas de control basados en el modelo dinámico del robot [95]. El modelo utilizado en este trabajo corresponde a un modelo reducido para el MP que se obtiene siguiendo la metodología propuesta previamente por [65], basada en considerar simplificaciones debido a las simetrías y geometrías de las partes del robot.

El control dinámico del MP se obtiene a partir de controladores basados en la pasividad y la parte novedosa de este artículo corresponde a la implementación que se ha llevado a cabo de dichos controladores basada en el concepto de componentes con el objetivo de permitir su reusabilidad en otras aplicaciones. Así, se describe la descomposición de éstos en componentes y su implementación haciendo uso del *middleware* para robots OROCOS.

Así, el resto del artículo se organiza de la siguiente manera. En la segunda sección 2.2, se describe el funcionamiento del *middleware* OROCOS utilizado para este trabajo. En la sección 2.3 se describe el robot paralelo empleado en el trabajo, así como las ecuaciones obtenidas para su modelado. En la sección 2.4 se procede a describir los controladores diseñados así como su implementación. En la sección 2.5 se muestran los resultados reales conseguidos con la plataforma de control desarrollada sobre el robot real. Y se finaliza con la sección 2.6 en donde se describen las conclusiones del trabajo

## 2.2. El middleware OROCOS

OROCOS (Open RObot COntrol Software) proporciona las cuatro funciones principales de un *middleware*: abstracción del sistema operativo y soporte de tiempo real, servicios *middleware*, infraestructura de comunicación y un modelo basado en componentes [51].

El proyecto OROCOS se puso en marcha en el 2001 a propuesta de EURON (la Red Europea de Robótica) gracias a un proyecto subvencionado por la Comunidad Europea. Pese a que OROCOS es un *middleware* de reciente creación y en continuo desarrollo, poco a poco se está consolidando como una de las mejores opciones para realizar el control de robots, estando presente en la actualidad en diversos proyectos de investigación de ámbito internacional.

El proyecto OROCOS soporta en la actualidad tres librerías: Kinematics

and Dynamics (KDL), Bayesian Filtering (BFL) y Orocos Toolchain.

La librería KDL permite reducir el modelado y la especificación del movimiento a un problema meramente geométrico (aunque con varios sistemas de referencia) con cálculos matemáticos, pudiéndose resolver un movimiento en base a unas especificaciones en cada una de las articulaciones. Para ello desarrolla un marco de aplicación independiente para el modelado, y otro para el cálculo de cadenas cinemáticas en robots, modelos biomecánicos humanos, figuras animadas por ordenador, máquinas herramientas, etc. Además, proporciona librerías predefinidas de clases de objetos geométricos, cadenas cinemáticas de varias clases (serie, humanoide, paralelas o móviles) así como su especificación de movimiento e interpolación.

La librería BFL proporciona un marco de aplicación independiente para, por ejemplo, el procesamiento de la información recursiva y algoritmos de cálculo basado en la regla de Bayes, tales como filtros de Kalman o Filtros de Partículas. Estos algoritmos pueden ser ejecutados como una aplicación en tiempo real, o ser utilizados para la estimación de las aplicaciones de cinemática y dinámica.

Relacionado con esta librería, cabe destacar que en la actualidad existen aplicaciones similares a la que propone OROCOS, llegando a ser, incluso, mejores y más sencillas en su uso. Por ello, el desarrollo en los últimos años de la BFL es mínimo, ya que no es tan usado como otras librerías del proyecto OROCOS.

Orocos ToolChain es la librería de más reciente creación del proyecto

OROCOS. Su primera versión apareció en Julio de 2010 y, poco a poco, se está convirtiendo en la herramienta más usada. Dado que ha sido la librería utilizada en el trabajo del presente artículo a continuación se verá con más detalle.

### 2.2.1. La librería Orocos Toolchain

La particularidad de la ToolChain es que en esta herramienta vienen incluidas otras herramientas como:

**AutoProj,** que es una herramienta para descargar y compilar las librerías necesarias de forma automática.

**Real Time Toolkit (RTT)** permite a los desarrolladores la creación de componentes totalmente configurables (incluso en tiempo de ejecución) e interactivos basados en el control de aplicaciones en tiempo real, pudiéndose usar en aplicaciones con características tales como capturar y graficar el flujo de datos entre los componentes, modificar los algoritmos en tiempo de ejecución, configurar los componentes y la aplicación a partir de archivos XML, interactuar con otros dispositivos directamente desde una interfaz gráfica de usuario o mediante el *prompt*, ampliar las aplicaciones con estructuras de datos propias y ejecutar la aplicación tanto en sistemas operativos estándar como sistemas de tiempo-real. Además, el RTT, permite que los componentes se ejecuten en cualquier sistema operativo ofreciendo todas las funciones y comandos del tiempo real, como la comunica-

ción de los componentes y la configuración de los mismos mediante un archivo XML.

**Orocos Component Library** permite crear todos los tipos de componentes (apoyándose en RTT). Cada uno de los componentes está definido a partir de una primitiva llamada *TaskContext*, la cual define el entorno (contexto) básico que debe tener un componente en OROCOS (que posteriormente se modifica y configura según las preferencias del usuario). Este contexto está descrito por cinco primitivas: *Event*, *property*, *Command*, *Method* y *Data port*. En la siguiente subsección se explica más en detalle en qué consiste un componente OROCOS.

**OroGen y TypeGen** son dos herramientas para generar código OROCOS a partir de unas cabeceras descritas, o de un fichero de descripción del componente (con una sintaxis ya predefinida).

De este modo, y en un único paquete, está todo lo necesario para comenzar a realizar componentes que tengan distintas funcionalidades. Además, una de las ventajas de Orocos ToolChain es su soporte multiplataforma ya que se puede trabajar tanto en Linux como en Windows o MAC (en algunos casos, sin utilizar la parte de tiempo real).

### 2.2.2. Componentes en OROCOS

Uno de los objetivos de OROCOS es que el software para sistemas complejos no se construya únicamente en tiempo de compilación sino también

en tiempo de lanzamiento a ejecución o incluso en ejecución. Esto se permite mediante la generación y ejecución de componentes. Un componente OROCOS es una unidad básica de funcionalidad que puede ejecutar (en tiempo real) uno o más programas en un único hilo (o *thread*). De cara al usuario, el sistema está libre de prioridades y todas las operaciones son no bloqueantes, incluido el intercambio de datos y otras formas de comunicación, como por ejemplo los eventos y comandos. Otro aspecto importante es que los componentes de tiempo real pueden comunicarse de forma transparente con componentes que no sean de tiempo real.

Así, mediante el diseño modular, se permite un rápido seguimiento del flujo de ejecución del programa, facilitando la creación de nuevos componentes que, combinándolos con otros, pueden conseguir nuevas funcionalidades. Esta estructura modular, además, permite una ejecución distribuida del código correspondiente a módulos distintos, pudiéndose obtener tiempos de ejecución menores que los obtenidos mediante una ejecución serie. Otra de las ventajas del planteamiento basado en componentes para desarrollar software es que el código se puede reusar, pudiéndose parametrizar su ejecución en tiempo de lanzamiento.

## 2.3. El robot paralelo desarrollado

En el ámbito del proyecto de investigación Identificación de Parámetros Físicos en Sistemas Mecánicos Complejos (DPI2005- 08732-C02-01/02) del Plan Nacional del Ministerio de Ciencia e Innovación se desarrolló un manipulador paralelo. Éste puede verse como una plataforma móvil

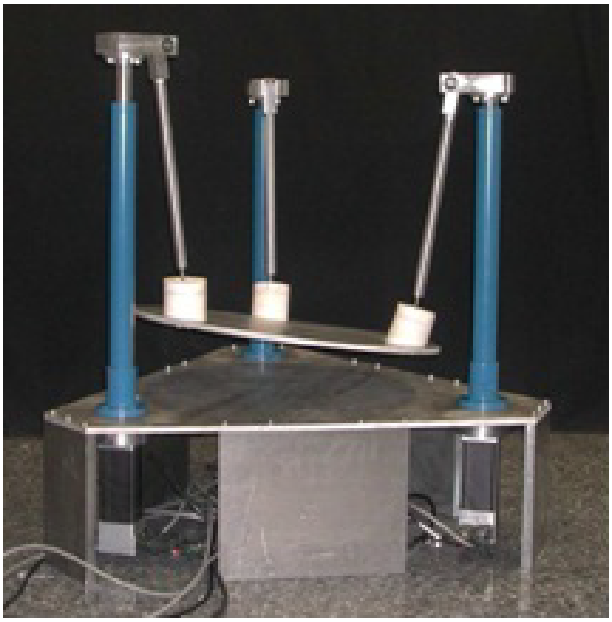


**Figura 2.1: Robot paralelo 3-PRS desarrollado**

conectada a la base fija mediante tres cadenas cinemáticas. Cada cadena consiste en un motor que mueve un actuador de husillo de bolas y una varilla de conexión. La figura 2.1 muestra el aspecto mecánico del robot desarrollado.

La elección de la arquitectura y movimiento del robot paralelo vino determinada por la necesidad de desarrollar un robot de bajo coste capaz de generar dos movimientos rotacionales (*roll* y *pitch*) y un movimiento lineal de elevación (z). Cuando se combinan movimientos prismáticos y de revolución se pueden encontrar arquitecturas 3-PRS [96] y 3-RPS [97], donde la notación R, P y S hacen referencia a las articulaciones de rotación, prismáticas y esféricas respectivamente. Para este robot se escogió la arquitectura 3-PRS tras comparar las ventajas e inconvenientes de cada una de ellas.

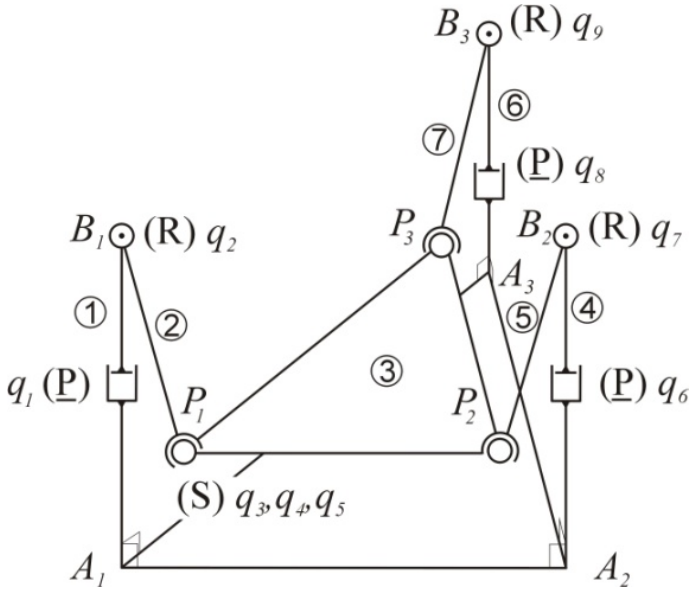



**Figura 2.2: Localización de los sistemas de coordenadas**

Para poder establecer el control del MP fue necesario desarrollar tanto el modelo cinemático como el dinámico del robot, los cuales se describen a continuación.

### 2.3.1. Modelo cinemático directo

La cinemática directa del MP consiste en, dados los movimientos lineales de los actuadores, encontrar los ángulos de balanceo ($\beta$) y alabeo ($\gamma$) y la elevación (z). Se puede utilizar la notación de Denavit-Hartenbert para establecer las coordenadas generalizadas del modelo cinemático. La tabla 2.1 muestra los parámetros D-H del robot considerado. A partir de la tabla 2.1 se puede ver como a partir de 9 coordenadas generalizadas, se puede definir la cinemática del robot. La localización de los sistemas de coordenadas para modelar la cinemática se muestra en la figura 2.2.

| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| $d_i$ | $q_1$ | 0 | 0 | 0 | 0 | $q_6$ | 0 | $q_8$ | 0 |
| $a_i$ | 0 | 0 | $l_a$ | 0 | 0 | 0 | 0 | 0 | 0 |
| $\theta_i$ | $\frac{\pi}{6}$ | $q_2$ | $q_3$ | $q_4$ | $q_5$ | $\frac{5\pi}{2}$ | $q_7$ | $\frac{-\pi}{2}$ | $q_9$ |
| $\alpha_i$ | 0 | $\frac{\pi}{2}$ | 0 | $\frac{\pi}{2}$ | $\frac{\pi}{2}$ | 0 | $\frac{\pi}{2}$ | 0 | $\frac{\pi}{2}$ |

**Tabla 2.1: Parámetros D-H para el PM de 3-DOF**

Las tres patas del robot paralelo se conectan a la plataforma móvil formando un triángulo equilátero. Por ello se puede ver que la longitud entre $p_i$ y $p_j$ es constante e igual a $l_m$. Así,

$$f_1(q_1, q_2, q_6, q_7) = \|(\vec{r}_{A1B1}+\vec{r}_{B1p1})-(\vec{r}_{A1A2}+\vec{r}_{A2B2}+\vec{r}_{B2p2})\|-l_m = 0 \tag{2.1}$$

$$f_2(q_1, q_2, q_8, q_9) = \|(\vec{r}_{A1B1}+\vec{r}_{B1p1})-(\vec{r}_{A1A3}+\vec{r}_{A3B3}+\vec{r}_{B3p3})\|-l_m = 0 \tag{2.2}$$

$$\begin{aligned} f_3(q_6, q_7, q_8, q_9) = \|&(\vec{r}_{A1A3} + \vec{r}_{A3B3} + \vec{r}_{B3p3}) \\ &-(\vec{r}_{A1A2} + \vec{r}_{A2B2} + \vec{r}_{B2p2})\| - l_m = 0 \end{aligned} \tag{2.3}$$

En la cinemática directa, la posición de los actuadores es conocida. Así, el sistema de ecuaciones 2.1-2.3 se puede ver como un sistema no-lineal con $q_2$, $q_7$y $q_9$ como desconocidas. Para la resolución del sistema no lineal el problema cinemático directo, en este trabajo, se ha utilizado el método

numérico de Newton-Rhapson ya que tiene una convergencia muy rápida (convergencia cuadrática) cuando la estimación inicial está cerca de la solución deseada [98]. El método es iterativo y se puede escribir como:

$$\begin{bmatrix} q_2 \\ q_7 \\ q_9 \end{bmatrix}^{i+1} = \begin{bmatrix} q_2 \\ q_7 \\ q_9 \end{bmatrix}^{i} J_i^{-1} \begin{bmatrix} f_1(q_2, q_7) \\ f_2(q_2, q_9) \\ f_3(q_7, q_9) \end{bmatrix}^{i} \tag{2.4}$$

En la ecuación $i$ significa que las variables y funciones se evalúan en la iteración $i$. La matriz $J$ es la matriz Jacobiana de $f_i$ con respecto a las variables $[q_2, q_7, q_9]$. El proceso iterativo finaliza cuando

$$\sqrt{(f_1(q_2, q_7)^i)^2 + (f_2(q_2, q_9)^i)^2 + (f_3(q_7, q_9)^i)^2} < \varepsilon \tag{2.5}$$

donde $\varepsilon$ es una cantidad positiva pequeña establecida por el usuario.

El método de Newton-Rhapson requiere una aproximación inicial tan cercana como sea posible al valor solución. En este caso esto no supone ningún problema ya que la posición inicial de la articulación que conecta la plataforma con el actuador es aproximadamente $\frac{2\pi}{5}$. La subsecuente aproximación inicial considera los valores de la posición previa del robot.

La localización de la plataforma móvil se define usando el sistema de coordenadas unido a él. Una vez encontradas las coordenadas generalizadas para las patas del robot, se puede encontrar la posición de los puntos $p_i$. Estos tres puntos comparten el plano de la plataforma. Basado en estos

puntos se puede construir la matriz rotacional de la plataforma con respecto a la base. A partir de dicha matriz de rotación se pueden calcular directamente tanto el resto de coordenadas generalizadas ($q_3$, $q_4$y $q_5$) del robot como la elevación (z) y los ángulos de balanceo ($\beta$) y alabeo ($\gamma$) de la plataforma.

### 2.3.2. Modelado cinemático inverso

La cinemática inversa consiste en, dados los ángulos de balanceo y alabeo de la plataforma ($\beta$ y $\gamma$) y la elevación (z), encontrar el movimiento lineal de los actuadores. Usando el sistema de ángulos fijo X-Y-Z; la matriz de rotación se puede definir como:

$$ {}^{O}R_p = \begin{bmatrix} c_\alpha c_\beta & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta c_\gamma - s_\alpha s_\gamma \\ s_\alpha c_\beta & s_\alpha s_\beta s_\gamma - c_\alpha c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma \\ -s_\beta & c_\beta s_\gamma & c_\beta c_\gamma \end{bmatrix} \tag{2.6} $$

donde $c$* y $s$* hacen referencia al cos(*) y sen(*) respectivamente. Dados $\gamma$ y $\beta$, el ángulo de guiñada o *yaw* ($\alpha$) se puede encontrar como sigue:

$$ \alpha = arctan2(s_\beta s_\gamma, (c_\gamma + c_\beta)) \tag{2.7} $$

Las posiciones de los actuadores ($q_1$, $q_6$ y $q_8$) se pueden calcular directamente a partir de dicha matriz de rotación.

### 2.3.3. Modelado dinámico

Como es bien conocido, la ecuación dinámica de los robots manipuladores se puede expresar mediante la siguiente ecuación no lineal:

$$M(\vec{q},\vec{\Phi})\cdot\ddot{\vec{q}} + \vec{C}(\vec{q},\dot{\vec{q}},\vec{\Phi}) + \vec{G}(\vec{q},\vec{\Phi}) = \vec{\tau} \qquad (2.8)$$

donde $M(\vec{q},\vec{\Phi})$, $\vec{C}(\vec{q},\dot{\vec{q}},\vec{\Phi})$ y $\vec{G}(\vec{q},\vec{\Phi})$ son la matriz de masas del sistema, el vector de fuerzas centrífugas y de Coriolis y los términos gravitacionales respectivamente, y como se puede apreciar, dependen de los parámetros dinámicos $\vec{\Phi}$ . $\vec{q}$ y $\vec{\tau}$ son los vectores de las coordenadas y pares generalizados. Para poder obtener los términos de la ecuación dinámica se debe construir el modelo para que éste esté en forma lineal a los parámetros dinámicos [65, 99]:

$$K(\vec{q},\dot{\vec{q}},\ddot{\vec{q}})\cdot\vec{\Phi} = \vec{\tau} \qquad (2.9)$$

En la ecuación 2.9 solo se puede identificar un conjunto de parámetros porque algunos de ellos tienen una contribución pequeña o insignificante en el comportamiento dinámico del sistema. Además, éstos son propensos al ruido en las medidas y a dinámicas no modeladas [100]. Para obtener la identificación del modelo dinámico se ha utilizado una metodología que aparece en [65].

Los términos de la ecuación 2.8 se pueden obtener una vez realizada la identificación del proceso. Así, se pueden calcular los términos del vector

gravitacional haciendo cero las velocidades y aceleraciones generalizadas de la ecuación 2.9:

$$K(\vec{q}, \dot{\vec{q}} = 0, \ddot{\vec{q}} = 0) \cdot \vec{\Phi} = \vec{G}(\vec{q}) \tag{2.10}$$

Los términos centrífugos y de Coriolis dependen de las coordenadas generalizadas y de las velocidades. Haciendo cero las aceleraciones generalizadas de la ecuación 2.9 otra vez se puede establecer que:

$$K(\vec{q}, \dot{\vec{q}}, \ddot{\vec{q}} = 0) \cdot \vec{\Phi} + \vec{G}(\vec{q}) + M^d A_d^{-1} \vec{b} = \vec{C}(\vec{q}, \dot{\vec{q}}) \tag{2.11}$$

Por último, la matriz de masa se puede determinar de la forma siguiente:

$$K^*(\vec{q}, \ddot{\vec{q}} = 0, \vec{e_i}) \cdot \vec{\Phi} - \vec{G}(\vec{q}) = M_i(\vec{q}) \tag{2.12}$$

Siendo $M_i(\vec{q})$ la columna i-ésima de la matriz de masas, y $\vec{e_i} = [0 \dots 1 \dots 0]^T$ un vector columna con un 1 en la posición i-esima.

Por lo tanto, las ecuaciones diferenciales que describen la ecuación del movimiento 2.8 se puede construir utilizando las ecuaciones 2.9 - 2.12.

### 2.3.4. Desarrollo del manipulador paralelo 3-PRS

En el manipulador paralelo 3-PRS desarrollado se pueden distinguir dos partes: la unidad mecánica y la unidad de control.

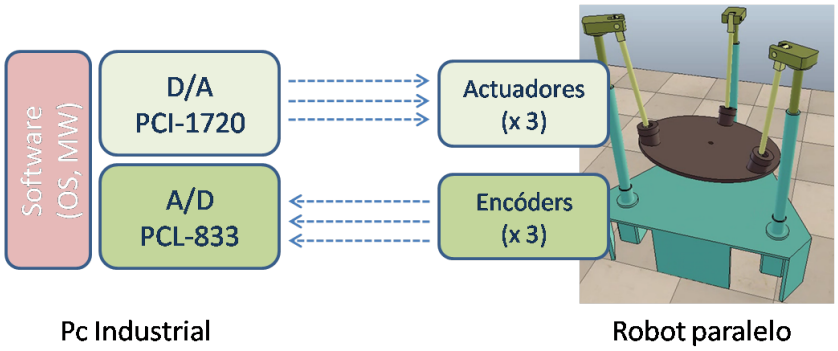



**Figura 2.3: Arquitectura de control del robot paralelo**

En lo que se refiere a la unidad mecánica (ver figura 2.1), para actuar sobre las 3 articulaciones del robot se han utilizado 3 motores *brush-less* BMS465, de AEROTECH, equipados con encóders que proporcionan 4.000 cuentas por revolución. Para actuar sobre dichos motores se utilizan las etapas de amplificación BA10, de AEROTECH.

De la misma forma que el robot paralelo, la especificación y el diseño del sistema de control del robot se realizó íntegramente en la Universidad Politécnica de Valencia. Así se optó por una arquitectura basada en un PC industrial de altas prestaciones 4U Rackmount con 7 slots PCI y 7 ISA. Tiene un procesador Intel R Core 2 Quad/Duo processor a 2,5GHz y una memoria de 4 Gb RAM.

El sistema (figura 2.3) se complementa con las tarjetas de adquisición de datos necesarias para poder acceder a las señales de robot. En concreto se tiene una tarjeta Advantech PCI-1720 para el suministro de las acciones de control a los 3 motores. Así mismo, se utiliza otra tarjeta de Advantech: la PCL-833 para la lectura de encóders.

La arquitectura propuesta proporciona 2 ventajas muy interesantes: su

bajo precio y su flexibilidad. Al ser un entorno abierto basado en PC se puede utilizar cualquier entorno de programación lo que permite, por ejemplo, implementar diferentes algoritmos de generación de referencias y control no-lineal, utilización de sensorización avanzada, como por ejemplo sensores de fuerza/par, visión artificial, sistemas inerciales, etc.

## 2.4. Diseño e implementación en Orocos de los controladores

### 2.4.1. Algoritmos de control para el robot paralelo

Los algoritmos de control implementados en este trabajo corresponden a controladores basados en la pasividad. Éstos resuelven el problema de control mediante la explotación de la estructura física del sistema robotizado, y en especial de su propiedad de pasividad. La filosofía de diseño de estos controladores es reconfigurar la energía natural del sistema de tal forma que se consiga el objetivo de seguimiento del control [101].

En el problema de control punto a punto (regulación del sistema), los controladores basados en pasividad se pueden ver como casos particulares de la siguiente ley de control general:

$$\tau_e = -K_p e - K_d \dot{q} - u \tag{2.13}$$

| Controlador | u |
|---|---|
| PD+G | $-G(q)$ |
| PD+G0 | $-G(q_d)$ |
| PID | $K_i \int_0^t e\, dt$ |

**Tabla 2.2: Controladores por pasividad punto a punto**

donde $e = q - q_d$ varía dependiendo de la clase de controlador tal como se muestra en la tabla 2.2.

Además, en este trabajo se han implementado y probado también controladores de trayectoria basados en la pasividad. Al igual que ocurría para los controladores punto a punto, podemos encontrar en la literatura varios controladores. En este trabajo se ha optado por el propuesto en [102], en el que la energía cinética y potencial se modifica de forma adecuada para que el controlador sea global y asintóticamente estable. La expresión del controlador es la siguiente:

$$\tau_c = M(q)\ddot{q}_d + C(q,\dot{q})\dot{q}_d + G(q) - K_p e - K_d \dot{e} \qquad (2.14)$$

En la ecuación 2.14 se puede apreciar cómo el controlador tiene dos partes: una compensación de los términos dinámicos del robot y un controlador tipo proporcional-derivativo.

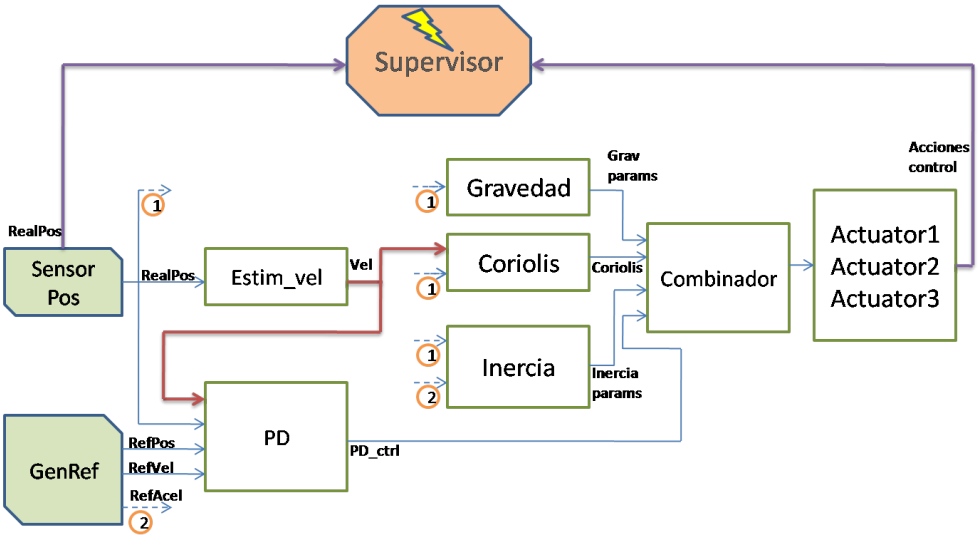



**Figura 2.4: Arquitectura de componentes de control del manipulador paralelo**

### 2.4.2. Entorno de desarrollo e implementación de los controladores

Para establecer el control y el desarrollo de aplicaciones con el manipulador paralelo, en este trabajo se ha utilizado la librería Orocos Toolchain con el sistema Linux Ubuntu, optándose por Xenomai para dotarlo de la funcionalidad de un sistema operativo de tiempo real.

Para la implementación de los controladores diseñados se planteó su especificación como componentes (ver figura 2.4). En primer lugar se tiene un componente *SensorPos* que accede en este caso a la tarjeta PCL-833 y que se utiliza para realizar la lectura de los valores de los encóders de las tres articulaciones del robot paralelo.

Los componentes *Actuador[1 | 2 | 3]* son los encargados de suministrar las acciones de control al robot, realizándose mediante la tarjeta de conver-

siones de Digital/Analógico PCI-1720 de Advantech. Para ello se tienen 3 instancias diferentes de un único componente, lo que permite en tiempo de ejecución seleccionar el canal de salida de la tarjeta y el puerto del componente por el que recibirá el valor de la acción de control a suministrar a la etapa de potencia del robot.

El componente *GenRef* es el encargado de generar los valores de referencia de movimiento correspondientes a la posición, velocidad y aceleración, las cuales se usarán como entrada al resto de módulos que componen el controlador.

Además, debido a que el robot no está equipado con sensores de velocidad, se ha tenido que desarrollar un componente (*Estim_vel*) encargado de realizar la estimación de la velocidad a partir de la posición real de las 3 articulaciones del robot.

Para la implementación de los algoritmos de control se han desarrollado una serie de componentes que se pueden combinar y reusar, permitiendo conformar distintas estrategias de control. Así, se ha implementado el componente *PD* que puede funcionar como un controlador pasivo punto-a-punto o formar parte de un controlador dinámico de trayectoria en donde los términos inerciales, gravitacionales y centrífugos y de Coriolis se calculan mediante los componentes *Inercia, Gravedad y Coriolis*. La selección de la estrategia de control punto-a-punto o trayectoria se realiza mediante el componente *Combinador*.

De forma adicional al controlador y los componentes encargados de la sensorización y actuación, se ha implementado un componente al cual se

le ha dado el nombre de *Supervisor* que es el encargado de almacenar el valor de las temporizaciones de cada uno de los componentes y el tiempo total de ejecución, así como las referencias y las posiciones reales del robot. Además, puede detectar de manera automática errores en la lectura de la posición del robot y/o errores en el sistema de actuación, de manera que si detecta alguna situación anómala, además de generar un aviso al operario humano, realiza una parada de emergencia mediante la desactivación de la etapa de potencia del robot paralelo.

Es importante destacar que OROCOS permite varias opciones para la temporización e interconexión de los puertos de las componentes. Con este *middleware* se puede asignar un periodo fijo a una determinada tarea, aunque también es posible despertar a una tarea mediante un puerto de evento.

En el esquema de control desarrollado se ha asignado un periodo de muestreo fijo de 10 ms al módulo de lectura de los encóders. El resto de módulos se ejecutan en cascada. De esta forma los módulos de control se ejecutan sólo cuando le llega el flujo de datos del módulo de encóders, y los 3 módulos de convertidores sólo se ejecutarán cuando el módulo combinador les envíe los valores calculados de las acciones de control.

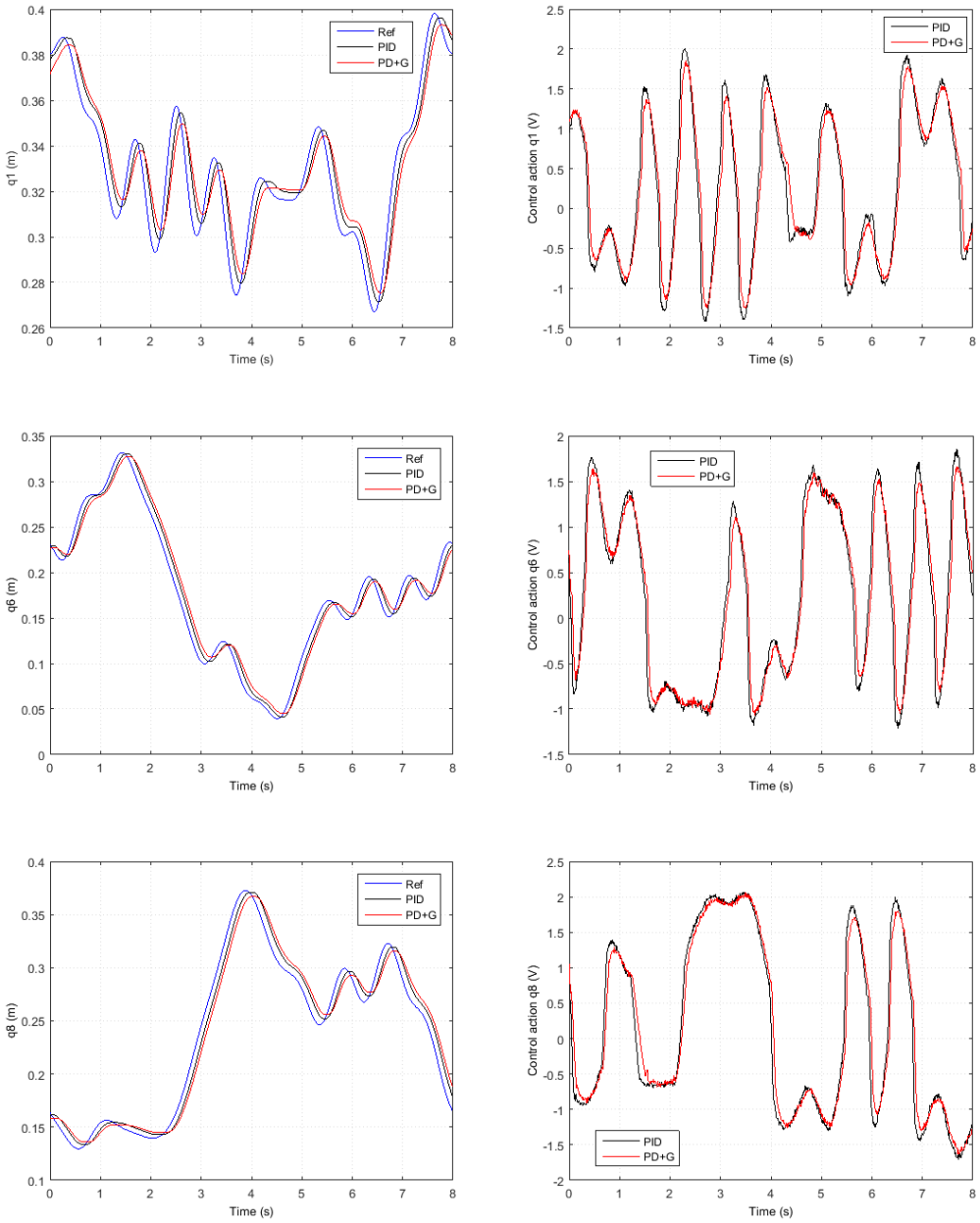
Por lo tanto, mediante este esquema en cascada, se está garantizando que un módulo sólo se ejecute cuando tenga todos los datos que necesita a través de sus puertos de entrada, estableciéndose el periodo de muestreo en el módulo que desencadena todo el proceso.

## 2.5. Resultados

A partir de la arquitectura hardware (robot/controlador) y software desarrollado, se han llevado a cabo diversas aplicaciones de control del manipulador paralelo. En las figuras siguientes se muestra el comportamiento del robot paralelo para distintas trayectorias y estrategias de control. Es necesario tener en cuenta que las referencias suministradas a la unidad de control del robot están en el espacio de la tarea, especificando así la orientación (*cabeceo y alabeo*) y la altura z de la plataforma. Sin embargo, el control se realiza en el espacio de las articulaciones ($q_1$,$q_6$ y $q_8$). Por ello el controlador calcula para cada iteración del bucle de control en tiempo real el problema cinemático inverso.

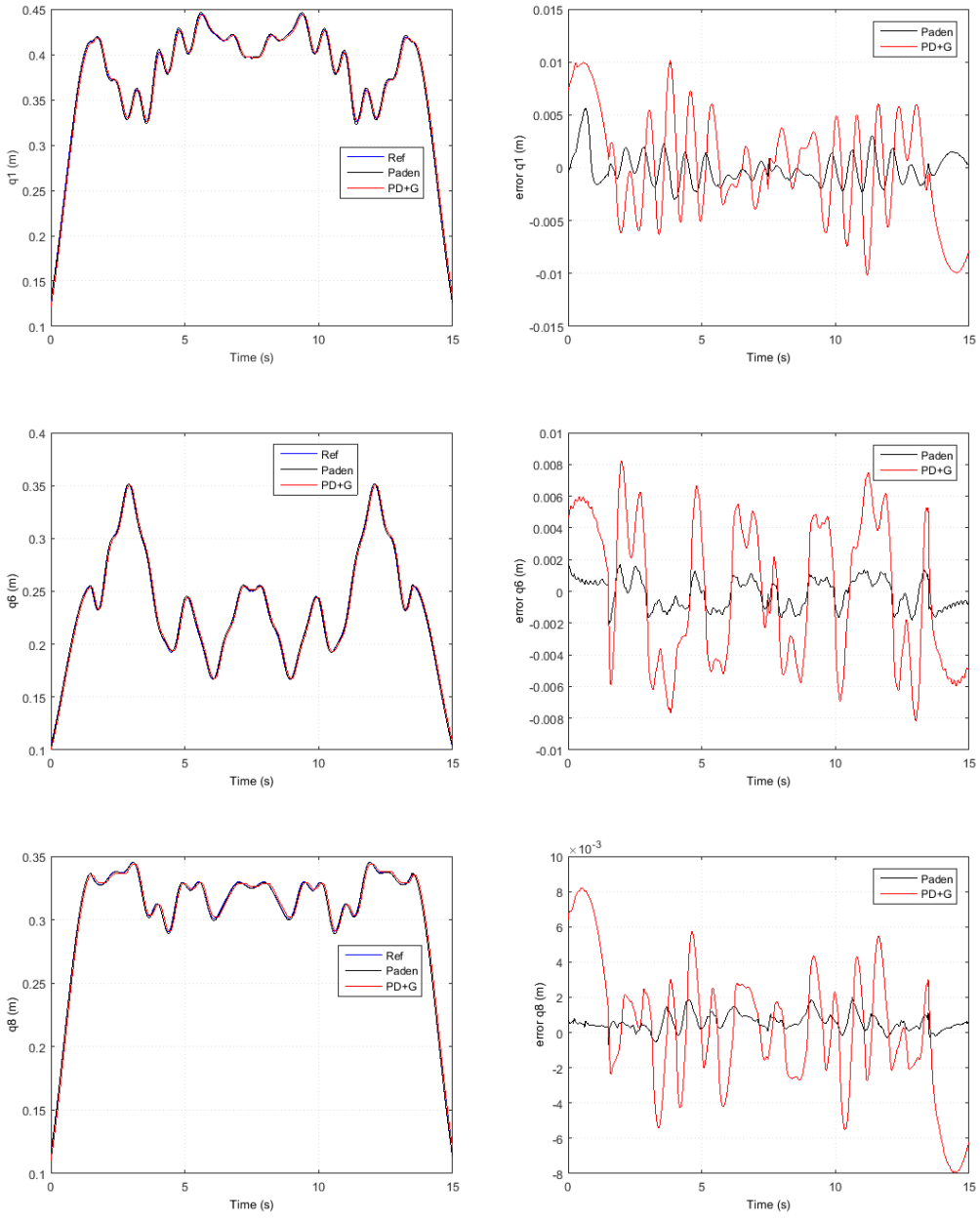
La figura 2.5 (a) muestra la respuesta de las 3 articulaciones del robot paralelo controlado por controladores punto a punto. En color azul se ha graficado las referencias de movimiento, en negro la respuesta de un PID y en rojo la respuesta del PD con la compensación de la gravedad. Se puede apreciar como en todos los casos la respuesta del robot es muy buena y sigue a la referencia de una forma muy precisa. La figura 2.5 (b) muestra las acciones de control de dichos controladores, en negro la acción de control de un PID y en rojo la del PD con la compensación de la gravedad.

La figura 2.6 compara la respuesta obtenida con un controlador pasivo punto a punto (PD con compensación de la gravedad, en color rojo) y el controlador de trayectoria pasivo de *Paden* (color negro). En la columna (a) se muestran las posiciones de las 3 articulaciones. La columna (b)




(a) posiciones q1, q6 y q8 (b) acciones de control

**Figura 2.5: Respuesta del robot paralelo con controladores punto a punto**




(a) posiciones q1, q6 y q8 (b) error qd - q

**Figura 2.6: Respuesta controlador punto a punto y trayectoria**

muestra los errores entre la referencia y la respuesta real del robot. Se puede comprobar fácilmente que los errores obtenidos con el controlador de trayectoria son mucho más pequeños que los obtenidos con el controlador punto a punto.

## 2.6. Conclusiones

Este artículo se ha propuesto una forma novedosa de implementar controladores en tiempo-real haciendo uso del *middleware* OROCOS y de una programación basada en componentes. Esto permite que las implementaciones desarrolladas y el intercambio de información sean independientes del hardware disponible, su desarrollo más homogéneo y reusables. Esta forma de implementar los controladores permite también el que, dependiendo de la situación en que se encuentre el sistema, sea posible lanzar a ejecución un determinado controlador u otro, reduciendo así la complejidad del código y haciéndolo más modular. Estas características ofrecen muchas ventajas de cara a trabajos futuros relacionados con cambios de controlador dependiendo del estado del sistema y de su entorno o en la distribución del cómputo dependiendo de la disponibilidad de recursos.

Para poder validar el desarrollo realizado se ha establecido el control de un manipulador paralelo 3-PRS. En este trabajo se han implementado diferentes controladores dinámicos basados en la pasividad, abordándose tanto el problema de control punto-a-punto como el control de trayectoria. En todos los casos se ha podido comprobar que la respuesta del robot ha sido muy buena.

# Capítulo 3

## Adaptive control of a 3-DOF parallel manipulator considering payload handling and relevant parameter models

El siguiente capítulo está basado en [103], un paper publicado en la Revista internacional *Robotics and Computer-Integrated Manufacturing*. Este paper expone la implementación de un control adaptativo, siguiendo el mismo principio de la ingeniería del software basada en componentes (reusando, además, controladores previamente desarrollados). Como se verá en el capítulo, el hecho de implementar un controlador adaptativo es de gran importancia cuando ciertos parámetros del modelo pueden cambiar con el tiempo.

**Abstract**

Model-based control improves robot performance provided that the dynamics parameters are estimated accurately. However, some of the model parameters change with time, e.g. friction parameters and unknown payload. Particularly, off-line identification approaches omit the payload estimation (due to practical reasons). Adaptive control copes with some of these structural uncertainties. Thus, this work implements an adaptive control scheme for a 3-DOF parallel manipulator. The controller relies on a novel relevant parameter dynamic model that permits to study the cases in where the uncertainties affect: (1) rigid body parameters, (2) friction parameters, (3) actuator dynamics, and (4) a combination of the former cases. The simulations and experiments verify the performance of the proposed controller. The control scheme is implemented on the modular programming environment Open Robot Control Software (OROCOS). Finally, an experimental setup evaluates the controller performance when the robot handles a payload.

## 3.1. Introduction

Researchers began to focus on parallel manipulators because of the advantages they hold over their serial counterparts: high stiffness, accuracy, speed, and payload handling. New architectures of PM have been proposed in academia while some of them has been implemented to real world applications e.g. medical applications [104, 105]. However, in or-

der to reach potential advantages over serial manipulator, PMs still require improvement in their design, modeling and control [106]. A vast literature deals with the kinematics and dynamic of a PMs, while the development of control schemes provides a field for improving a robot's performance [107] which is particularly in the interest of this work.

One of the most common control technique applied to PMs is the family of PID controllers which are mainly designed for trajectory tracking control (see [93, 108–110] among others) as well as the implementation of control strategies based on fuzzy logic [93]. Nowadays, the demand for fast and accurate robots points out the need for the implementation of model-based control [108]. For instance, computed torque control (CTC) linearizes and decouples the system dynamics through feedback loop compensation [107, 111–113]. However, CTC computes the dynamic model in real time, which increases the computational burden on the control system. On the other hand, Augmented PD control (APD) improves the tracking control by compensating the system dynamics which is computed off-line [114]. APD computes the dynamic model with the desired trajectory values ignoring the changes in the actual system state variables. CTC and APD require accurate values of dynamic model parameters, which are found through experimental parameter identification techniques [63]. Nevertheless, due to the topology of PMs – especially for lower-mobility PMs (less than 6 DOF) – some of the robot's links moves with poor excitation which affects the identification of the parameters [100]. In addition, the payload may be unknown and some of the system parameters may change (e.g. friction parameters) such that mo-

del based control loses performance.

Robust control deals with time variant parameters, thus, in [115] a nonlinear task, space control is applied to 6 DOF PM. However, the control design is based on practical implementation aspects discussed in [116]. Another approach for dealing with time-varying uncertainties is adaptive control. In [117], and adaptive control is developed, considering a linear feedback controller with a dynamic feed forward compensator, for a 6 DOF PM. The control approach identifies the parameters of the dynamic model through a nonlinear scheme. However, the authors used a simplified model without considering the identifiability of the dynamics parameters.

In [118, 119], an adaptive computed torque control and nonlinear adaptive control have been developed on a planar PM. The controllers are implemented in task space which requires extra sensors (the control signal and the measurements are normally applied in actuator space). Thus, task space coordinates were found from the actuated joint measurement through forward kinematics which increases the computational burden and could be impractical for a spatial PM.

The adaptive controllers applied thus far lack of the following consideration: (1) to adapt the model for only a subset of the parameters of a relevant parameters model and (2) to setup an experiment where the robot handles a payload while moving. Motivated by this consideration, in this chapter, an adaptive control scheme for a low-cost 3-DOF spatial PM is developed. The control scheme updates the model's parameters on line due to the implementation of a relevant parameter model [65] that can be

computed in real time [120]. The model includes 12 parameters: three rigid bodies, three actuator dynamics and six frictions parameters (in order to have a model linear in parameters, the friction in the actuated joints considers a Coulomb plus viscous friction model). Simulations and experiments evaluate four study cases of adaptive control, as regard to the subset of parameters identified online. The first case identifies on-line only the rigid body parameters. As will be seen in the modeling section, the rigid body parameters include the platform's mass and therefore this model compensates for an unknown payload. The second case explores friction parameters, the third considers actuator dynamics uncertainties and the fourth case combines the three former cases.

Different adaptive controls and a fixed passivity-based controller (PD+) [102] are implemented in the modular programming middleware OROCOS. The tracking trajectory performances of the adaptive controllers and the PD+ controller were compared through simulations and experiments. Moreover, the experiments section includes a novel setup where a payload is placed onto the platform while the robot is executing a task.

This chapter is organized as follows: Section 3.2 shows the PM design, while Section 3.3 deals with the model-based control schemes. Section 3.4 shows the results from the simulations and the experiments. Finally, Section 3.5 summarizes the conclusions.

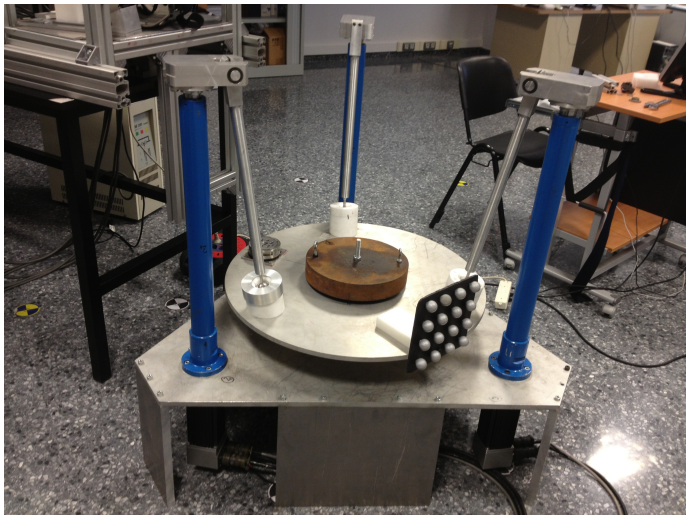


**Figure 3.1: Parallel robot used in the experiment**

## 3.2. The 3-DOF parallel manipulator

### 3.2.1. Physical description of the low-cost PM

As mentioned before, a 3-DOF spatial PM was used to address the controller design problem. The robot (see Fig. 3.1) consists of three kinematic chains, with each chain having a PRS configuration (P, R and S standing for prismatic, revolute and spherical joint, respectively). The underline format (P) stands for the actuated joint. The choice of the PRS configuration was guided by the need to develop a low-cost robotic platform with two DOF of angular rotation in two axes (rolling and pitching) and one DOF of translation motion (heave). In [121], a complete description of the mechatronic development process of the PM is presented.

The physical system consists of three legs connecting the moving platform to the base. Each leg consists of a direct drive ball screw (prismatic joints) and a coupler. The lower part of the coupler is connected with a

revolute joint to the ball screw, while the upper part is connected to the moving platform through a spherical joint. The lower part of the ball screws are perpendicularly attached to the platform's base and are positioned at the base in an equilateral triangular configuration. The ball screw transforms the rotational movement of the motor into linear motion.

The motors in each leg are brushless DC servomotors equipped with power amplifiers. The actuators are Aerotech BMS465 AH brushless servomotors. Aerotech BA10 power amplifiers operate the motors. The control system was developed on an industrial PC.

### 3.2.2. Kinematic and dynamic model

For modeling purposes, mobile reference systems have been attached to the robot links using Denavit–Hartenberg's notation; a detailed explanation is provided in [121]. As it has been explained in Section 2.3, nine generalized coordinates are used to model the robot ($q_i$, where $i = 1 \dots 9$), being the active ones $q_1$, $q_6$ and $q_8$ (associated with the actuated prismatic joints). The forward kinematics (Subsec 2.3.1) is solved using a geometric approach. From the rigid body link assumption, Fig. 2.2 shows that the length between the locations of the spherical joints $P_i$ and $P_j$ is constant and equal to $l_m$. Here, the position of the actuators is known ($q_1$, $q_6$ and $q_8$), with $q_2$, $q_7$ and $q_9$ as unknown. The Newton–Raphson (N–R) numerical method has been chosen to solve the nonlinear system, converging rather quickly when the initial guess is close to the desired solution [98].

Once those coordinates have been obtained, the location of points $P_i$ is acquired.

On the other hand, the inverse kinematics analysis consists of finding the actuated generalized coordinates given the roll, the pitch angles, and the heave of the reference system attached to the mobile platform. For the actual paralel robot, a detailed approach can be seen in Subsection 2.3.2.

One of the objectives of this work is to develop an open control architecture allowing the implementation and testing of dynamic control schemes. The dynamic controller requires that the equation of motion be described as follows:

$$M(\vec{q},\vec{\Phi})\cdot\ddot{\vec{q}} + C(\vec{q},\dot{\vec{q}},\vec{\Phi})\cdot\dot{\vec{q}} + \vec{G}(\vec{q},\vec{\Phi}) = \vec{\tau} \qquad (3.1)$$

In 3.1, $M$ represents the system mass matrix, $C$ is the matrix grouping the centrifugal and Coriolis terms, $\vec{G}$ is the vector corresponding to gravitational terms and $\vec{\tau}$ is the vector of generalized forces. It is worth noting that 3.1 is valid only when the system is modeled by a set of independent generalized coordinates. In this work, a coordinate partitioning procedure has been considered to model the system by a set of independent generalized coordinates. The actuated joint coordinates are the set chosen as the independent coordinates $\vec{q} = [q_1\ q_6\ q_8]^T$.

The dynamic model parameters are experimentally identified. This set of parameters not only includes the rigid body parameters, but also the rotor and screw dynamics of the robot actuators, as well as the friction

in actuated joints. The assumed Coulomb and viscous friction in the ith joint has been modeled as follows:

$$\tau_{f_i} = -(F_{c_i} \cdot sign(\dot{q}_i) + F_{v_i} \cdot \dot{q}_i) \tag{3.2}$$

In 3.2, $F_{c_i}$ and $F_{v_i}$ are the coefficients of the Coulomb and viscous friction respectively. Due to the characteristics of the actuators (ball screws), only friction in the actuated joint is considered. Eq. 3.1 can be rewritten (see [63, 65, 108]) as follows:

$$[K_{rb}\, K_r\, K_f] \cdot \begin{bmatrix} \vec{\Phi}_{rb} \\ \vec{\Phi}_r \\ \vec{\Phi}_f \end{bmatrix} = \vec{\tau} \tag{3.3}$$

In 3.3, the vectors $\vec{\Phi}_{rb}$, $\vec{\Phi}_r$ and $\vec{\Phi}_f$ group the rigid body, rotor and friction parameters. $K_i$ is the part of the coefficients matrix that determines the linear relationship between the corresponding parameters (rigid body, rotor and friction) and the generalized forces.

| $\vec{\Phi}_f$ | Base parameter | Value (SI units) |
|---|---|---|
| 1 | $F_{c_1}$ | 152.70 |
| 2 | $F_{v1}$ | 3267.45 |
| 3 | $F_{c_2}$ | 118.32 |
| 4 | $F_{v_2}$ | 2127.48 |
| 5 | $F_{c_3}$ | 247.97 |
| 6 | $F_{v_3}$ | 2120.09 |

**Table 3.1: Friction base parameters for the 3-PRS PM**

| $\vec{\Phi}_r$ | Base parameter | Value (SI units) |
|---|---|---|
| 1 | $J_1$ | 222.75 |
| 2 | $J_2$ | 222.75 |
| 3 | $J_3$ | 222.75 |

**Table 3.2:** Actuator base parameters for the 3-PRS PM

From equation 3.3, a set of base parameters corresponding to the complete base parameter model can be obtained (Tables 3.1-3.3). In Table 3.3, $l_m$ is the characteristic length of the mobile platform, $l_r$ is the length of the coupler links connecting the platform to the linear actuators, $m_i$, $mx_i$, $my_i$, $mz_i$, $Ixx_i$, $Ixy_i$, $Ixz_i$, $Iyy_i$, $Iyz_i$, and $Izz_i$ are the mass, the first and second moments of inertia of the $i$th link with regard to its local reference system.

The base parameter model cannot always be properly identified. For this reason, a reduced model containing the relevant parameters is obtained in this paper through a process that considers the robot's leg symmetries, the influence on the dynamic behavior of the robot the statistical significance of the identified parameters, and the physical feasibility of the parameters [65].

The relevant parameters model consists of the friction parameters from Table 3.1, the actuator parameters from Table 3.2 and the rigid body parameters 11, 17 and 18 from Table 3.3.

$$\Omega_1 = my_3 - sin(\frac{2}{3\pi})l_m \sum_{i=1}^{5} m_i \tag{3.4}$$

| $\vec{\Phi}_{rb}$ | Base parameter | Value (SI units) |
|---|---|---|
| 1 | $I_{zz2} - l_r^2 \sum_{i=1}^{2} m_i$ | -2.7068 |
| 2 | $mx_2 + l_r \sum_{i=1}^{2} m_i$ | -4.071 |
| 3 | $my_2$ | 0 |
| 4 | $I_{xx3} - (sin(\frac{2}{3\pi})l_m)^2 \sum_{i=1}^{5} m_i$ | -5.2431 |
| 5 | $I_{xy3} + cos(\frac{2}{3\pi})sin(\frac{2}{3\pi})l_m^2 \sum_{i=1}^{5} m_i$ | 2.9887 |
| 6 | $I_{xz3}$ | 0.1539 |
| 7 | $I_{yy3} - (cos(\frac{2}{3\pi})l_m)^2 \sum_{i=1}^{3} m_i + sin(\frac{2}{3\pi})l_m \sum_{i=4}^{5} m_i$ | 1.6483 |
| 8 | $I_{yz3}$ | 0 |
| 9 | $I_{zz3} - l_r^2 \sum_{i=1}^{3} m_i$ | -3.8104 |
| 10 | $mx_3 - cos(\frac{2}{3\pi})l_m \sum_{i=1}^{5} m_i + l_m \sum_{i=4}^{5} m_i$ | -0.0014 |
| 11 | $my_3 - sin(\frac{2}{3\pi})l_m \sum_{i=1}^{5} m_i$ | -11.568 |
| 12 | $mz_3$ | -0.5623 |
| 13 | $I_{zz5} - lr^2 \sum_{i=4}^{5} m_i$ | -2.6436 |
| 14 | $mx_5 - lr \sum_{i=4}^{5} m_i$ | -4.884 |
| 15 | $my_5$ | 0 |
| 16 | $I_{zz7} + lr^2 \sum_{i=1}^{5} m_i$ | 8.7408 |
| 17 | $\sum_{i=1}^{7} m_i$ | 38.8162 |
| 18 | $mx_7 + l_r \sum_{i=6}^{7} m_i$ | 16.3594 |
| 19 | $my_7$ | 0 |

**Table 3.3: Rigid body base parameters for the 3-PRS PM**

$$\Omega_2 = \sum_{i=1}^{7} m_i \tag{3.5}$$

$$\Omega_3 = mx_7 + l_r \sum_{i=6}^{7} m_i \tag{3.6}$$

In 3.4-3.6, $m_3$ is the mass of the platform. It can be seen that if a payload is placed in the center of the platform, its mass can be identified together with the mass of the platform.

## 3.3. Model-based position control schemes

Eq. 3.1 has several properties that can be exploited to facilitate dynamic controller designs. One of the most useful properties is that there is a re-parametrization of all unknown parameters into a parameter $[px1]$ vector $\vec{\Phi}$ that rewrites the system dynamics linearly in parameters. Therefore, the following holds:

$$M(\vec{q},\vec{\Phi})\cdot\ddot{\vec{q}} + C(\vec{q},\dot{\vec{q}},\vec{\Phi})\cdot\dot{\vec{q}} + \vec{G}(\vec{q},\vec{\Phi}) \equiv M_0(\vec{q})\cdot$$
$$\ddot{\vec{q}} + C_0(\vec{q},\dot{\vec{q}})\cdot\dot{\vec{q}} + \vec{G}_0(\vec{q}) + Y(\vec{q},\dot{\vec{q}},\ddot{\vec{q}})\cdot\vec{\theta} \tag{3.7}$$

The term in 3.7 for the actual 3-PRS parallel robot based on relevant parameters can be written as follows:

$$C(\vec{q},\dot{\vec{q}})\cdot\dot{\vec{q}} = \begin{bmatrix} F_{v_1}\dot{q}_1 + F_{c_1}sign(\dot{q}_1) \\ F_{v_2}\dot{q}_6 + F_{c_2}sign(\dot{q}_6) \\ F_{v_3}\dot{q}_8 + F_{c_3}sign(\dot{q}_8) \end{bmatrix} + \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{bmatrix} \tag{3.8}$$

$$M(\vec{q})\cdot\ddot{\vec{q}} = \begin{bmatrix} J_1 & 0 & 0 \\ 0 & J_2 & 0 \\ 0 & 0 & J_3 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_6 \\ \ddot{q}_8 \end{bmatrix} + \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix} \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{bmatrix} \tag{3.9}$$

$$G(\vec{q}) = g \begin{bmatrix} G_{11} & G_{12} & G_{13} \\ G_{21} & G_{22} & G_{23} \\ G_{31} & G_{32} & G_{33} \end{bmatrix} \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{bmatrix} \tag{3.10}$$

In addition, $M_0(.)$, $C_0(.)$ and $\vec{G}_0(.)$ represent the known part of the system dynamics, and $Y(\vec{q}, \dot{\vec{q}}, \ddot{\vec{q}})$ is a regressor matrix with dimensions $[nxp]$ containing nonlinear but known functions.

Having a dynamic model that is linear in parameters, the left hand side of 3.1 can be written as follows:

$$M_0(\vec{q})\cdot\ddot{\vec{q}} + C_0(\vec{q}, \dot{\vec{q}})\cdot\dot{\vec{q}} + \vec{G}_0(\vec{q}) + Y(\vec{q}, \dot{\vec{q}}, \ddot{\vec{q}})\cdot\vec{\theta} = \vec{\tau} \tag{3.11}$$

In the following, different adaptive scenarios are presented: (1) rigid body parameters, (2) friction parameters, (3) actuator dynamics, and (4) all the aforementioned.

### 3.3.1. Adaptive Model I

If the rigid body parameters constituting the reduced model are assumed to be unknown, then 3.11 can be written as follows:

$$\vec{\tau} = \begin{bmatrix} J_1 & 0 & 0 \\ 0 & J_2 & 0 \\ 0 & 0 & J_3 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_6 \\ \ddot{q}_8 \end{bmatrix} + \begin{bmatrix} F_{v_1}\dot{q}_1 + F_{c_1}sign(\dot{q}_1) \\ F_{v_2}\dot{q}_6 + F_{c_2}sign(\dot{q}_6) \\ F_{v_3}\dot{q}_8 + F_{c_3}sign(\dot{q}_8) \end{bmatrix} + Y_1(\vec{q}, \vec{\dot{q}}, \vec{\ddot{q}}) \cdot \vec{\theta_1} \quad (3.12)$$

where

$$Y(\vec{q}, \vec{\dot{q}}, \vec{\ddot{q}}) = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix} + \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} + g \begin{bmatrix} G_{11} & G_{12} & G_{13} \\ G_{21} & G_{22} & G_{23} \\ G_{31} & G_{32} & G_{33} \end{bmatrix} \quad (3.13)$$

$$\vec{\theta_1} = \begin{bmatrix} \Omega_1 & \Omega_2 & \Omega_3 \end{bmatrix}^T \quad (3.14)$$

### 3.3.2. Adaptive Model II

In this case Coulomb and viscous friction parameters are unknown, thus,

$$\vec{\tau} = \begin{bmatrix} J_1 & 0 & 0 \\ 0 & J_2 & 0 \\ 0 & 0 & J_3 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_6 \\ \ddot{q}_8 \end{bmatrix} + \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix} \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{bmatrix} +$$
$$+ \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{bmatrix} + g \begin{bmatrix} G_{11} & G_{12} & G_{13} \\ G_{21} & G_{22} & G_{23} \\ G_{31} & G_{32} & G_{33} \end{bmatrix} \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{bmatrix} \vec{\ddot{q}} + Y_2(\vec{q}, \vec{\dot{q}}) \cdot \vec{\theta_2} \tag{3.15}$$

where

$$Y_2(\vec{q}, \vec{\dot{q}}) = \begin{bmatrix} \dot{q}_1 & 0 & 0 & sign(\dot{q}_1) & 0 & 0 \\ 0 & \dot{q}_6 & 0 & 0 & sign(\dot{q}_6) & 0 \\ 0 & 0 & \dot{q}_8 & 0 & 0 & sign(\dot{q}_8) \end{bmatrix} \tag{3.16}$$

$$\vec{\theta_2} = \begin{bmatrix} F_{v_1} & F_{v_2} & F_{v_3} & F_{c_1} & F_{c_2} & F_{c_3} \end{bmatrix}^T \tag{3.17}$$

### 3.3.3. Adaptive Model III

When the parameters of the actuator dynamics are assumed to be unknown, the following equations hold:

$$\vec{\tau} = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix} \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{bmatrix} + \begin{bmatrix} F_{v_1}\dot{q}_1 + F_{c_1} sign(\dot{q}_1) \\ F_{v_2}\dot{q}_6 + F_{c_2} sign(\dot{q}_6) \\ F_{v_3}\dot{q}_8 + F_{c_3} sign(\dot{q}_8) \end{bmatrix} + \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{bmatrix} + g \begin{bmatrix} G_{11} & G_{12} & G_{13} \\ G_{21} & G_{22} & G_{23} \\ G_{31} & G_{32} & G_{33} \end{bmatrix} \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{bmatrix} + Y_3(\vec{\ddot{q}}) \cdot \vec{\theta_3} \tag{3.18}$$

where

$$Y_3(\vec{\ddot{q}}) = \begin{bmatrix} \ddot{q}_1 & 0 & 0 \\ 0 & \ddot{q}_6 & 0 \\ 0 & 0 & \ddot{q}_8 \end{bmatrix} \tag{3.19}$$

$$\vec{\theta_3} = \begin{bmatrix} J_1 & J_2 & J_3 \end{bmatrix}^T \tag{3.20}$$

### 3.3.4. Adaptive Model IV

In the same way, combinations where all the relevant parameters are unknown can be considered. For example, if all the dynamic parameters are unknown

$$\vec{\tau} = Y_4(\vec{q}, \vec{\dot{q}}, \vec{\ddot{q}})\vec{\theta}_4 \tag{3.21}$$

where

$$Y(\vec{q}, \vec{\dot{q}}, \vec{\ddot{q}}) = \begin{bmatrix} M_{11} & M_{12} & M_{13} & \ddot{q}_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ M_{21} & M_{22} & M_{23} & 0 & \ddot{q}_6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ M_{31} & M_{32} & M_{33} & 0 & 0 & \ddot{q}_8 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$
$$+ \begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & 0 & \ddot{q}_1 & 0 & 0 & sign(\dot{q}_1) & 0 & 0 \\ C_{21} & C_{22} & C_{23} & 0 & 0 & 0 & 0 & \ddot{q}_6 & 0 & 0 & sign(\dot{q}_6) & 0 \\ C_{31} & C_{32} & C_{33} & 0 & 0 & 0 & 0 & 0 & \ddot{q}_8 & 0 & 0 & sign(\dot{q}_8) \end{bmatrix}$$
$$+ g \begin{bmatrix} G_{11} & G_{12} & G_{13} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ G_{21} & G_{22} & G_{23} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ G_{31} & G_{32} & G_{33} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.22}$$

$$\vec{\theta}_4 = \begin{bmatrix} \Omega_1 & \Omega_2 & \Omega_3 & J_1 & J_2 & J_3 & F_{v_1} & F_{v_2} & F_{v_3} & F_{c_1} & F_{c_2} & F_{c_3} \end{bmatrix}^T \tag{3.23}$$

### 3.3.5. Control scheme

In recent years, the passivity-based approach to robot control has received a lot of attention. This approach solves the robot control problem by

exploiting the robot system's physical structure, and specifically its passivity property. The design philosophy of these controllers is to reshape the system's natural (kinetic and potential) energy in such a way that the control objective is achieved.

Bayard and Wen proposed a number of adaptive passivity-based control schemes that do not suffer from the parameter drift problem [122]. These authors have developed a class of adaptive controllers of robot motion, but in this work a different one has been developed for the parallel robot,

$$\tau_c = M_0(\vec{q})\ddot{\vec{q}}_d + C_0(\vec{q},\dot{\vec{q}}_d)\dot{\vec{q}}_d + \vec{G}_0(\vec{q}) + Y(\vec{q},\dot{\vec{q}}_d,\ddot{\vec{q}}_d)\hat{\vec{\theta}} - K_d\dot{\vec{e}} - K_p\vec{e} \tag{3.24}$$

$$\frac{d}{dt}\left\{\hat{\theta}(t)\right\} = -\Gamma_0 \cdot Y^T(\vec{q},\dot{\vec{q}}_d,\ddot{\vec{q}}_d)\cdot\vec{s}_1 \tag{3.25}$$

where $\vec{s}_1 = \dot{\vec{e}} + \Lambda_1\vec{e}$, with $\Lambda_1 = \lambda_1 I, \lambda_1 > 0$ and $M_0(.)$, $C_0(.)$ and $\vec{G}_0(.)$ represent the known part of the system dynamics, and $Y^T(.)$ is the regressor matrix.

The closed-loop system 3.1–3.24 and 3.25 is convergent; that is, the tracking error asymptotically converges to zero and all internal signals remain bounded under suitable conditions on the controller gains $K_p$ and $K_d$.

In order to compare and validate the adaptive controller, another

passivity-based controller has been implemented and tested. The PD+ controller proposed in [102] is implemented and can be written as follows:

$$\tau_c = M(\vec{q})\ddot{\vec{q}}_d + C(\vec{q},\dot{\vec{q}})\dot{\vec{q}}_d + \vec{G}(\vec{q}) - K_d\dot{\vec{e}} - K_p\vec{e} \qquad (3.26)$$

As in 3.1, $M(.)$, $C(.)$, $\vec{G}(.)$ represent the mass matrix, the centrifugal and Coriolis forces, and the gravitational forces of the robot's dynamic equation. This controller has been chosen because it has very good robust properties. In addition, since both are passivity-based controllers, their expressions are similar and therefore it is easy to compare and analyze their characteristics.

## 3.4. Results

In order to validate the performance of the adaptive control algorithm, first of all, several Matlab/Simulink schemes for the parallel robot simulation have been developed. The proposed controller is then implemented in an actual 3-DOF PM.

### 3.4.1. 3-PRS PM simulations

As seen in the previous section, the reduced PM model has 12 parameters: three rigid bodies, three actuator dynamics and six corresponding

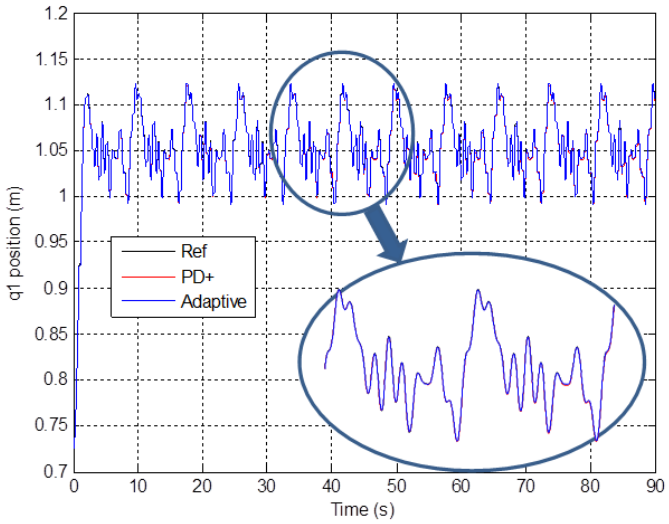



**Figure 3.2: Reference and simulated position for the first joint with a PD+ and an adaptive controller**

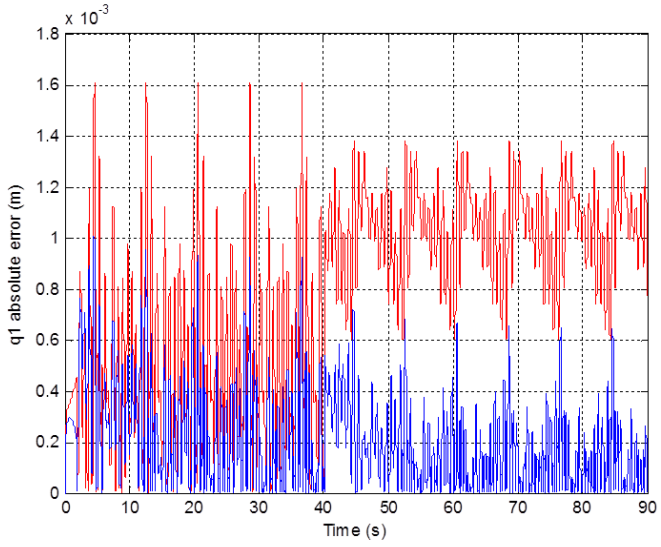



**Figure 3.3: Absolute error for the first joint with a PD+ and an adaptive controller**

to Coulomb and viscous friction. Depending on which parameters are assumed to be unknown, different adaptive schemes can be developed. In addition, the selection of the adaptive scheme also depends on which parameters are properly identified. This can be measured by the standard deviation of the identified dynamic parameters. A previous study shows that the rigid body parameters are those with a higher standard deviation in comparison with the other identified parameters [65]. Therefore, an adaptive controller based on the Adaptive Model I (Eqs. 3.12–3.14) has been simulated. This model only considers the rigid body parameters to be unknown.

Fig. 3.2 shows the reference and the simulated position for the first joint obtained with the adaptive controller. The control action is calculated by Eqs. 3.24 and 3.25 using the regressor matrix 3.13. In order to compare the control performance, the PD+ controller Eq. 3.26 has also been simulated. The simulations consider that a mass of 30 kg is placed on the mobile platform $m_3$ at $t = 40s$. The overall value of the mass changes from 12 kg to 42 kg.

As can readily be appreciated in Fig. 3.3, the error presented with both controllers before modifying $m_3$ is very similar. However, the PD+ controller response is much worse after modifying this mass on the mobile platform. This is because the PD+ controller uses unbiased values for its dynamic parameters. Given that the adaptive controller calculates an estimation of the rigid body parameters, it can take into account and compensate for changes in the platform mass.

In order to analyze the response of the different adaptive controllers con-

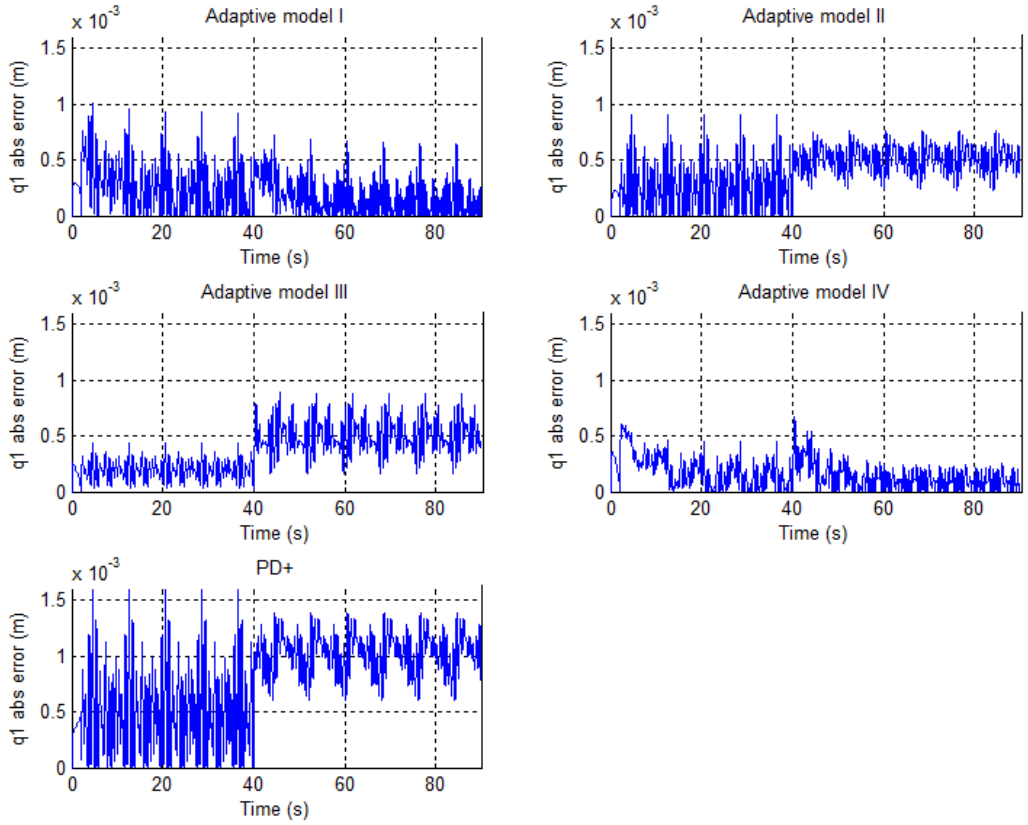



**Figure 3.4: Absolute first joint position error obtained for a reference in which a mass of 30 kg is added over the platform at the middle of the simulated execution, with Adaptive I–IV and PD+ models**

sidered, another test has been carried out. In this case, three different adaptive controllers have been simulated, which are based on Adaptive Model II (Eqs. 3.15–3.17), Adaptive Model III (Eqs. 3.18–3.20) and Adaptive Model IV (Eqs. 3.21–3.23). As in the previous case, for these simulations a mass of 30 kg has been added to the platform at $t = 40s$.

Fig. 3.4 illustrates the absolute error obtained for the two periods: before and after adding the 30 kg mass. In this figure, the error obtained with the Adaptive Model I, Adaptive Model II, Adaptive Model III, Adaptive

Model IV and PD+ model can be seen. As can be appreciated in Fig. 3.4, the controllers based on Adaptive Model II and Model III present a mean error that is twice that of the period where the load is added to the robot platform. This is because such controllers do not calculate an estimation of the parameters that change when the mass is added to the platform and thus their responses are very similar to the PD controller. However, given that the adaptive controller based on Model IV computes an estimation of the rigid body, actuator dynamics and friction parameters, it obtains a similar error before and after adding the mass to the platform. In addition, their response is virtually the same as the response of the adaptive controller based on Model I.

The selection of the adaptive model depends, among other factors, on which parameters are assumed to be unknown or may vary due to changes in the operating conditions. Clearly, the larger the number of unknown parameters, the greater the size of some of the constants of the adaptive controller. This may hamper the correct tuning of these constants. For this reason, in the case of variations in the mass added to the platform for example, control based on Adaptive Model I is recommended given that this is the simplest controller able to deal with the variation in parameters.

### 3.4.2. Experimental results on the actual 3-PRS PM

In addition to the simulation schemes, the PD+ and adaptive controllers described in this chapter have been implemented in a modular way, using

real-time middleware Open Robot Control Software (OROCOS). These controllers have been used and analyzed with the actual parallel robot presented in section 3.2.

OROCOS [51] is implemented entirely in C++ and, because it is a component-based middleware (being closely linked with a component-based software development), it allows the creation of modules that can operate in real time [39]. Furthermore, within the OROCOS environment there are libraries that are very useful for creating these components. In particular, one of the most relevant is the "OROCOS Toolchain", which is very important as it includes the "Real Time Toolkit (RTT)" and the "OROCOS Component Library (OCL)". The first one deals with everything related to the real-time execution of components, as well as the connection between them, while the second provides the basic primitives for building components. Therefore, through the component-based software development that provides the OROCOS environment, the following advantages can be seen [39]:

1. Through the modular design, the program execution flow can very easily be monitored, facilitating the creation of new components and their insertion into the model to obtain new features.

2. The modular structure allows the execution of multiple modules in a distributed manner, obtaining a lower running time than if they were running serially.

3. The code is fully reusable, which allows an unlimited number of examples to be created for each module.

4. Once loaded, modules are configurable and reconfigurable both in setup time and in running time, being able not only to change specific parameters for each module, but also general parameters such as the execution priority, etc.

Thus, when a number of modules are implemented and a control scheme is required, it is as simple as inserting the necessary modules to configure them, making connections with each other, and making them run. Therefore, given that the different control schemes have common parts due to the development of several modules; these modules are reused to implement different controllers.

Finally, note that although the development of component-based software can be a complicated task at first, in the long run it facilitates the programmer's work because if a module works correctly in one particular scheme, it will certainly work correctly in another control scheme. Therefore, as well as the advantages discussed above, this approach minimizes the chance of possible programming errors in the implementation of any module.

Fig. 3.5 shows the OROCOS diagram for the adaptive controller implemented based on Adaptive Model I. The *SensorPos* module provides the three joint positions of the robot using the Advantech's PCL-833 encoders card. *Gener_Refe* module calculates the movement references for the robot joints. *PD* module implements a proportional-derivative type controller. *Coriolis* and *Inertia* modules calculate the robot's fixed dynamic terms of the equation of motion (Eq. 3.12). $Y$ module calculates the regressor

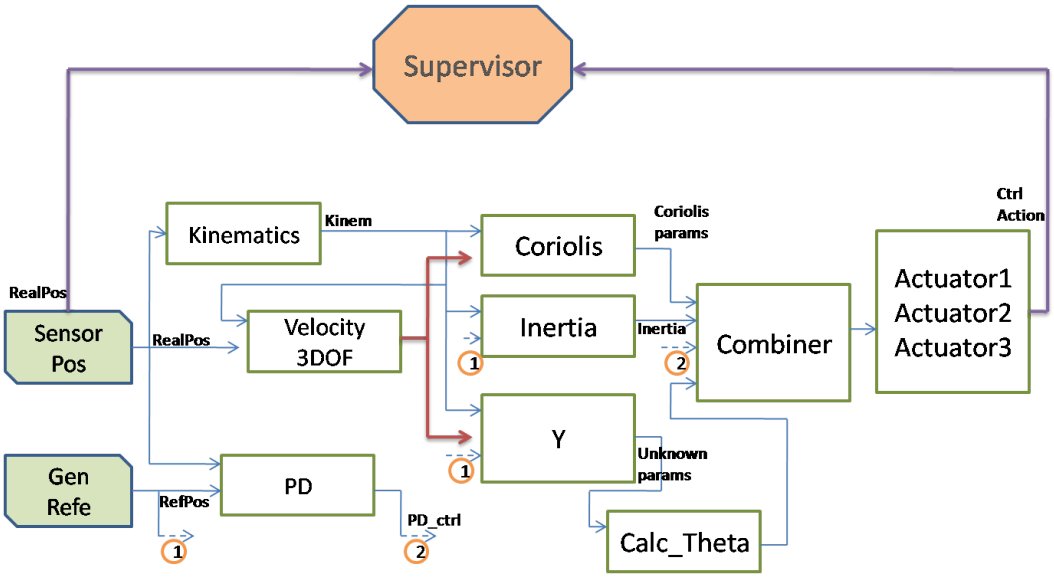



**Figure 3.5: Component diagram implemented in order to perform an adaptive controller using the middleware Open Robot Control Software (OROCOS)**

matrix of Eq. 3.13. The *Calc_theta* module calculates the parameter estimation of Eq. 3.14. The *combiner* module calculates the control action depending on the control strategy selected. Finally, the *actuator* modules are responsible for carrying out digital-analog conversions through Advantech's PCI-1720 cards. The control scheme also provides the *supervisor* module, which is responsible for monitoring the correct operation of the system. This module is in charge of deactivating the control unit and stops the robot in the case of detecting a malfunction in the system. Due to the distributed execution of the control, the sample time in this experiment has been 10 ms.

Fig. 3.6 illustrates the reference and the robot $q_1$ position with PD+ and adaptive controllers. The actual robot motion reference is the same as the reference used in simulation. The only difference is that in the middle of

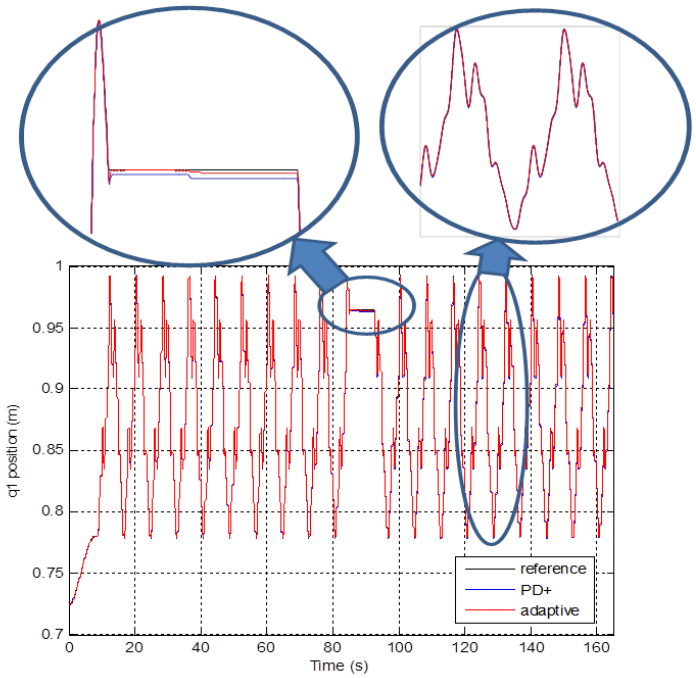



**Figure 3.6: Reference and real position for the first joint of the parallel robot for both adaptive and PD+ controllers**

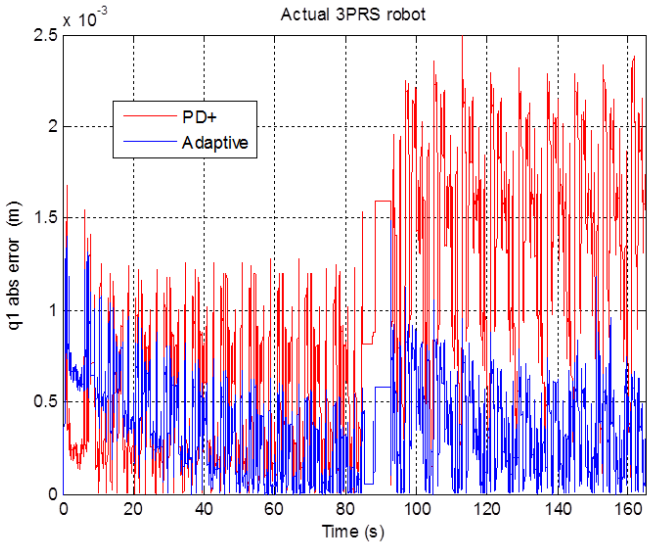



**Figure 3.7: Absolute first position error, obtained with the real robot, for a reference in which a mass of 30 kg is added over the platform at the middle of the real execution, with adaptive and PD+ controllers**

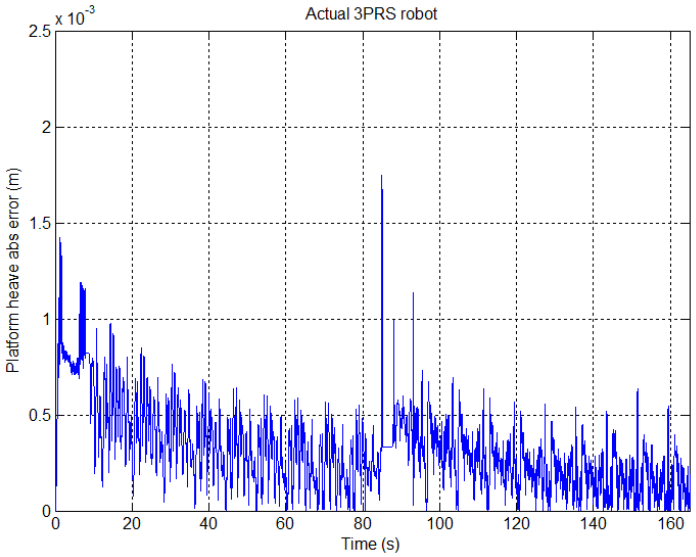



**Figure 3.8: Absolute heave position error, obtained with the real robot, with an adaptive controller**

execution, the robot remains in the same position for 8 s (between $t = 85$ and $t = 93$ s). This time allows a load of 30 kg to be placed onto the robot platform. Fig. 3.7 illustrates the absolute error values, and the absolute position error for the platform heave is shown in Fig. 3.8. The control action provided by the adaptive controller is shown in Fig. 3.9. Finally, Fig. 3.10 shows the real robot with the added mass.

The results obtained with the actual robot are consistent with those obtained in the simulation: because of the estimation of the online dynamic parameters, the change in the load means that robot response using an adaptive controller is significantly better than that obtained with the PD+ controller.

Table 3.4 demonstrates the absolute mean error and the mean square root error (RMS) between the references and the measured positions of the parallel robot for the two periods (before and after changing the robot

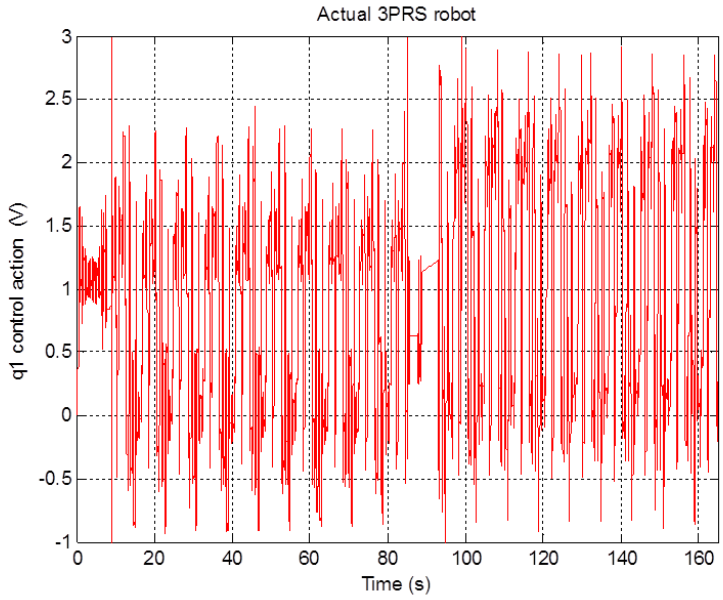



**Figure 3.9: Control action applied to the robot provided by the adaptive controller**

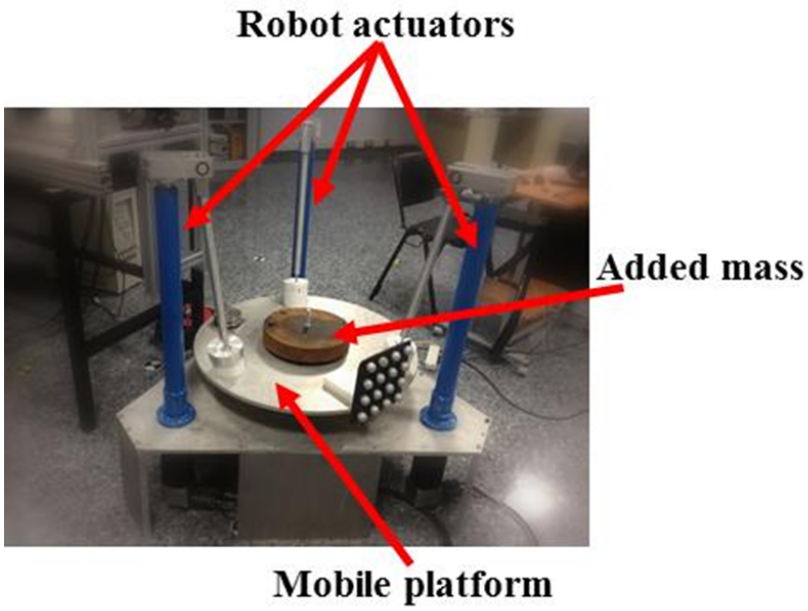



**Figure 3.10: Parallel robot used in this experiment, with the 30 kg added mass over the platform**

| Period | $\frac{\sum_{j=i}^{n}\sum_{i=1}^{DOF} abs(e_{i,j})}{nDOF}$ | | $\sqrt{\frac{\sum_{j=i}^{n}\sum_{i=1}^{DOF} abs(e_{i,j})^2}{nDOF}}$ | |
|---|---|---|---|---|
| | PD+ C. | Adaptive C. | PD+ C. | Adaptive C. |
| 1 | 0.000504 | 0.000337 | 0.000622 | 0.000412 |
| 2 | 0.001432 | 0.000378 | 0.001541 | 0.000446 |

**Table 3.4: Error values (m)**

load using a mass of 30 kg). As can readily be appreciated, the adaptive controller provides a better response in comparison with the PD+ controller. In fact, for the second period, the error obtained with the PD+ controller has an increment of about 184% greater than the one obtained with adaptive control.

## 3.5. Conclusion

Adaptive control was developed for the trajectory tracking control of a 3-DOF parallel manipulator. The controller took advantages of a simplified relevant parameters model that is cost-effective real-time. Four study cases of adaptive control were evaluated considering the on-line identification of: the rigid body parameters, the friction parameters, the actuator dynamics, and a combination of the former parameters. A novel experimental setup was proposed that evaluates the performance of the adaptive controller: a payload is placed onto the platform while the robot is executing a task. The adaptive control and a fixed PD+ control scheme were implemented modularity in OROCOS environment. Simulations and experiments showed that adaptive control outperform, as regard to trajectory tracking accuracy, the PD+ control when a payload is added

onto the platform. The on-line update of the relevant parameters takes into account for an unknown payload and changes in friction coefficients, thus, overcoming the limitation of the PD+ that relies on fixed parameters.

# Capítulo 4

## Hybrid force/position control for a 3-DOF 1T2R parallel robot: Implementation, simulations and experiments

ESTE capítulo está basado en [123], el cuál es una versión extendida de [124] (artículo presentado en el *International Symposium on Multibody Systems and Mechatronics*), siendo galardonado por el Comité Científico del MUSME, con el reconociemtno de ser publicado en la Edición Especial del *MUSME 2014* en el *International Journal Mechanics Based Design Of Structures And Machines*. La novedad de este paper es que, gracias al sensor de fuerza instalado en el sistema, se consigue diseñar, implementar y testear un controlador híbrido fuerza/posición para un robot paralelo de tres grados de libertad.

**Abstract**

A robot interacting with the environment requires that the end effector position is tracked and that the forces of contact are kept below certain reference values. For instance, in a rehabilitation session using a robotic device, the contact forces are limited by the allowed strength of the human limbs and their complex-joints. In these cases, a control scheme which considers both position and force control is essential to avoid damage to either the end effector or the object interacting with the robot. This chapter therefore develops a real-time force/position control scheme for a three-DOF parallel robot whose end effector holds a DOF one translation (1 T) and two rotations (2 R). The implemented hybrid force/position control considers, as a reference, the normal force on the mobile platform, which is measured by means of a load cell installed on the platform. The position control is designed to track the orientations of the robot either in joint or task space using a model-based control scheme with identified parameters. Moreover, the force control is based on a PD action. The control scheme is developed through simulations, before being applied to an actual parallel robot. The findings show that with the implemented controller, the actual robot accomplishes the reference values for the normal force on the mobile platform, while at the same time the platform accurately follows the required angular orientation.

## 4.1. Introduction

The inherent capabilities of a parallel robot (PR), compared with its counterpart a serial robot, have made it an interesting research topic in the robotic community. These capabilities can be summed up as follows: high stiffness, high load-carrying footprint ratio, high speed and accuracy. In these robots the end effector is attached to a mobile platform which is connected by several kinematic chains (also known as legs) to a fixed based. The advantages over serial robots stem from the fact that the load on the end effector can be shared by the legs. The similarity with the human body is that when we are unable to carry a load precisely with one arm, we use both arms, thus giving us greater precision and stiffness, although we lose workspace. The forward kinematics, dynamics model, and control of PMs are also cumbersome. Given that in certain cases the advantages outweigh the disadvantages, researchers have developed several applications based on PRs, such as motion simulators, tire testing machines, flight simulators, and medical applications. Different mechanical architectures of PRs can be found in the following references [55, 56, 90, 106].

Seminal papers on PRs have focused on platforms with 6 degrees of freedom (DOF): 3 translational (3T) and 3 rotational (3R), an idea that has continued to be developed with new 6-DOF architectures nowadays [125]. However, many applications require less than 6 DOF. Actually, one of the fastest PR architectures is based on 3T1R motion [87]. Also, pick-and-place tasks can be accomplished with 3 translational DOF (3T), the best example of which is the Delta Robot [88]. In additional, medical appli-

cations such as cardiopulmonary resuscitation equipment are accomplished with 3T PRs [89], as are tool heads for manufacturing facilities [90]. Other developments with 3T1R motion can be found in [126]. For low-cost applications designers often search for robot architectures with fewer DOF. One such application has 1T2R motions; in this case, the 3-PRS [127] and the 3-RPS [97] architectures have been proposed. Note that R, P and S stand for revolution, prismatic and spherical joint at each leg, respectively.

On the other hand, independently of the robot's DOF, the trajectory at the end-effector is controlled using position control: point-to-point or tracking control, joint or task space. However, a variety of applications require the forces being applied to the end-effector to be kept below certain reference values. This is the case, for instance, with robots involving assembly tasks where one or more parts need to be handled, ensuring proper contact, or industrial operations such as milling or deburring [128]. New developments in parallel robots for medical applications involve interaction with delicate parts of the human body where the forces at the end-effector must be applied carefully, for instance, robots for in vivo biopsy [129]. For such applications it is necessary to develop an accurate robot force control scheme, but while different force control approaches have been implemented for serial robots [130], the issue of force/position control of PRs has barely been addressed in the robotics literature [131].

In a previous work, the authors developed a low-cost PR with 3-PRS with one translation (1T) and two rotations (2R) DOF of its end effector [121]. The robot has the advantage that the control architecture is

open, which has allowed model-based control schemes to be implemented and tested [120]. One interesting application of 1T2R robots is an ankle rehabilitation device, but in a rehabilitation session using a robotic device the contact forces are limited by the allowed strength of the human limbs and joints undergoing rehabilitation. The aim of this work is therefore to develop a real-time hybrid force/position control such that the normal force on the mobile platform follows a reference signal while at the same time tracking the orientation of the platform. In order to implement the control scheme, the force is measured by means of a load cell installed on the end- effector, and the force controller part is based on a PD action. The position control is designed to track the orientations of the robot either in joint or task space considering a model based control scheme [132], using an identified model [133]. Simulations conducted in a Matlab/Simulink environment have made it possible to develop the control scheme which is then applied to the actual PR.

The chapter is organized as follows: Section 4.2 briefly describes the test bed PR and develops the kinematic and dynamic models. Section 4.3 discusses the different control schemes using a simulated robot. Section 4.4 shows the results of the force/position control when it is applied to the actual prototype, and Section 4.5 presents the conclusions.

## 4.2. The test bed robot

The choice of the test bed robot architecture was determined by the need to develop a low-cost robot which is capable of generating angular rota-

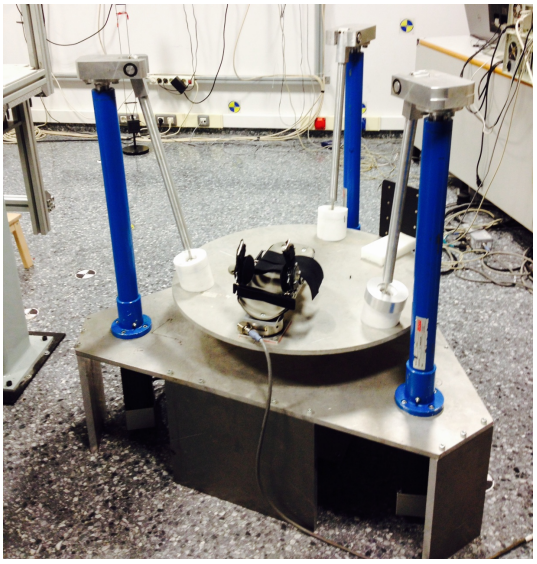


**Figure 4.1: The test bed low-cost parallel robot**

tion on two axes, as well as a prismatic movement. Two different architectures were considered: three-RPS and three-PRS. The second was chosen after comparing the advantages and disadvantages of each proposal. For example, one advantage of the PRS architecture is that the actuators are located on the fixed base, while in the RPS architecture the actuators move with revolute joints.

Figure 4.1 shows the actual prototype. The physical system consists of three legs connecting the moving platform to the base. Each leg consists of a direct-drive ball screw (prismatic joints) and a coupler, as well as the motor.

The motors in each leg are brushless DC servomotors equipped with power amplifiers. The actuators are Aerotech BMS465 AH brushless servomotors. Aerotech BA10 power amplifiers operate the motors. The control system was developed on an industrial PC, which has several advantages:

the first is that it is a completely open system, providing a powerful platform in which to program high-level tasks. Therefore, applications such as trajectory generation or control strategies based on external sensing (machine vision and force sensors, etc.) can be developed. The second significant advantage of the control system is its cost, given that the software architecture is based on free and open software.

### 4.2.1. Kinematic model

Robot force or position control algorithms involve establishing the direct and inverse kinematic models. The inverse kinematic model is used to obtain the motion at the actuated joints corresponding to the motions required at the end effector. On the other hand, since the actuated joints are usually the variables that can be measured through incremental encoders, the direct kinematic model is needed to determine the actual motion at the end effector.

The modified Denavit–Hartenberg (D–H) convention is used for modeling purpose. The robot is cut open at the spherical joints shown in Fig. 4.2 as $P_2$ and $P_3$, thus obtaining three legs. One leg consists of three links: a sliding rod, a coupler, and the platform (coordinates $q_1 \cdot q_5$), and the two other legs consist of two links: a sliding rod and a coupler (coordinates $q_6 \cdot q_7$ and $q_8 \cdot q_9$). Numbers 1, 4, and 6 in the figure apply to the sliding rod which is attached to the coupler (numbers 2, 5, and 7) by a revolute joint (R), and to the base by an actuated prismatic joint (P). The active coordinates $q_1$, $q_6$, and $q_8$ are associated with the actuated prismatic joints,

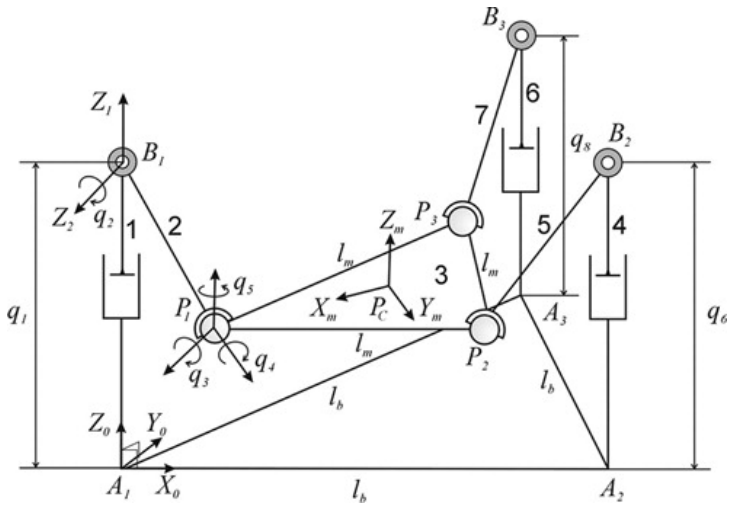



**Figure 4.2: Kinematic diagram of the three-PRS parallel manipulator, type of joints, and generalized coordinates**

while the passive coordinates $q_2$, $q_7$, and $q_9$ are associated with the revolute joints. Coordinates $q_3$, $q_4$, and $q_5$ are associated with the spherical joint located at $P_1$. Note that the spherical joint is modeled, according to the D–H parameters, as three orthogonal axes intersecting at the center of an ideal sphere (representing the spherical joint).

In the forward position problem, the position of the actuators is known ($q_1$, $q_6$, and $q_8$ are known) but the remaining coordinates need to be found, finally obtaining the roll ($\gamma$), pitch ($\beta$), and height ($z$). In contrast, inverse kinematics problem consists of finding the movement of the linear actuators from the roll ($\gamma$), pitch ($\beta$), and height ($z$) of the platform. These mathematical models can be shown in detail in Chapter 2.

### 4.2.2. Jacobian matrix

The kinematic model of a robot seeks relationships between end-effector coordinates and the actuated joint position. The relationship between the velocities of the joint coordinates and the position and orientation of the robot is obtained through the Jacobian matrix. Furthermore, transposing this matrix provides the relationship between the external forces applied to the end effector (often called the wrench) and the actuated joint forces. Thus, the matrix is useful for both the velocity and the force problem. In order to obtain the Jacobian, the following vector expression can be obtained from Fig. 4.2:

$$\vec{r}_{OP_c} + \vec{r}_{P_cP_i} = \vec{r}_{OA_i} + q_1 \cdot \vec{r}_{li} + l_r \cdot \vec{r}_{2i} \tag{4.1}$$

where $\vec{r}_{li}$ and $\vec{r}_{2i}$ are unit vectors in the direction of $A_iB_i$ and $B_iP_i$, respectively, $l_r$ is the length of the coupler, $\vec{r}_{OP_c}$ is the position vector from the global coordinate system to the end-effector location, and $\vec{r}_{P_cP_i}$ is the position vector between the end effector and the spherical joint $P_i$. Finally, $\vec{r}_{OA_i}$ defines the position vector between the intersection of the line defining the prismatic joint motion with the base plane and the global coordinate system.

Deriving Eq. 4.1 and multiplying both sides of the equation by $\vec{r}_{2,i}$:

$$\vec{r}_{2,i} \cdot \vec{v}_{P_c} + \vec{r}_{2i} \cdot (\vec{\omega}_{P_C} \times \vec{r}_{P_CP_i}) = \vec{r}_{2,i} \cdot \vec{r}_{1,i} \dot{q}_i \tag{4.2}$$

where $\vec{v}_{P_c} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T$ contains the components of the linear velocity at the end effector and $\vec{\omega}_{P_C} = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T$ contains the components of the angular velocity.

Equation 4.2 can be applied to each leg. Defining $\vec{b}_i = \vec{r}_{P_C P_i}$ the following equations hold:

$$\begin{bmatrix} \vec{r}_{21} \cdot \vec{r}_{11} & 0 & 0 \\ 0 & \vec{r}_{22} \cdot \vec{r}_{12} & 0 \\ 0 & 0 & \vec{r}_{23} \cdot \vec{r}_{13} \end{bmatrix} \cdot \begin{bmatrix} \dot{q}_1 \\ \dot{q}_6 \\ \dot{q}_8 \end{bmatrix} = \begin{bmatrix} \vec{r}_{21}^T (\vec{b}_1 \times \vec{r}_{21})^T \\ \vec{r}_{22}^T (\vec{b}_2 \times \vec{r}_{22})^T \\ \vec{r}_{23}^T (\vec{b}_3 \times \vec{r}_{23})^T \end{bmatrix} \cdot \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{4.3}$$

We can therefore verify that

$$J_q \cdot \dot{\vec{q}} = J_x \cdot \dot{\vec{x}}_{P_c} \tag{4.4}$$

However, it is important to bear in mind that not all the variables of the $\dot{\vec{x}}_{P_c}$ vector are independent given that only three of the coordinates at the end effector are independent. Consequently, the relationship between the end-effector coordinates has to be determined. The revolute joints, at points $B_i$, impose the constraint that each leg moves on a plane whose

normal vector corresponds to the axis of the revolute joint. Taking this into account, the following set of equations can therefore be written:

$$x = -\frac{l_m}{\sqrt{3}} sin(\alpha) cos(\beta) \tag{4.5}$$

$$y = -\frac{2l_m}{\sqrt{3}} (cos(\alpha)cos(\beta) - sin(\alpha)sin(\beta)sin(\gamma) - cos(\alpha)cos(\gamma)) \tag{4.6}$$

$$tan(\alpha) = \frac{sin(\gamma)cos(\beta)}{cos(\gamma) + cos(\beta)} \tag{4.7}$$

The partial derivatives of Eqs. 4.5–4.7 provide the 6 × 3 Jacobian matrix relating the dependent end-effector coordinates to the independent ones. That is

$$\begin{bmatrix} \dot{x} & \dot{y} & \dot{z} & \dot{\alpha} & \dot{\beta} & \dot{\gamma} \end{bmatrix}^T = J_r \cdot \begin{bmatrix} \dot{z} & \dot{\beta} & \dot{\gamma} \end{bmatrix} \tag{4.8}$$

where

$$J_r = \begin{bmatrix} \frac{\delta x}{\delta z} & \frac{\delta x}{\delta \gamma} & \frac{\delta x}{\delta \beta} \\ \frac{\delta y}{\delta z} & \frac{\delta y}{\delta \gamma} & \frac{\delta y}{\delta \beta} \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \frac{\delta \alpha}{\delta z} & \frac{\delta \alpha}{\delta \gamma} & \frac{\delta \alpha}{\delta \beta} \end{bmatrix} \tag{4.9}$$

The above equations provide the relationship, including the change in platform angle over time. Nevertheless, the Jacobian matrix needed by the wrench has to be written with respect to the angular velocity at the end effector. Based on the fact that the angular velocity is related to the change in the angles over time, this matrix can be obtained by the following equations:

$$\begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T = \begin{bmatrix} cos(\alpha)cos(\beta) & -sin(\alpha) & 0 \\ sin(\alpha)cos(\beta) & cos(\alpha) & 0 \\ -sin(\beta) & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \dot{\alpha} & \dot{\beta} & \dot{\gamma} \end{bmatrix}^T \tag{4.10}$$

Thus, from Eqs. 4.4, 4.8, and 4.10, the following relationship can be written as follows:

$$\dot{\vec{q}} = J_q^{-1} \cdot J_x \cdot J_r \cdot \begin{bmatrix} \dot{z} & \omega_x & \omega_y \end{bmatrix}^T = J_p \cdot \begin{bmatrix} \dot{z} & \omega_x & \omega_y \end{bmatrix}^T \tag{4.11}$$

where $J_p$ is the Jacobian matrix relating the coordinates of the end effector to the actuated joint space. The equation relating the wrench at the end effector with the actuation forces can be written as follows:

$$\vec{\tau} = J_q^T \cdot \begin{bmatrix} F_z & N_x & N_y \end{bmatrix}^T \tag{4.12}$$

Equation 4.12 can be used to find the required actuation forces caused by the external force applied to the end effector, which is useful for developing force control.

### 4.2.3. Dynamic model

In order to implement dynamic controllers, the equation of motion of the parallel robot can be described as follows:

$$M(\vec{q},\vec{\Phi})\cdot\ddot{\vec{q}} + \vec{C}(\vec{q},\dot{\vec{q}},\vec{\Phi})\cdot\dot{\vec{q}} + \vec{G}(\vec{q},\vec{\Phi}) = \vec{\tau} \tag{4.13}$$

From Eq. 4.13, it can be seen that the system mass matrix $M$, the vectors corresponding to the centrifugal and Coriolis forces $\vec{C}$, and the gravitational forces $\vec{G}$ depend on the dynamic parameters $\vec{\Phi}$ and the external generalized forces $\vec{\tau}$. In order to identify the dynamic parameters, the model needs to be written in linear parameters [65]:

$$K(\vec{q},\dot{\vec{q}},\ddot{\vec{q}})\cdot\vec{\Phi} = \vec{\tau} \tag{4.14}$$

Equation 4.14 represents a set of three linear equations, thus $K(\vec{q},\dot{\vec{q}},\ddot{\vec{q}})$ is a 3 × 25 matrix which is a function of the positions, velocities, and accelerations of the generalized coordinates. In addition, $\vec{\Phi}$ is the 25 × 1 vector of dynamic parameters. As can be seen, the systems of equation hold a number of equation which is less than the unknown parameters. Thus, for identifying the dynamic parameters, Eq. 4.14 has to be applied to a set of several robot states so that an over-determined system is built up, which can be solved using Least-Square Techniques. The states of the robot which are needed to build the over determined system are obtained by measuring the position and forces of the robot when it follows a prescribed trajectory. The trajectory is designed so that all the parameters in

the model contribute to the actuation torques [65]. However, even with a well-designed trajectory, some of the dynamic parameters are irrelevant in terms of their contribution to the actuation torques. In this project, only those relevant parameters which contribute to the dynamics model are identified and used for model-based control. This approach was used in a previous work, of which the procedure followed and the dynamics parameters can be seen in [120].

The gravitational term $\vec{G}$ is obtained, taking into account that it depends only on the generalized coordinates. Thus, by zeroing the generalized velocities and accelerations, the following expression is obtained:

$$K(\vec{q}, \dot{\vec{q}} = 0, \ddot{\vec{q}} = 0) \cdot \vec{\Phi} = \vec{G}(\vec{q}) \tag{4.15}$$

The columns in the system mass matrix can be determined as follows:

$$K(\vec{q}, \dot{\vec{q}} = 0, \vec{e}_i) \cdot \vec{\Phi} - \vec{G}(\vec{q}) = M_i(\vec{q}) \tag{4.16}$$

In the last equation, $M_i(\vec{q})$ is the *i*-ith column of the mass matrix and $\vec{e}_i = \begin{bmatrix} 0 & \dots & 1 & \dots & 0 \end{bmatrix}^T$ a column vector with 1 in the *i*-ith position.

Finally, by once again zeroing the generalized accelerations in Eq. 4.14, the centrifugal and Coriolis terms (which depend on the generalized coordinates and velocities) can be obtained.

## 4.3. Development of the simulated parallel robot

### 4.3.1. Position joint space control

After the dynamic problem has been solved and the dynamic parameters have been obtained and validated through parameter identification of the parallel robot, the real-time control can be addressed. In this work, various joint space control strategies based on inverse dynamics have been implemented. This type of control is discussed in more detail in [134–137]. These strategies are potentially very useful given that they reduce the nonlinear control problem to the control problem of a linear system, for which many tools are available. Assuming the dynamic model as

$$\vec{x}^{(n)} = f(\vec{x}) + b(\vec{x}) \cdot \vec{u} \tag{4.17}$$

where $f(\vec{x})$ and $b(\vec{x})$ are nonlinear functions and $\vec{u}$ is the control input.

If the control input has the expression

$$\vec{u} = \frac{1}{b(\vec{x})} \left[ \vec{a} - f(\vec{x}) \right] \tag{4.18}$$

the non linearities will be canceled, and the simple input–output relation will be obtained:

$$\vec{x}^{(n)} = \vec{a} \tag{4.19}$$

where $\vec{a}$ is a new input vector to be defined below.

In the case of robot systems, the dynamic model is expressed by Eq. 4.13. Working with this expression

$$\ddot{\vec{q}} = M^{-1}(\vec{q})(\vec{\tau}_c - \vec{C}(\vec{q}, \dot{\vec{q}})\cdot\dot{\vec{q}} + \vec{G}(\vec{q}) \tag{4.20}$$

the following terms can be defined:

$$f(\vec{x}) = M^{-1}(\vec{q})(-\vec{C}(\vec{q}, \dot{\vec{q}})\cdot\dot{\vec{q}} - \vec{G}(\vec{q})) \tag{4.21}$$

$$b(\vec{x}) \equiv M^{-1}(\vec{q}) \tag{4.22}$$

Using the general expression 4.18

$$\vec{\tau}_c = \frac{1}{M^{-1}(\vec{q})}\left[\vec{a} - M^{-1}(\vec{q})(-\vec{C}(\vec{q}, \dot{\vec{q}})\cdot\dot{\vec{q}} - \vec{G}(\vec{q}))\right] \tag{4.23}$$

the controllers based on the inverse dynamics could be viewed as particular cases of the following control law:

$$\vec{\tau}_c = M(\vec{q})\cdot\vec{a} + \vec{C}(\vec{q}, \dot{\vec{q}})\cdot\dot{\vec{q}} + \vec{G}(\vec{q}) \tag{4.24}$$

The inverse dynamics control 4.24 shows how the non-linearities, such as Coriolis vector $\vec{C}(\vec{q},\dot{\vec{q}})\cdot\dot{\vec{q}}$, as well as the gravity term $\vec{G}(\vec{q})$ can be simply compensated by adding these forces to the control input.

Depending on the expression used in $\vec{a}$, different control strategies can be obtained. In this work, a trajectory controller has been implemented. For this controller, the $\vec{a}$ expression is

$$\vec{a} = \ddot{\vec{q}} - K_d\dot{\vec{e}} - K_p\vec{e} \tag{4.25}$$

where $\vec{e} = \vec{q} - \vec{q}_d$ ,$\dot{\vec{e}} = \dot{\vec{q}} - \dot{\vec{q}}_d$ , $K_d$, and $K_p$ are diagonal and positive-definite matrices and $\vec{q}_d(t)$, and is the time-varying trajectory and its successive derivatives which describe the desired velocity and acceleration, respectively.

The control law of Eqs. 4.24 and 4.25 applied to the robot dynamics gives the close-loop equation:

$$\ddot{\vec{e}} - K_d\dot{\vec{e}} - K_p\vec{e} = 0 \tag{4.26}$$

The error equation 4.26 is exponentially stable by a suitable choice of the matrices $K_d$ and $K_p$. Figure 4.3 shows the inverse dynamics controller proposed. The figure shows how non-linearities, such as Coriolis, gravity, and inertia terms, can easily be compensated for by adding these forces to the control input. The proportional and derivative terms compose the linear auxiliary control input $\vec{a}$.

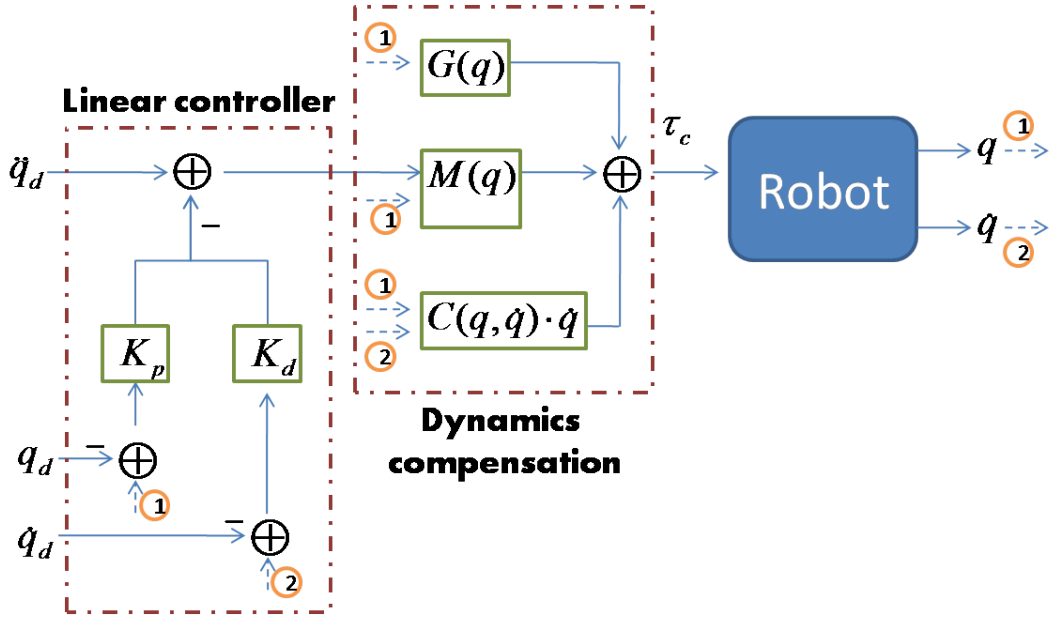



**Figure 4.3: Point-to-point inverse dynamic controller**

In order to validate the methodology proposed in this chapter, various Matlab/Simulink schemes have been developed. Figure 4.4 illustrates the reference and real position of the robot joint $q_1$. As can be seen, a good response is obtained in following the trajectory. The response is quite similar for joints $q_6$ and $q_8$.

### 4.3.2. Position task space control

Control of parallel robots is naturally achieved in the joint space, since the control inputs are the joint torques. Nevertheless, the user specifies a motion in the task space, and thus it is important to extend the control problem to the task space. This can be achieved by following the equation below [137], for example

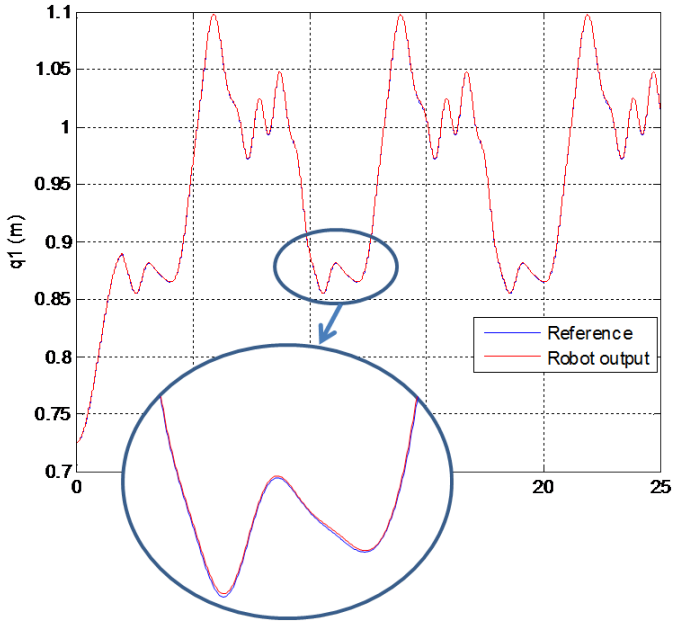



**Figure 4.4: Reference and robot position**

$$\vec{\tau}_c = J^T(\vec{q}) \cdot K_p \cdot (\vec{x}_d - \vec{x}) - J^T(\vec{q}) \cdot K_d \cdot J(\vec{q}) \cdot \dot{\vec{q}} + \vec{G}(\vec{q}) \tag{4.27}$$

where $\vec{x}_d$ is the desired robot location vector, $x$ is the robot location vector, $J$ is the Jacobian matrix,$\dot{\vec{q}}$ and $\vec{q}$ are the robot position and velocity, and $\vec{G}(\vec{q})$ is the robot gravity term. Figure 4.5 shows the task space controller that has been developed.

As in the previous cases, simulations have also been implemented with this controller. Figure 4.6 shows the height ($z$) reference for the platform and the robot height, while Fig. 4.7 illustrates the parallel robot platform orientation: the *gamma* and *beta* angles.

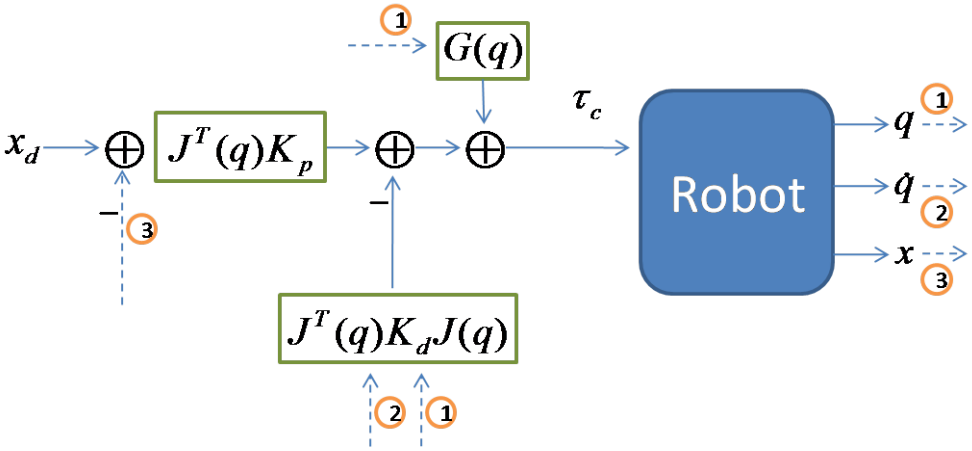



**Figure 4.5: Task space controller**

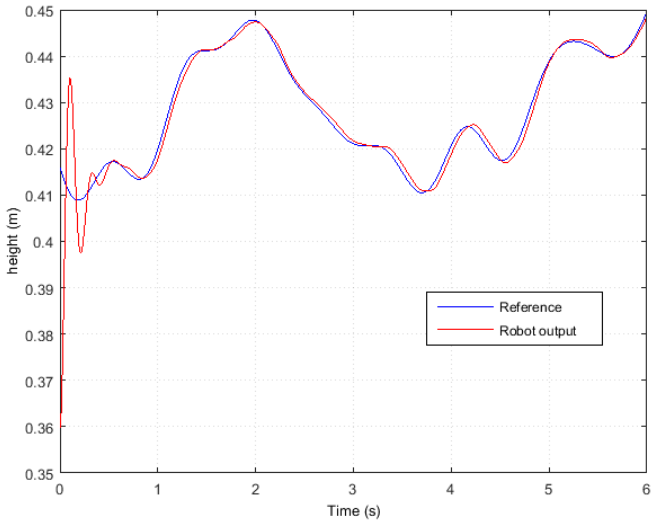



**Figure 4.6: Response task space controller: robot height**

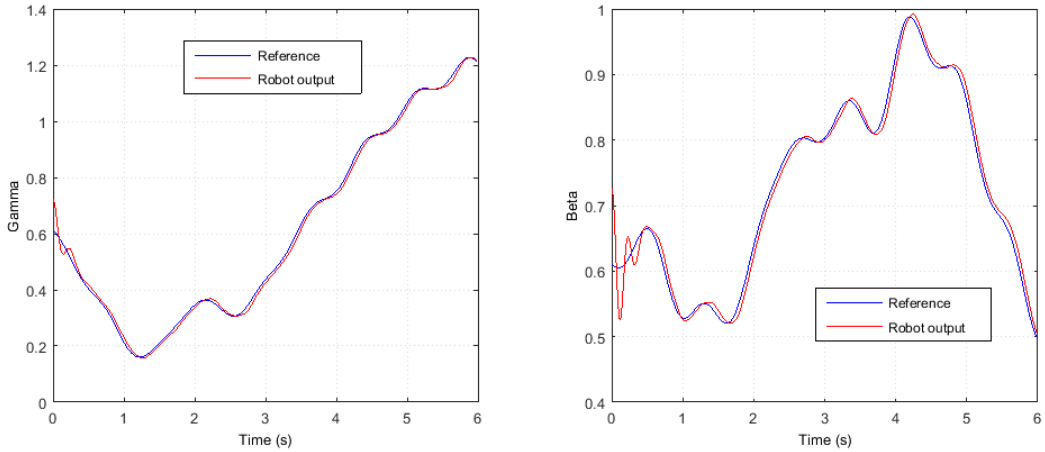



**Figure 4.7: Response task space controller: robot gamma and beta angles**

### 4.3.3. Force control

In addition to controlling position, force control is increasingly being used in industrial robots. Various types of control strategies can be chosen to establish force control, the first of which is based on typical force control and consists of following a force reference. This control is restricted to a linear control [138], such as PID force control. The effect of the three parameters of a PID controller is well known and has been extensively described in the literature [139]. The integral term ensures zero tracking error while the function of the derivative term is to dampen the system. The resulting force control law is given in Eq. 4.28:

$$\tau_f = K_p(F_{ref} - f) + K_i \int (F_{ref} - f)dt + K_d \frac{d}{dt}(F_{ref} - f) \tag{4.28}$$

where $\tau_f$ is the force control action, $F_{ref}$ is the force reference, $f$ is the measured force, and $K_p$, $K_i$, and $K_d$ are proportional, integral, and deri-

vative constants.

A Matlab/Simulink scheme has been developed in order to simulate and analyze force control. This scheme uses the force control of Eq. 4.28 and follows the control architecture presented in [140]. As mentioned above, due to the configuration of the parallel robot, it has three degrees of freedom: the height ($z$) of the platform and the orientation ($\gamma$ and $\beta$). Thus, the robot has an established control force/position: force control in $z$ axis and orientation control of the robot's platform. The figure below illustrates a control scheme developed for this work.

Figure 4.8 shows the controller developed in this work. The position reference for the parallel robot is $x_d$, while $f_d$ is the force reference that the robot shall apply to the environment. The specific force control algorithm is programmed in the *Force Controller* block using the error signal between the force reference and the force measured on the robot's platform. Finally, new position references $q_d$ are computed and sent to the joint space robot controller. These references are obtained through the outputs of the parallel robot's inverse kinematics ($\hat{q}_d$) and the force controller ($\Delta q$).

In Fig. 4.9 the force applied by the robot (in red) follows the reference with a very small degree of error.

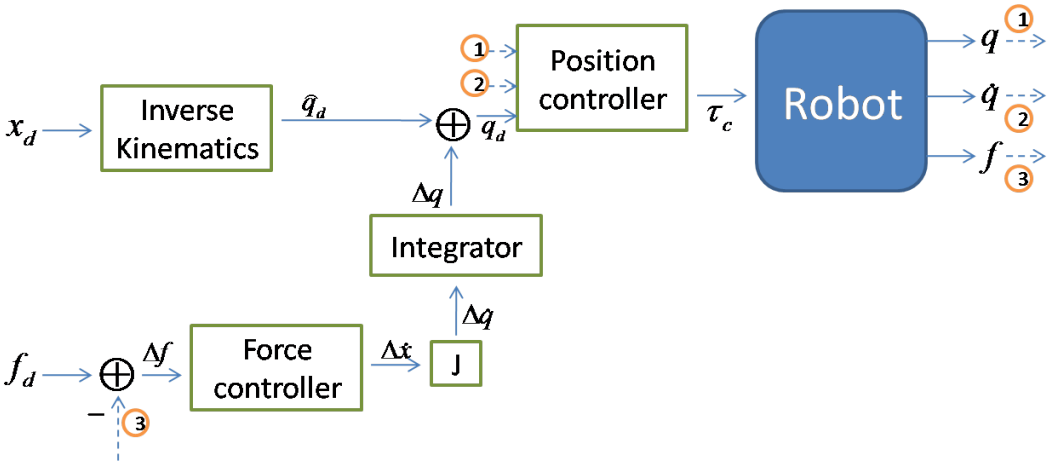



**Figure 4.8: Force control scheme developed**

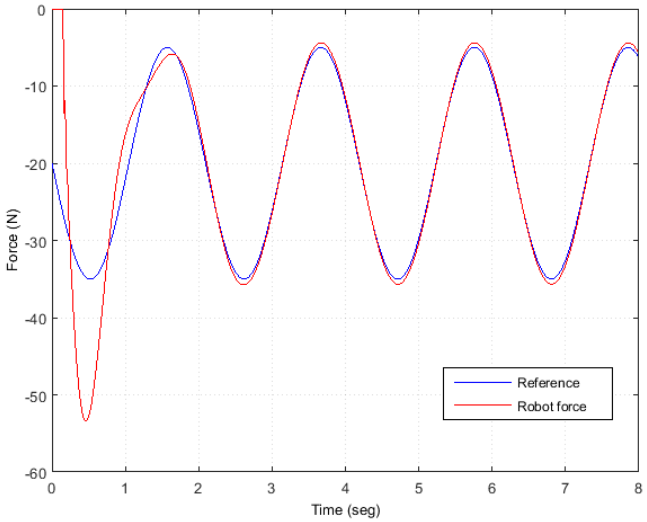



**Figure 4.9: Parallel robot force controller response**

## 4.4. Development of force control over the actual prototype

An industrial PC has been used to implement the control architecture for the parallel robot. It is based on a high performance 4U Rackmount industrial system with 7 PCI slots and 7 ISA slots. It has a 2.5 GHz Intel Pentium Core 2 Quad/Duo processor and 4 GB SDRAM.

The industrial PC is equipped with two AdvantechTM data acquisition cards: a PCI-1720 and a PCL-833. The first is used to supply the control actions for each parallel robot actuator and the second is used to read the encoder measurements.

The force control is established with the ATI sensor Delta SI-330-30. This is a sensor made of silicon strain gauges with six DOF which can measure forces and moments on the XYZ axes, providing a measurement error of nearly zero. The control unit measures these signals using the network force/torque sensor system. It provides an Ethernet/IP communication interface. Figure 4.10 illustrates the control architecture developed for this robot.

The parallel robot is controlled using the C++ programming language (with Orocos middleware). The PC is equipped with the Linux Ubuntu system, patched with Xenomai (a real-time kernel). Thus, real-time features are available. Using this environment, joint space control strategies based on inverse dynamics have been implemented. The sample time used for the control algorithm is 10 mS.

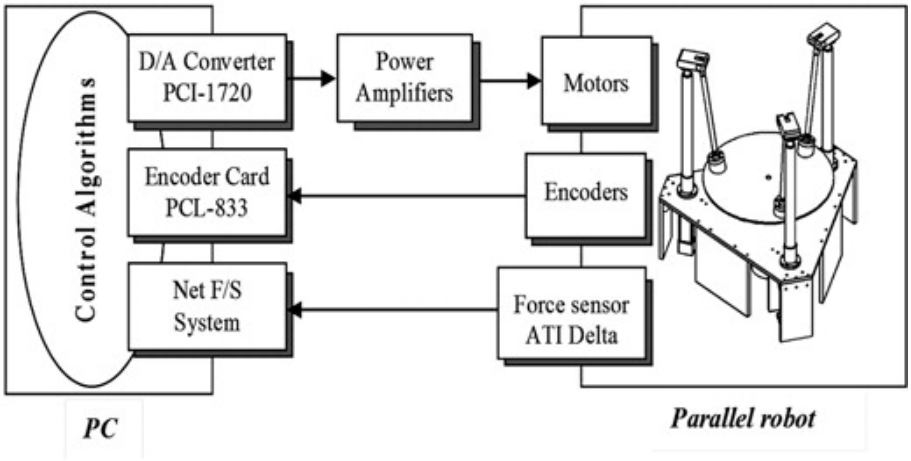



**Figure 4.10: Control architecture developed**

Figure 4.11 shows the reference and the real position (for the joint $q_1$) of the parallel robot. As can be seen in Fig. 4.12, a high accuracy is obtained.

Based on the calculated Jacobian matrix, a task space position controller has also been implemented using the actual parallel robot and Orocos middleware. The sample time is the same as for the joint space controller: 10 mS. Figure 4.13 shows the robot height while Fig. 4.14 and Fig. 4.15 show the platform orientation (roll and pitch). As occurs with joint space control, errors in the executions are very small.

Finally, the force robot control has been implemented with the parallel robot. The sample time of this controller is 10 mS. Figure 4.16 illustrates the reference and the force applied by the robot on the $z$ axis. For the first 7s (approximately), the force applied by the robot is zero as during that time the platform is moving down and has no contact with the environment. From that point on, the terminal element collides with the environment and the control force is established, using a sinusoidal force

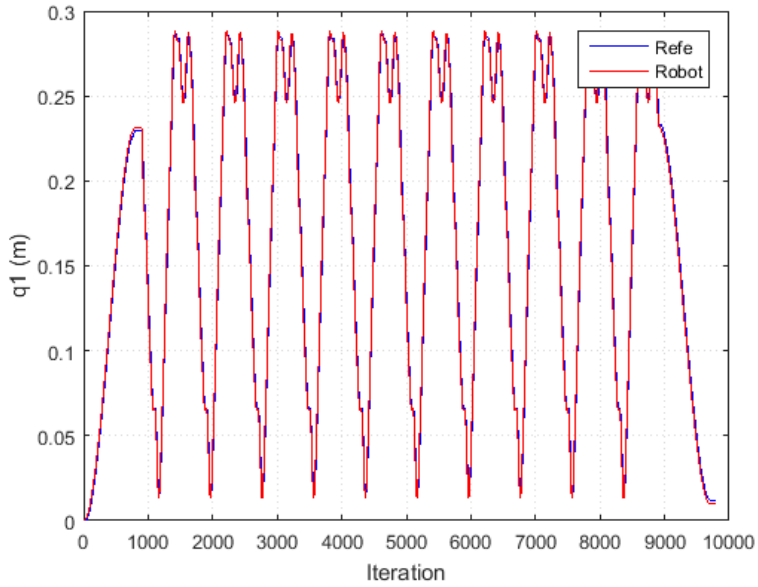



**Figure 4.11: $q_1$ reference and position**

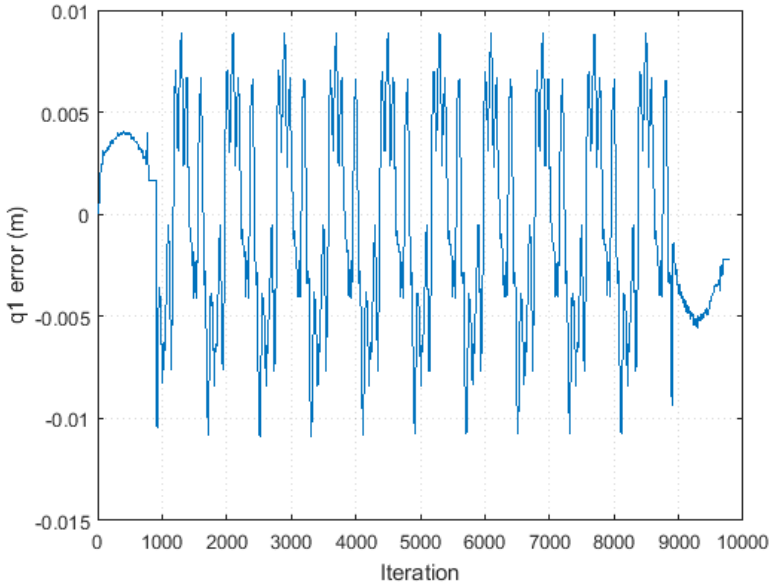



**Figure 4.12: Actual robot position and error**

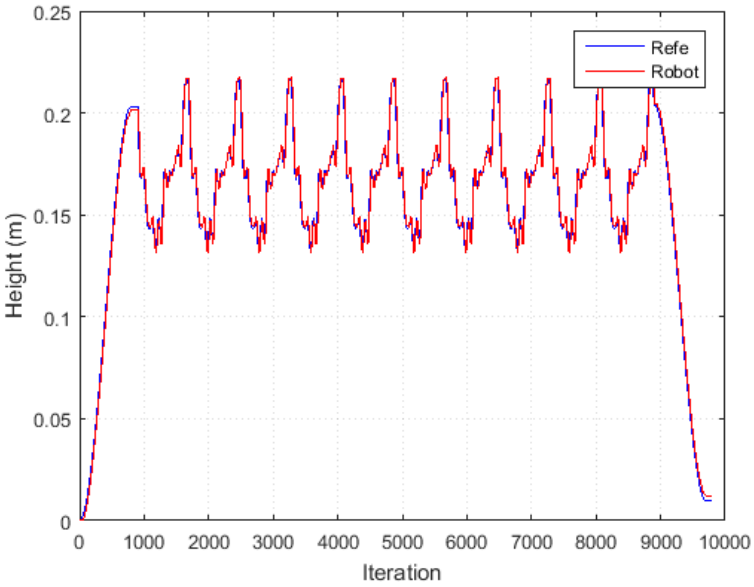



**Figure 4.13: Height of the robot platform**

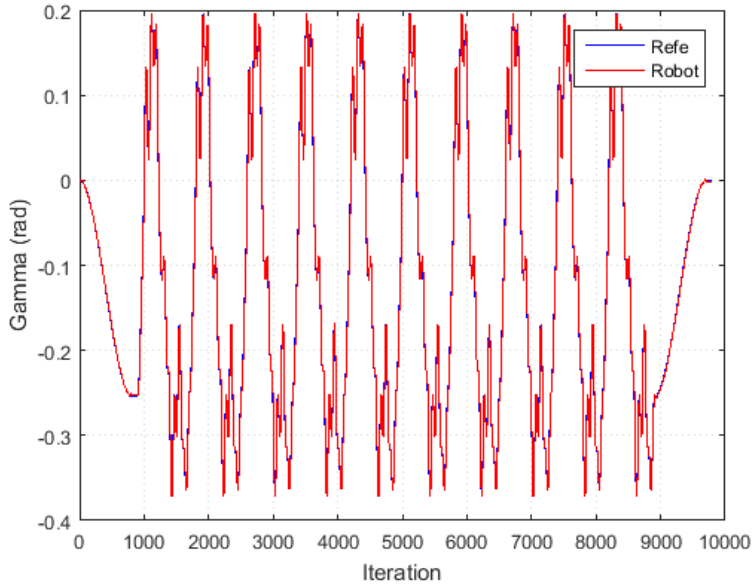



**Figure 4.14: Robot platform orientation (roll)**

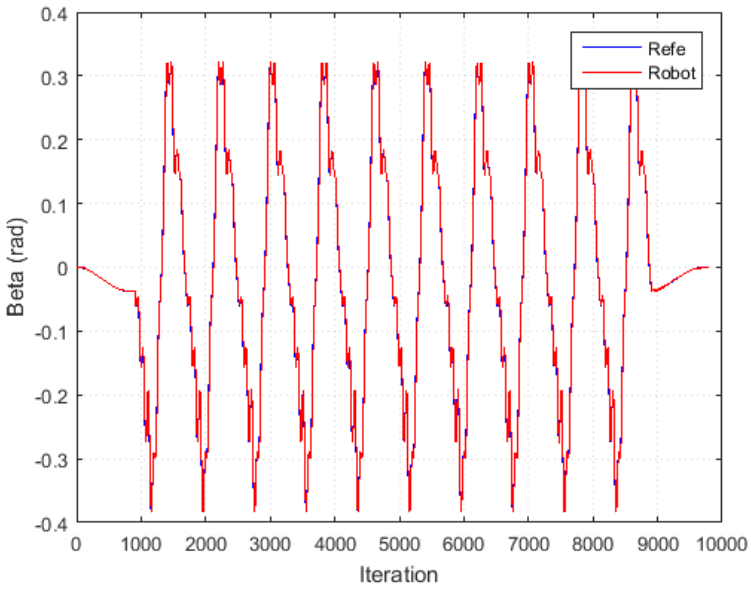



**Figure 4.15: Robot platform orientation (pitch)**

reference. As can be seen in the figure, the force applied by the robot follows the reference very accurately. Figure 4.17 shows the control action for the actuator of coordinate $q_1$.

## 4.5. Conclusions

This chapter studied the position and force control of a parallel robot with three DOF. In addition to designing the robot, a completely open and flexible control system (based on an industrial PC) was developed. The PC was equipped with card D/A converters, encoder reading, and a force sensor. Programs and control applications were developed with the programming language C++.

In order to establish control of the robot, direct and inverse kinematic

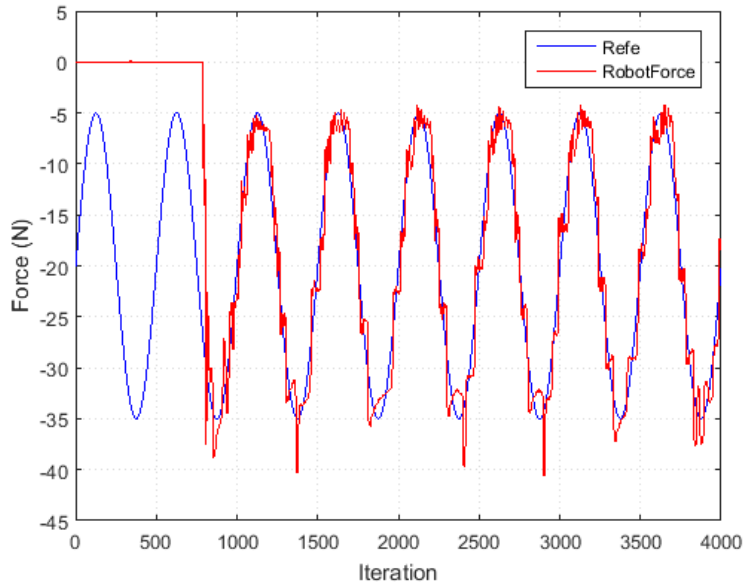



**Figure 4.16: Reference and actual robot force**

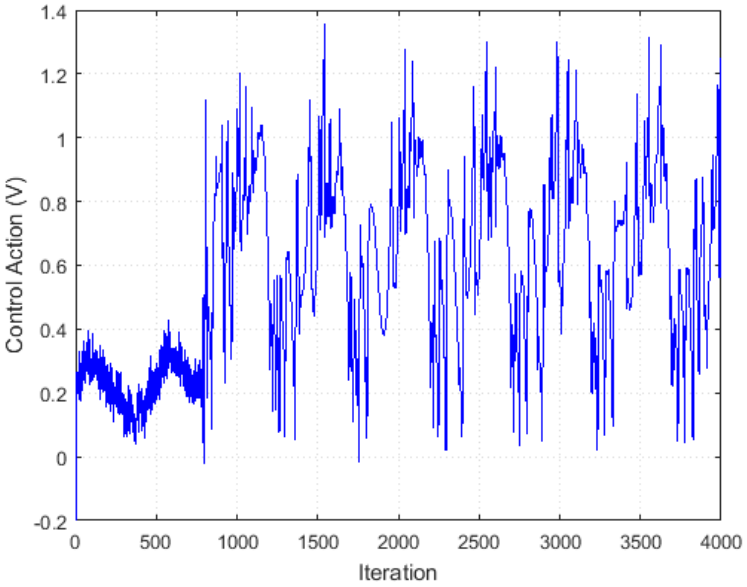



**Figure 4.17: $q_1$control action**

models of the robot were solved, as well as the dynamic model and the Jacobian needed to implement force control.

A model-based controller was chosen to develop the position control. The response obtained with this position control, both in simulations and with the real robot, showed accurate response in terms of position error.

A hybrid force/position controller was also implemented. A force sensor was installed on the platform in order to measure the forces and torques applied by the robot. The performance obtained from simulations and with the actual prototype demonstrated that the robot can accomplish the required task.

# Parte III

## Desarrollo de un robot de rehabilitación de miembros inferiores basado en componentes

# Capítulo 5

## A 3-PRS parallel manipulator for ankle rehabilitation: towards a low-cost robotic rehabilitation

En el siguiente capítulo, se presenta una aplicación capaz de realizar diversos ejercicios de rehabilitación en miembros inferiores, usando todos los controladores previamente desarrollados. El capítulo está basado en la publicación científica [141], en la cual, una bota ortopédica está equipada con un sensor de fuerza permitiendo la realización de ejercicios en el tobillo para entrenar movimientos referentes a la dorsi/plantar flexión, inversión y eversión. Además de ello, usando la integración ROS-Orocos, se ha implementado una aplicación que permite la tele-operación del robot mediante un dispositivo Bluetooth.

**Abstract**

This chapter presents the design, kinematics, dynamics and control of a low-cost parallel rehabilitation robot developed at the Universitat Politècnica de Valencia. Several position and force controllers have been tested to ensure accurate tracking performances. An orthopedic boot, equipped with a force sensor, has been placed over the platform of the parallel robot to perform exercises for injured ankles. Passive, active-assistive and active-resistive exercises have been implemented to train dorsi/plantar flexion, inversion and eversion ankle movements. In order to implement the controllers, the component-based middleware Orocos has been used with the advantage over other solutions that the whole scheme control can be implemented modularity. These modules are independent and can be configured and reconfigured in both configuration and run-time. This means that no specific knowledge is needed by medical staff, for example, to carry out rehabilitation exercises using this low-cost parallel robot. The integration between Orocos and ROS, with a CAD model displaying the actual position of the rehabilitation robot in real time, makes it possible to develop a teleoperation application. In addition, a teleoperated rehabilitation exercise can be performed by a specialist using a Wiimote (or any other Bluetooth device).

## 5.1. Introduction

The development of mechatronic devices for applying forces or controlling human motions is not new in biomechanics. Indeed, there are several precedents such as in surgery, rehabilitation and, to a lesser extent, in functional assessment and trials for diagnostic support areas [105, 142]. In the field of rehabilitation, the idea is to develop a device that mimics the work done by the patient along with the physiotherapist during the rehabilitation session.

The quality of healthcare and the clinical rehabilitation process improve when using robotic devices to assist physiotherapists, because it can aid by developing evidence-based therapy (deliver the optimal therapy to a particular patient by monitoring his or her biomedical variables). In addition, a robotic device can increase the productivity of therapists. In this sense, robotic rehabilitation systems allow patients to perform a wide range of self-administered tasks from passive repetitive actions to functional activities and from assistive tasks to those providing opposition. These systems allow patients to train repetitively and intensively and provide physiotherapists with tools that allow them to treat patients with minimal supervision.

Recent rehabilitation devices have been reviewed in [143, 144]. They include a broad range of application such as systems for gait training [145–147], modified isokinetic tables [148], systems for rehabilitation of the upper limb [149] or active orthoses [150–152].

In the particular case of the rehabilitation of the lower limb, devices have

been developed to generate ankle motions, usually intended for neurological rehabilitation [153–155], although other applications are intended for ankle sprains [156]. The characteristics of the exercises to be performed in each case are very different, which means that the robot can be configured to suit different needs [157].

On the other hand, in order to reproduce these movements, a great effort and a very precise control of movements are needed. This control should cover both position and effort aspects when performing each type of exercise. These features make parallel robots more advantageous than other solutions [142].

The first device proposed for ankle rehabilitation is the Rutgers Ankle [73]. The device is a six Degree-of-Freedom (6-DOF) Parallel Robot consisting of a mobile platform and a fixed based connected by several open kinematic chains. The 6-DOF allows the ankle joint, which is called the ankle joint complex, to move within the range of the ankle motion. However, ankle rehabilitation does not necessarily require 6-DOF. For this reason, parallel robots with 3-DOF and 4-DOF are proposed in [156]. In addition, the range of motion may vary depending on the patient's ankle, which is why a reconfigurable device with respect to the range of motion of each patient's ankle is proposed in [157]. In the references presented so far, the ankle joint complex is modeled as a spherical joint. Recently, the 2-DOF ankle model was proposed based on the functional aspects of the ankle kinematic model. This ankle model was then used to develop a tripod parallel robot [158].

In addition to parallel robot-based devices, wearable 3-DOF and 4-DOF

parallel robots are proposed in [154] and [159]. In these two papers, the mobile platform is linked to the foot while the fixed platform is attached to the lower extremity. Due to the fact that parallel robots have a small workspace and suffer from singular configurations, robot configurations with actuation redundancy have been proposed [160]. However, actuation redundancy has the disadvantage of making them more expensive.

More recent examples of parallel robots used in ankle rehabilitation are the one proposed in [153], where selection and design of the control algorithms are based on the analysis of the rehabilitation protocol taking into account the dynamics of the system and the dynamics of the interaction between the human and the robot. In [161] a robot is proposed for dynamic posturography studies, where multi-axial perturbations are required. An analysis of accuracy, workspace range and dynamic performance on the control of roll, pitch and yaw angles is achieved. In the above case, no force sensor measurement was taken into account.

In addition to finding a suitable mechanical solution, it is very important to provide adaptability in terms of the type of exercises that can be performed. This involves not only an appropriate kinematic and dynamic design but also developing control systems that are capable of monitoring both movements and forces and integrating the whole system into an easily configurable interface for medical staff.

In this project, we propose a new prototype of parallel robot for ankle rehabilitation with a high level of versatility: patient adaptation, different types of exercises and ease of use for medical personnel.

Based on the design specifications, a 3-DOF PRS parallel robot was conceptualized (see Fig. 5.1). A platform type was developed instead of a wearable robot. The platform type allows to account for the weight of the patient, so proprioceptive exercise can be performed. The prototype has two rotational DOF; thus, the main rehabilitation exercise: Dorsi/Plantar flexion exercise and Eversion/Inversion can be performed. In addition, the robot has one translational DOF that accounts for the height of the patient while sitting on a chair, making it adaptable in a vertical direction. A force sensor has been added to allow a feed forward signal to determine the strength achieved by the patient. This makes it possible to implement different controllers that will be used depending on the patient activity required (active or passive motions). Another major advantage of this project is how the controllers are implemented. The software design technique is based on modular programming. The functionalities of the program are separated into independent and interchangeable modules. The controller's modules can be loaded and executed depending on the rehabilitation exercises required.

This allows the functionality of the robot to be changed dynamically. In this way, the physiotherapist may adapt the therapy based on how the patient is recovering. Furthermore, a remote control module has been put forward, meaning that a specialist can use a WiiMote (or any other Bluetooth device with accelerometers) to perform an "e-rehabilitation".

The chapter is organized as follows: the parallel robot design is shown in Section 5.2. Section 5.3 deals with the model-based position and force control schemes. Section 5.4 presents the robot hardware and software

architecture. Experiments and results with the ankle rehabilitation robot are shown in Section 5.5. Finally, conclusions are presented in Section 5.6.

## 5.2. The 3-DOF Parallel Manipulator

### 5.2.1. Physical description of the low-cost PM

As mentioned before, a 3-DOF spatial PM was used to address the design problem. The robot consists of three kinematic chains; each chain has a PRS configuration (P, R, and S stand for prismatic, revolute, and spherical joint, respectively). The underlined format (P) indicates the actuated joint. The choice of the PRS configuration was guided by the need to develop a low-cost robot with 2-DOF of angular rotation in two axes (Dorsi/Plantar Flexion and Eversion/Inversion) and 1-DOF of translation motion (height). In [121], a complete description of the mechatronic development process of the PM is presented.

The physical system consists of three legs connecting the moving platform to the base. Each leg is driven by a direct drive ball screw (prismatic joints). A coupler bar is connected to the ball screw with a revolute joint. A spherical joint connects the upper part of the coupler to the moving platform. The lower part of the ball screws are perpendicularly attached to the platform's base. The ball screws are distributed on the base in an equilateral triangular configuration. The ball screw transforms the rotational movement of the motor into linear motion.

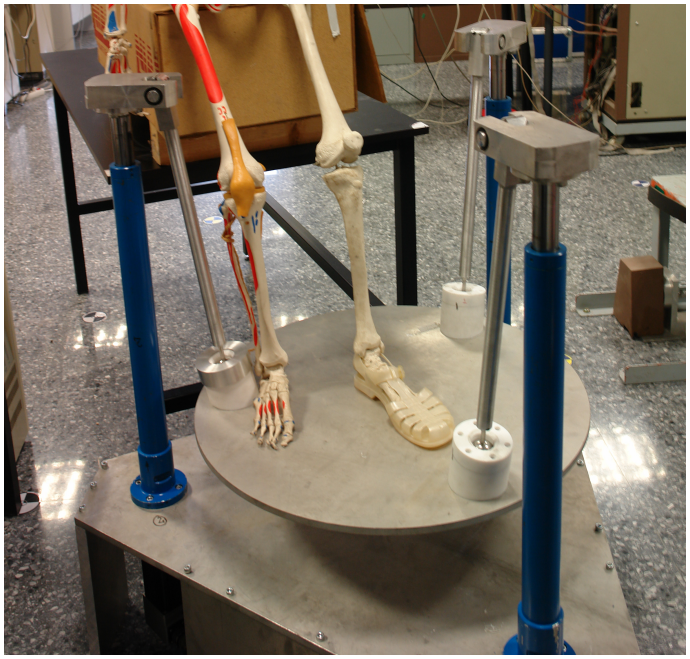


**Figure 5.1: The 3-PRS low-cost parallel manipulator**

The motors in each leg are brushless DC servomotors equipped with power amplifiers. The actuators are Aerotech BMS465 AH brushless servomotors. The specifications of these motors are a stall torque of 2.86 N.m and a peak torque of 11.43 N.m, both continuous. The lead of the ball screw is 20 mm, Aerotech BA10 power amplifiers operate the motors and the control system was developed on an industrial PC.

### 5.2.2. Kinematic and dynamic model

Not only the kinematics of the robot (see Subsec 4.2.1) are needed for the development of a rehabilitation robot, but also the implementation and testing of the dynamic robot control schemes. The motion dynamic controller requires the general equation of motion to be described as follows

$$M(\vec{q},\vec{\Phi})\cdot\ddot{\vec{q}} + \vec{C}(\vec{q},\dot{\vec{q}},\vec{\Phi})\cdot\dot{\vec{q}} + \vec{G}(\vec{q},\vec{\Phi}) = \vec{\tau} \tag{5.1}$$

In Eq. 5.1, $M$ stands for the system mass matrix, $\vec{C}$ is the matrix grouping the centrifugal and Coriolis terms, $\vec{G}$ is the vector corresponding to gravitational terms and $\vec{\tau}$ is the vector of generalized forces. In this work, a coordinate partitioning procedure has been considered in order to allow the system to be modeled by a set of independent generalized coordinates grouped together in the vector $\vec{q} = \begin{bmatrix} q_1 & q_6 & q_8 \end{bmatrix}^T$

Equation 5.1 can be rewritten (see [65, 108]) as follows

$$\begin{bmatrix} K_{rb} & K_r & K_f \end{bmatrix} \cdot \begin{bmatrix} \vec{\Phi}_{rb} \\ \vec{\Phi}_r \\ \vec{\Phi}_f \end{bmatrix} = \vec{\tau} \tag{5.2}$$

In Eq. 5.2, $\vec{\Phi}_{rb}$, $\vec{\Phi}_r$, $\vec{\Phi}_f$ are the vectors grouping the rigid body, rotor and friction parameters. $K_i$ is the part of the regressor matrix determining the linear relationship between the corresponding parameters (rigid body, rotor, and friction) and the generalized forces.

From Eq. 5.2, a set of base parameters corresponding to the complete base parameter model can be obtained. However, not even those parameters can always be identified properly. Thus, the reduced model containing only the relevant parameters will be obtained through a process that considers the robot's leg symmetries, the influence on the dynamic behavior of the robot, the statistical significance of the parameters identified

and the physical feasibility of the parameters [65].

The dynamic terms of the equation of motion for the actual parallel robot based on relevant parameters can be written as follows

$$M(\vec{q})\cdot\ddot{\vec{q}} = \begin{bmatrix} J_1 & 0 & 0 \\ 0 & J_2 & 0 \\ 0 & 0 & J_3 \end{bmatrix} \cdot \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_6 \\ \ddot{q}_8 \end{bmatrix} + \begin{bmatrix} M_{11}(\vec{q}) & M_{12}(\vec{q}) & M_{13}(\vec{q}) \\ M_{21}(\vec{q}) & M_{22}(\vec{q}) & M_{23}(\vec{q}) \\ M_{31}(\vec{q}) & M_{32}(\vec{q}) & M_{33}(\vec{q}) \end{bmatrix} \cdot \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{bmatrix} \tag{5.3}$$

$$C(\vec{q},\dot{\vec{q}})\cdot\dot{\vec{q}} = \begin{bmatrix} F_{v_1}\dot{q}_1 + F_{c_1}sign(\dot{q}_1) \\ F_{v_2}\dot{q}_6 + F_{c_2}sign(\dot{q}_6) \\ F_{v_3}\dot{q}_8 + F_{c_3}sign(\dot{q}_8) \end{bmatrix} + \begin{bmatrix} C_{11}(\vec{q},\dot{\vec{q}}) & C_{12}(\vec{q},\dot{\vec{q}}) & C_{13}(\vec{q},\dot{\vec{q}}) \\ C_{21}(\vec{q},\dot{\vec{q}}) & C_{22}(\vec{q},\dot{\vec{q}}) & C_{23}(\vec{q},\dot{\vec{q}}) \\ C_{31}(\vec{q},\dot{\vec{q}}) & C_{32}(\vec{q},\dot{\vec{q}}) & C_{33}(\vec{q},\dot{\vec{q}}) \end{bmatrix} \cdot \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{bmatrix} \tag{5.4}$$

$$G(\vec{q}) = g\cdot \begin{bmatrix} G_{11}(\vec{q}) & G_{12}(\vec{q}) & G_{13}(\vec{q}) \\ G_{21}(\vec{q}) & G_{22}(\vec{q}) & G_{23}(\vec{q}) \\ G_{31}(\vec{q}) & G_{32}(\vec{q}) & G_{33}(\vec{q}) \end{bmatrix} \cdot \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{bmatrix} \tag{5.5}$$

Where $J_i$ are the actuator relevant parameters, $F_{Ci}$ and $F_{Vi}$ are the friction base parameters, and $\Omega_1$ are the rigid body base parameters considered as relevant ones. $M_{ij}$, $C_{ij}$ , and $G_{ij}$ are the elements of the inertial, Coriolis and gravity matrices that depend on the $\Omega_1$ parameters.

## 5.3. Robot Position and Force Control Schemes

### 5.3.1. Model-based position control schemes

In recent years, the passivity-based approach to robot control has gained a lot of attention. This approach solves the robot control problem by exploiting the robot system's physical structure, and specifically its passivity property. The design philosophy of these controllers is to reshape the system's natural energy in such a way that the tracking control objective is achieved [101].

For the tracking problem, the kinetic and potential energy must be modified as required in passivity based controllers. The general expression of the controllers is [121]

$$\tau_c = M(q)\cdot a + C(q,\vec{\dot{q}})\cdot\vec{v}_1 + G(q) - K_p\cdot e - K_d\cdot v_2 \qquad (5.6)$$

Where $a$, $v_1$, $v_2$, and $e$ varies according to the type of controller (see Table 5.1). In all these controllers, the control law has two parts: compensation of robot dynamics and a proportional and derivative controller.

where $\vec{\dot{q}}_r = \vec{\dot{q}} - \Lambda_1\vec{e}$, $\Lambda_1$ is a symmetric positive definite matrix and $\vec{q}_d$ is the desired position.

| Controller | $a$ | $v_1$ | $e$ | $v_2$ |
|---|---|---|---|---|
| Paden, Panja [102] | $\ddot{\vec{q}}_d$ | $\dot{\vec{q}}_d$ | $\vec{q} - \vec{q}_d$ | $\dot{\vec{e}}$ |
| Slotine, Li [162] | $\ddot{\vec{q}}_r$ | $\dot{\vec{q}}_r$ | $0$ | $\dot{\vec{e}} + \Lambda_1 \vec{e}$ |
| Sadegh, Horowitz [163] | $\ddot{\vec{q}}_r$ | $\dot{\vec{q}}_r$ | $\vec{q} - \vec{q}_d$ | $\dot{\vec{e}} + \Lambda_1 \vec{e}$ |

**Table 5.1: Passivity-based tracking controllers**

The first controller is a variation of the classic PD with a gravity compensation regulator. The second is a tracking controller based on the sliding mode theory. In the last case, some modifications have been made to the control law and the energy function. It allows the system's asymptotic stability to be shown using the Lyapunov theory.

### 5.3.2. Robot force control

In this work, different types of force control strategies have been developed and used. The first type is based on explicit force control, which consists of following the force reference value using feedback (and feedforward, if necessary). Usually, the explicit control is restricted to a linear control [138], as is the case of the classic PID force controller in Eq. 5.7.

$$F = K_p \cdot (F_{ref} - f) + K_I \cdot \int (F_{ref} - f) dt + K_d \cdot \frac{d}{dt}(F_{ref} - f) \qquad (5.7)$$

where $F_{ref}$ and $f$ are the force reference and the force measured, respectively. $F$ is the force control action.

Even a PID controller performs in a decent way, the force control application has some specific problems that may require modifications of this

classic controller. Firstly, in constrained motion, the dynamics of the system depend on the characteristics of the environment. Basically, this means that the parameters of the controller must be re-tuned for every application. Sometimes the characteristics of the environment are not known in advance. On the other hand, it may be demonstrated that the integral term can make the system unstable. One solution may be to use a force feed forward term instead of the integrator [164]

$$F = F_{ref} + K_p \cdot (F_{ref} - f) + K_d \cdot \frac{d}{dt}(F_{ref} - f) \tag{5.8}$$

The second type of controller implemented in this work is based on the impedance control proposed in [165]. In this case, the purpose is not to follow a reference value, but to ensure the desired dynamic behavior of the system (which is known as mechanical impedance). Typical specified dynamics correspond to a second order spring-damper system determined by the following expression

$$Z(s) = \frac{F(s)}{v(s)} = \frac{M \cdot s^2 \cdot X + B \cdot s \cdot X + K \cdot X}{s \cdot X} = M \cdot s + B + \frac{K}{s} \tag{5.9}$$

where $M$, $B$, and $K$ are the mass, damping and stiffness parameters and $s$ the Laplace transform operator. These parameters determine the closed-loop system behavior with respect to contact force $F$.

The description of the different methods for implementing the impedance control lies outside the scope of this work, but typically they are a combination of linear feedback and inverse dynamics, both described in [27].

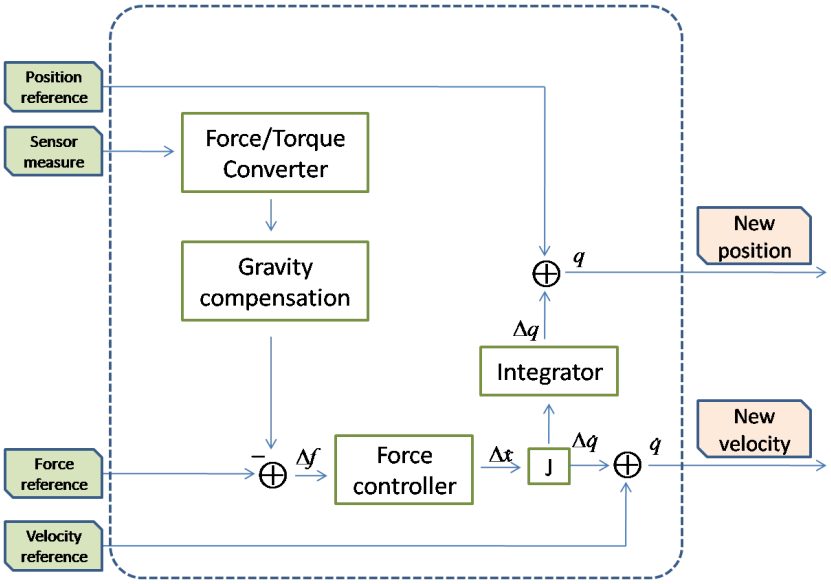



**Figure 5.2: Structure of the force control**

An interesting alternative is proposed in [166], where sliding control is established.

Figure 5.2 shows the implementation of the force control architecture developed in this work in more detail. The controller receives the position and velocity references for the robot axes, as well as the force reference that the robot shall apply to the human/environment. The actual force in the robot end effector is measured by the platform-mounted force sensor. The specific force control algorithm is programmed into the *"Force Controller"* block. Finally, the new position and velocity references are computed by means of the Jacobian matrix.

The proposed architecture provides several advantages. On the one hand, a very accurate force control can be established. On the other hand, as it is an open, flexible architecture, any force control algorithm can be implemented.

## 5.4. Robot Control Architecture

### 5.4.1. Robot hardware architecture

In order to implement the control architecture for the parallel robot, an industrial PC has been used. It is based on a high-performance 4U Rack-mount industrial system with 7 PCI slots and 7 ISA slots. It has a 2.5 GHz Intel R Pentium R Core 2 Quad/Duo processor and 4 GB SDRAM.

The industrial PC is equipped with 2 AdvantechTM data acquisition cards: a PCI-1720 and a PCL-833. The PCI-1720 card has been used for supplying the control actions for each parallel robot actuator. It provides four 12-bit isolated digital-to-analog outputs for the Universal PCI 2.2 bus. It has multiple output ranges (0$\sim$5 V, 0$\sim$10 V,±5 V,±10 V), programmable software and an isolation protection of 2500 VDC between the outputs and the PCI bus.

The PCL-833 card is a 4-axis quadrature encoder and counter add-on card for an ISA bus. The card includes four 32-bit quadruple AB phase encoder counters, an on board 8-bit timer with a wide range time-based selector and it is optically isolated up to 2500 V.

In order to establish the force control, the robot has been equipped with the Delta SI-330-30 ATI sensor. This is a sensor with 6-DOF capable of measuring forces and torques in the XYZ axes using a monolithic instrumented transducer. The maximum range of forces is ± 3700 N for X and Y, and ±10000 N for the Z axis. The maximum range of torque is ±270

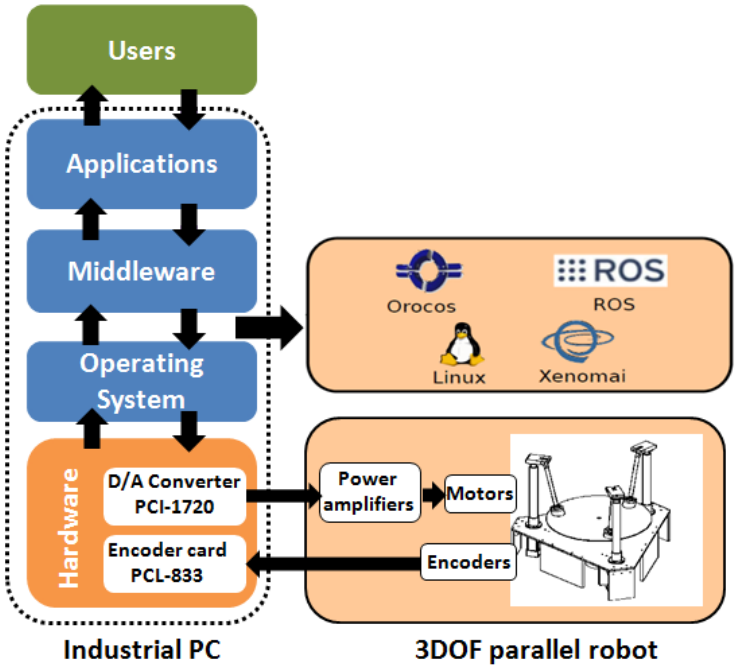



**Figure 5.3: Robot control architecture**

N.m for X and Y, and ±400 N.m for the Z axis. In order to transmit the signals from the sensor to the control unit (industrial PC in this case) there are three options: data acquisition card, F/T controller or Ethernet communication system. In this work, the last option has been implemented. The NET F/T system provides Ethernet/IP and CAN bus communication interfaces and is compatible with standard Ethernet. This device can be easily connected to any local area network, allowing more than 7000 Hz frequency for measurement of the six components, ensuring real-time communication.

Figure 5.3 shows the control architecture based on an industrial PC developed for this study.

The programming language used to control the parallel robot is C++. The

PC is equipped with the Linux Ubuntu operating system, patched with Xenomai (a real-time kernel). Therefore, real-time characteristics are available.

Because the control architecture is based on an industrial PC, it has two main advantages: first, it is totally open and gives a powerful platform for programming high-level tasks based on the Ubuntu 12.04 operating system. Thus, any controller and/or control technique can be programmed and implemented, such as automatic trajectory generation, control based on external sensing using a force sensor or artificial vision, etc. The second advantage is the low cost: the total cost of the hardware (the computer and the industrial data acquisition cards) does not exceed $ 2000. In addition, the operating system and all the development and programming tools are free software, so this does increase the cost of the control system.

### 5.4.2. Robot software architecture

It is well-known that a very difficult aspect of creating a robotic prototype is the software architecture. In recent years, component-based software development has been increasing, with the aim of achieving an execution in a distributed way, as well as the reusability of the code developed. In order to implement different robot controllers, the real-time middleware Orocos [51] (Open Robot Control Software) has been used. Because the Orocos environment provides component-based software development, it involves the following advantages that have been used in this project:

- Modular design and structure.
- Fully reusable code and modules.
- Modules configurable and reconfigurable during both setup and running time.
- Distributed execution of the modules, improving total execution time.

Due to the modular design and structure, when a number of modules are implemented and a control scheme is required, it is as simple as inserting the necessary modules to configure them, making connections with each other and making them run. Therefore, because the different control schemes have common parts, as several modules have been developed, these modules are reused to implement different controllers.

Note that, although it can be a complicated task at first, the development of component-based software makes the programmer's job easier in the end because if a module works correctly in one particular scheme, it will certainly work correctly in another control scheme (thus, a lot of programming errors are avoided since each module is only programmed once). Moreover, depending on how the modules are interconnected and how the variables are set, a different control will be realized. as Orocos allows ports connection and disconnection in run-time, as well as parameters configuration and reconfiguration in run-time too. Therefore, besides the advantages discussed above, this approach minimizes the chance of programming errors in the implementation of any of the modules.

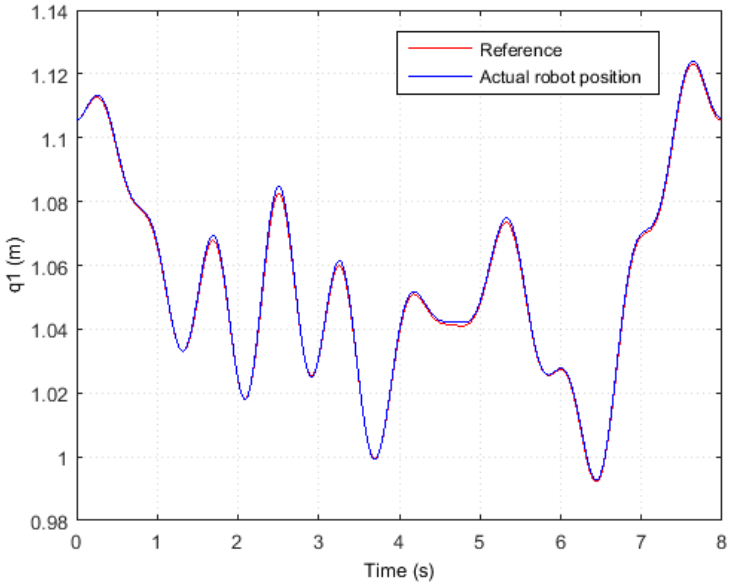



**Figure 5.4: $q_1$ robot position**

Different position and force controllers for ankle rehabilitation have been implemented and tested in the parallel robot using Orocos. Figure 5.4 shows the reference and the robot $q_1$ position. The position controller was able to generate very good tracking performance, providing a mean error of 7.61e-4 m. Figure 5.5 shows the reference and the force applied by the robot for the Z-axis.

Orocos is one of the best motion control frameworks available at the moment but it represents some constraints when trying to achieve something else than the control itself. One of the solutions is ROS (Robot Operating System) [38], that was designed from the beginning with the idea that in the robotics sector no solution will ever be infallible and therefore different approaches must coexist in order to provide the application builder with a complete toolbox.

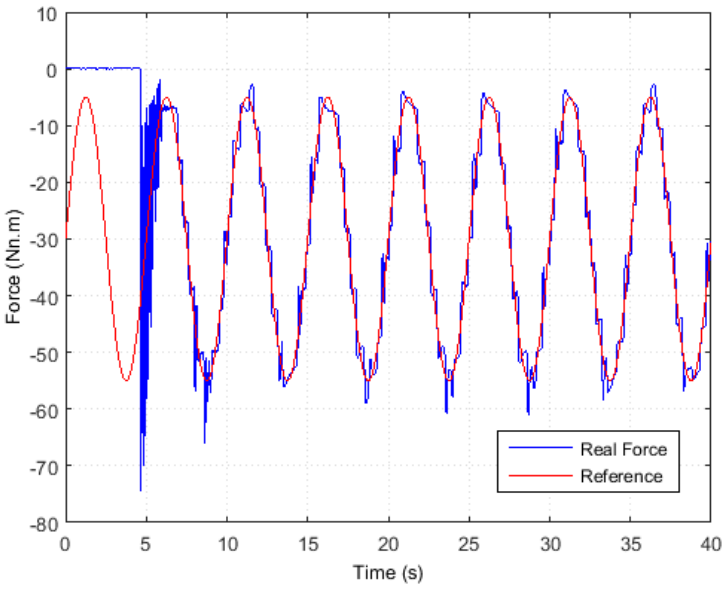



**Figure 5.5: Robot force**

ROS has been shaped to be a conglomeration of various tools which are organized in packages. Each package or "stack" may contain libraries, executables or scripts and a manifest which defines the dependencies on other packages. A package can group different nodes which are executable that use the ROS client library to communicate over the ROS network with other nodes. That communication handshake is established by the master (*roscore*) which acts as a master that governs all the connections. Despite starting and ending all connections between nodes, *roscore* does not see the actual messages passing through as in a conventional router. The information exchange between nodes can be either by using (requesting) services or, the most popular, by passing messages. These messages are published into the topics (by the nodes), allowing other nodes to subscribe to them.

A significant step forward was done with the ROS package called

*rtt_ros_integration* [167] allowing Orocos components to connect to the ROS network, making possible for both middlewares to publish and to subscribe to all the available streamed topics. Moreover, custom typekits can be generated making possible the use of any ROS message type, may it be a predefined one from the ROS messages libraries or a user-defined one.

What is important is that while ROS has many tools and functionalities that are useful in the development of robotic applications, Orocos provides a solid core for the main control scheme in real time. In other words, ROS and Orocos complement each other, broadening the range of solutions they can offer as standalone platforms.

## 5.5. Ankle Rehabilitation Robot

The robot presented in Section 5.2 is the one used to show how the robot operates in order to carry out a rehabilitation of the lower limbs, in particular, the ankle. As can be seen in Fig. 5.6, four of the possible movements of the ankle are plantar/dorsiflexion, inversion and eversion, which are represented by gamma and beta.

On the other hand, several studies have determined the range of motion of the ankle [168]. Obviously, the maximum range is determined by each of the patients [169], so the exercise must be slightly different (position reference and/or force reference) depending on the patient who is being treated. A patient with a first degree ankle sprain is not the same as

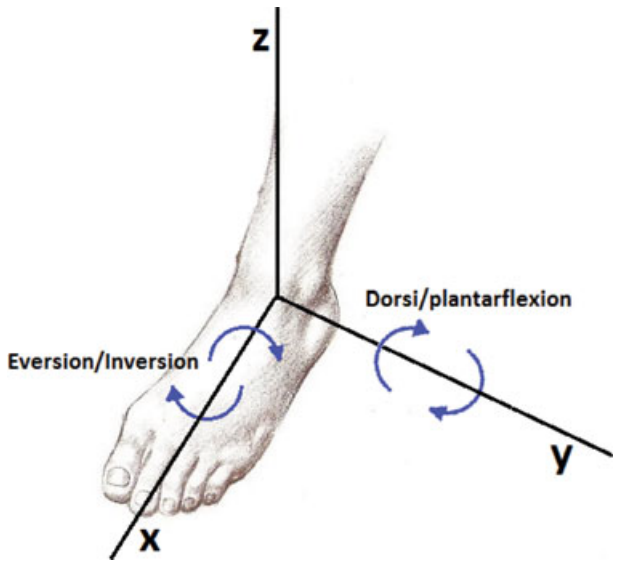



**Figure 5.6: Ankle movements**

another with a third degree ankle sprain. Table 5.2 shows the maximum allowable motion of the ankle, both in the X-axis (roll) and the Y-axis (pitch), and the maximum working range of the robot presented above.

| Type of movement | Max. human ankle | Max. parallel robot |
|---|---|---|
| Dorsiflexion | 20.3º-29.8º | 50º |
| Plantarflexion | 37.6º-45.8º | 50º |
| Eversion | 15.4º-25.9º | 50º |
| Inversion | 22º-36º | 50º |

**Table 5.2: Ankle and robot range of motion**

Table 5.3 shows the maximum torque for a passive movement of a human ankle, as well as the measured range for the force sensor used (described in Section 5.4.1).

Taking into account the specifications of the motors and the ball screw mentioned in Section 5.2, forces much greater than 800 N can be supplied for each of the three actuators. Considering the distance (577 mm)

| Type of movement | Max. human ankle passive mov | Max. force sensor |
|---|---|---|
| Dorsiflexion | –33.1 ± 16.5 N·m | ± 270 N·m |
| Plantarflexion | 40.1 ± 9.2 N·m | ± 270 N·m |
| Eversion | –48.1 ± 12.2 N·m | ± 270 N·m |
| Inversion | 34.1 ± 14.5 N·m | ± 270 N·m |

**Table 5.3: Ankle and force sensor range of torque**

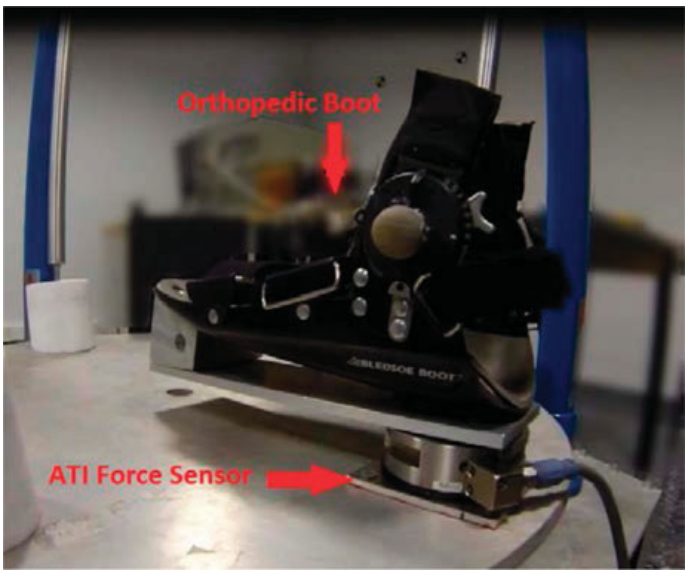



**Figure 5.7: Orthopedic boot**

between the spherical joints located on the mobile platform, the parallel robot can supply the range of torques required in order to reproduce human ankle movements.

The placement of the boot on the parallel robot platform can be seen in Fig. 5.7. This boot allows the patient's foot to be attached by Velcro strips (Fig. 5.8). Thus, because of the foot is attached correctly, the rehabilitation exercises can be performed properly. In addition, a force sensor has been placed at the base of the boot to monitor the forces and torques applied. The forces applied to the ankle are monitored using this force sensor,

**Figure 5.8: Foot attached to the orthopedic boot**

and several exercises to rehabilitate and strengthen injured ankles have been implemented.

The most common injuries are ankle sprains (representing 38% of locomotor system injuries) and these cause stretching or tearing of the ligaments due to a sudden movement in the direction of eversion [170]. In a large percentage of cases, sprains are not treated and rehabilitation is not performed. As a result, between 80% and 90% of sprains can become chronic if the injury is not correctly rehabilitated, leaving ligament instability (which increases over time as more injuries occur). Therefore, proper rehabilitation is necessary.

As for the different exercises that can be performed with this parallel robot, in order to repair or strengthen injured ankles these exercises can be passive or active. Passive exercises are performed without any voluntary movement by the patient, while active exercises are performed

with voluntary movement by the patient. In order to show how the robot operates, a series of exercises have been performed with healthy subjects.

### 5.5.1. Passive exercises with the parallel robot

In passive ankle exercises, the robot is programmed to follow a specific position reference prescribed by a specialist. Thus, using the low-cost parallel robot developed in this project, a number of references have been generated to rehabilitate an injured ankle.

Since the dynamic position control proposed by Paden–Panja has been used (see Table 5.1), the position error is around 0.5 mm, so the movement is very accurate.

This position reference implemented for a passive rehabilitation exercise consists of a sinusoidal signal at a roll frequency of 0.16 Hz, in order to exercise the plantar/dorsiflexion.

The reference and actual position of the roll of the platform can be seen in Fig. 5.9. The error is imperceptible, so the movement indicated by the specialist is done very accurately.

Furthermore, to follow a reference with high precision, the forces and torques applied to the ankle are being monitored all the time. Figure 5.10 shows the torque in the X-axis, Y-axis, and Z-axis. Since both plantarflexion and dorsiflexion have been trained in this exercise, the torque in the X–Z-axis is nearly zero, while the Y-axis has significant values. This indicates that the rehabilitation process has been successful.

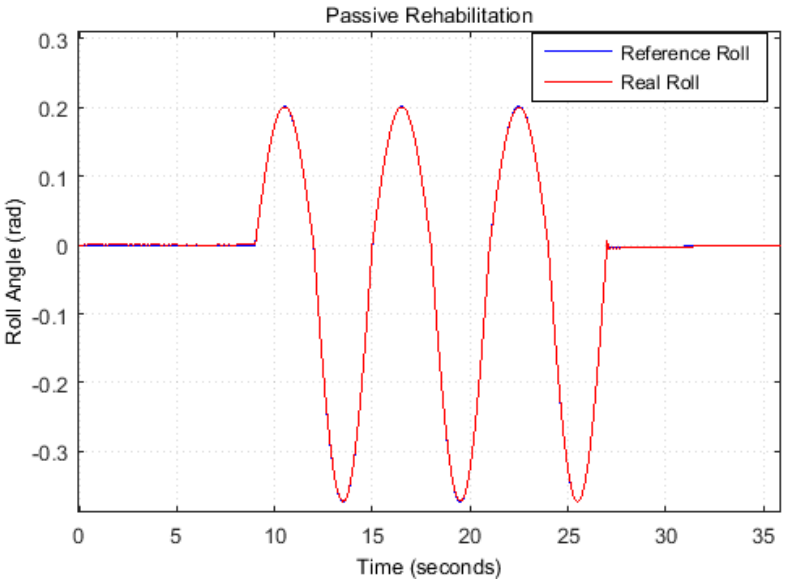



**Figure 5.9: Passive rehabilitation**

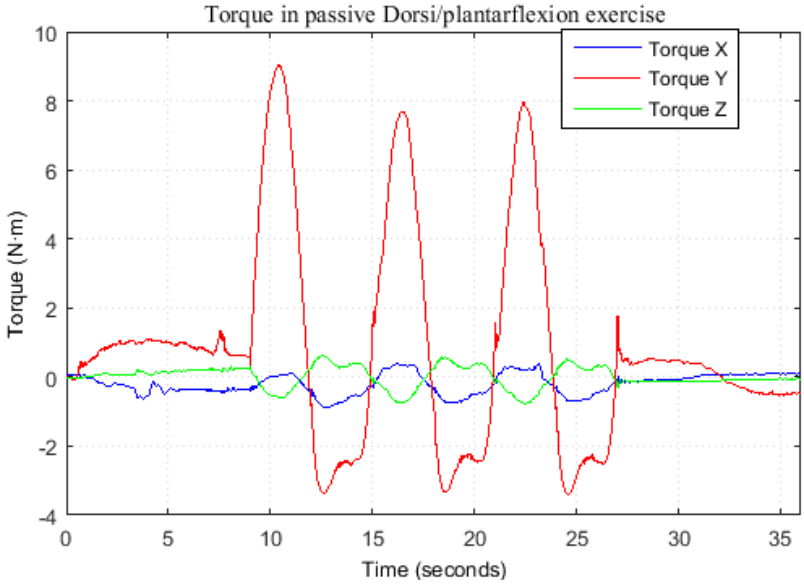



**Figure 5.10: Torque measured in passive rehabilitation**

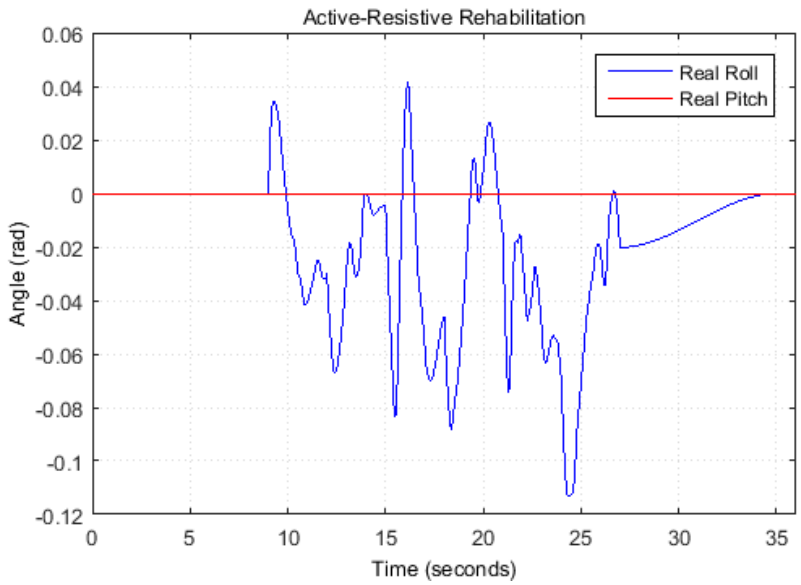



**Figure 5.11: Active-resistive rehabilitation**

### 5.5.2. Active-resistive exercises with the parallel robot

There are several types of active movements, such as strengthening or resistive, in which the patient has to overcome a resistance imposed by the specialist.

Specifically, a resistive application has been proposed in which the aim is to keep the platform of the parallel manipulator in a horizontal position, doing opposed torques to the motion of the platform. A low frequency sinusoidal position reference in roll (and thus strengthens plantarflexion and dorsiflexion) has therefore been implemented. Figures 5.11 and 5.12 below show the actual position of the robot (roll and pitch) and the torques measured at the patient's ankle.

As seen in the previous graphs, the patient, by the torques measured

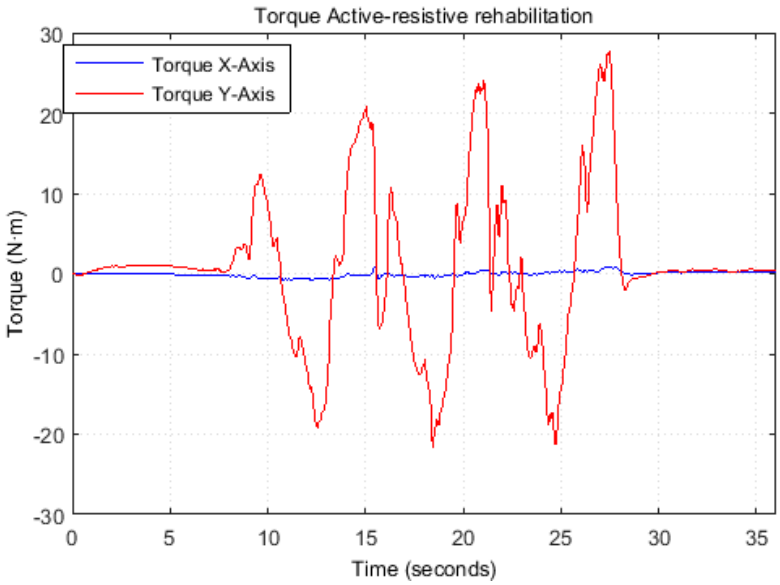



**Figure 5.12: Forces measured in active resistive rehabilitation**

in the ankle, is able to account for the sinusoidal reference. Thus, the platform is, generally, kept in a horizontal position ($roll = pitch = 0$).

### 5.5.3. Active-assistive exercises with the parallel robot

The main difference between active-assistive and active-resistive exercises is that in assistive movements the patient is not able to carry out the movement against gravity by him or herself. These types of exercise are usually done at an early stage in the rehabilitation process. For this reason, external help is needed to perform them correctly.

The assistive exercise developed is based on the fact that the platform assists the patient to perform the movement, depending on the torques applied by the injured ankle to the load cell.

Thus,

$$\vec{q}_{Newref} = \vec{q}_{ref} + K_{st} \cdot \vec{T}_{xy} \tag{5.10}$$

where $\vec{q}_{ref}$ is the position reference, $K_{st}$ is a stiffness constant, $\vec{T}_{xy}$ is the torque measured in the ankle and $\vec{q}_{Newref}$ is the position reference modified considering these two parameters. The position reference generated supports itself in a horizontal position ($Gamma = Beta = 0$) and at a constant height ($Z = constant$) and, depending on the forces read, a new reference position modified is generated dynamically, in order to assist the movement of the injured patient.

As can be seen in Figs. 5.13 and 5.14 above, the platform changes its orientation in accordance with the data read by the force sensor. Obviously, depending on the stage of the patient's rehabilitation, movement of the platform will be more or less sensitive to the force measurements made by the ankle.

### 5.5.4. Configuration of exercises for each patient using Orocos

As seen in the previous sections, several active and passive rehabilitation exercises have been implemented. One of the main disadvantages in rehabilitation robotics is the difficulty of health personnel learning how the robot works. Therefore, robots in rehabilitation clinics are very ex-

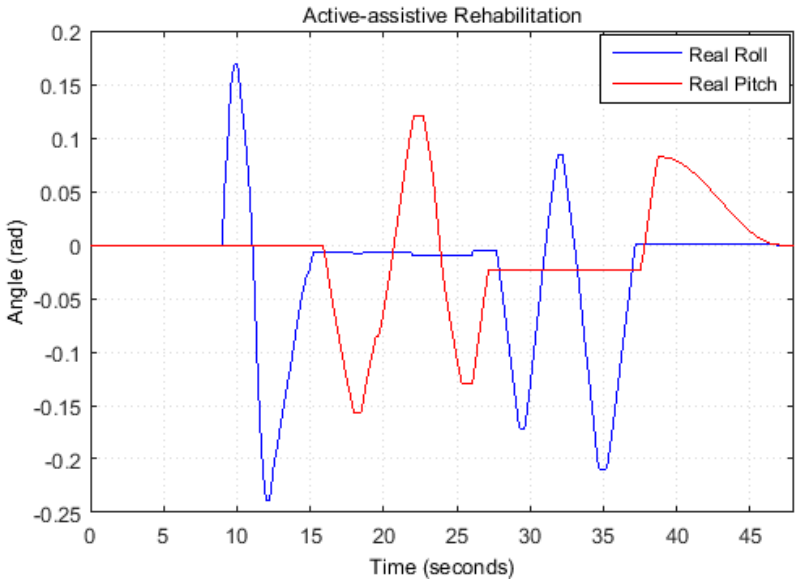



**Figure 5.13: Active-assistive rehabilitation**

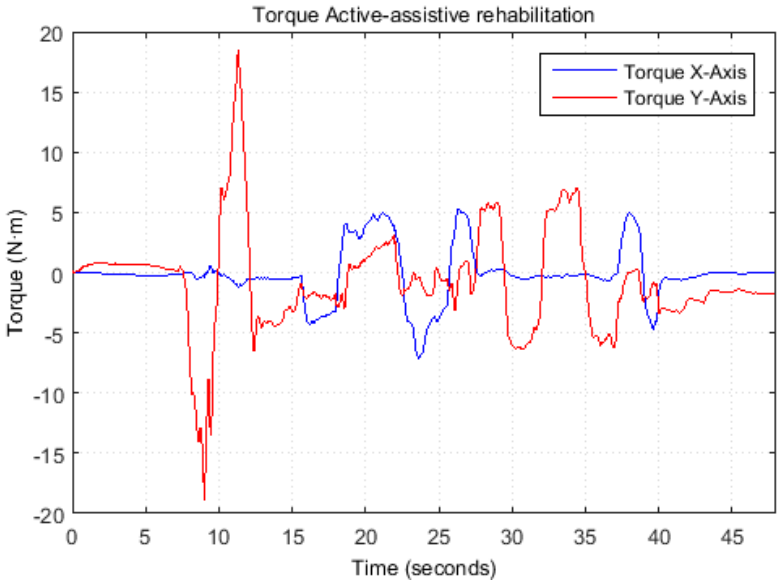



**Figure 5.14: Forces measured in Active-Assistive rehabilitation**

pensive and have limited functionality due to the manufacturer's own restrictions.

The low-cost robot presented in this article is controlled using the Orocos component-based middleware. One of the main advantages of Orocos is that the control scheme is developed modularly. Afterwards, once the modules are implemented, they are loaded into the system, configured according to the application the specialist wishes to perform, connected and executed.

This way, medical personnel do not need to have prior training to control the robot. All they do is select the rehabilitation exercise, set a number of parameters based on the patient (e.g., stiffness and movement exercise) and start the rehabilitation. The controller is the combination of all the parts corresponding to the modules required for accomplishing the selection made, as shown in Fig. 5.15.

Another important aspect of the modular implementation of the control system is the possibility of configuring and reconfiguring Orocos components at run-time. This is very useful, for example, in active-assistive exercises in which, depending on the severity of the injury, the robot must assist the patient more or less.

In experimental tests that have been performed, depending on the subject, the value of the stiffness constant in Eq. 5.10 should be changed in the rehabilitation module. Thus, this value is easily modified in the configuration phase, so a recompilation of the module is not required.

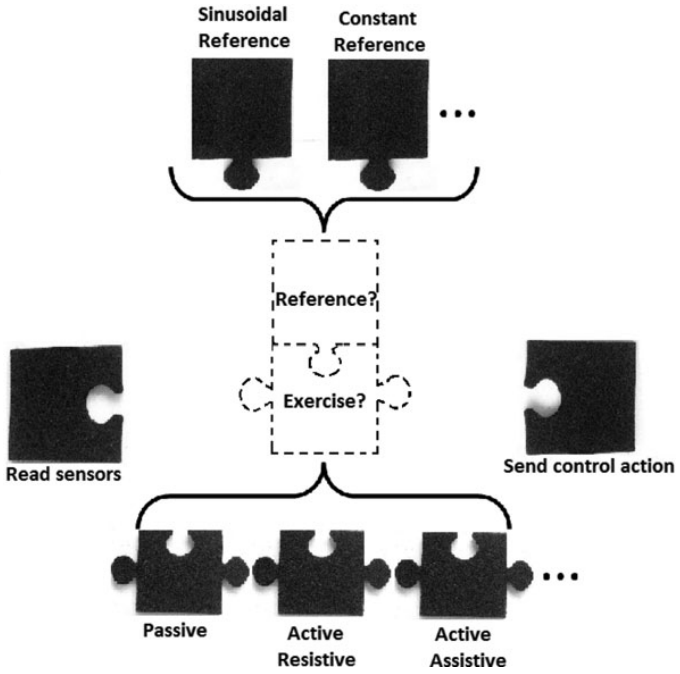



**Figure 5.15: Modular scheme**

### 5.5.5. Teleoperation and display of the parallel robot using ROS

Finally, using the middleware ROS, as well as integration with Orocos [167], an application has been developed for displaying and teleoperating the parallel robot. To perform teleoperation, a WiiMote has been used and, for displaying the robot in real-time (taking advantaje of the modular architecture), two different solutions for visualization have been proposed: Gazebo Simulator [30] and V-REP [171].

As can be seen in Fig. 5.16, the scheme is divided into two parts: Industrial PC (on the left hand side) and client PC-laptop (on the right hand side).

More specifically, in the industrial PC, the position control is running in

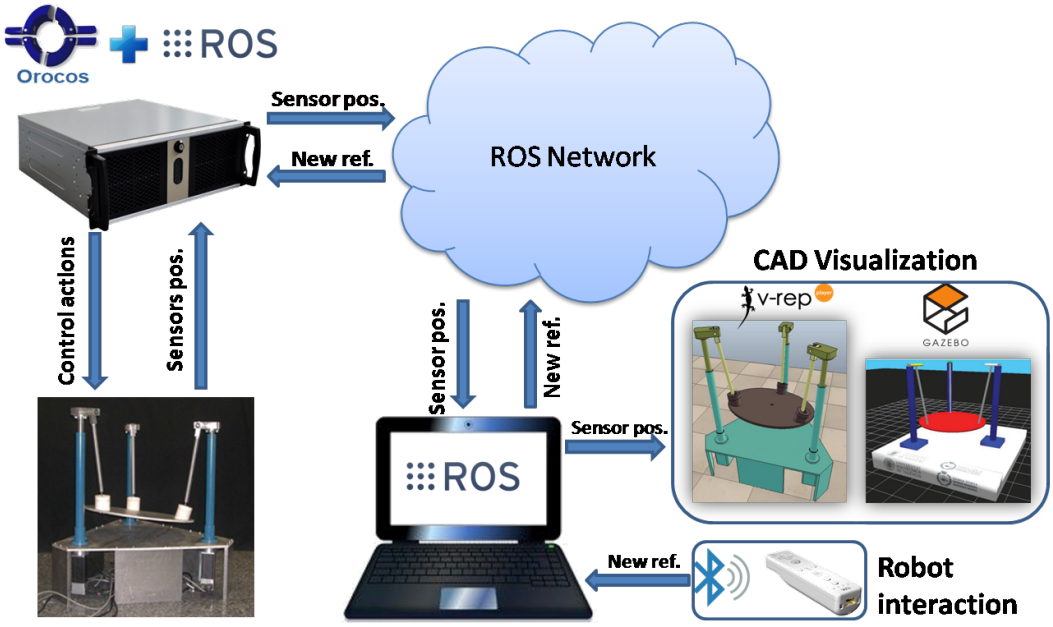



**Figure 5.16: Scheme application Orocos-ROS integration**

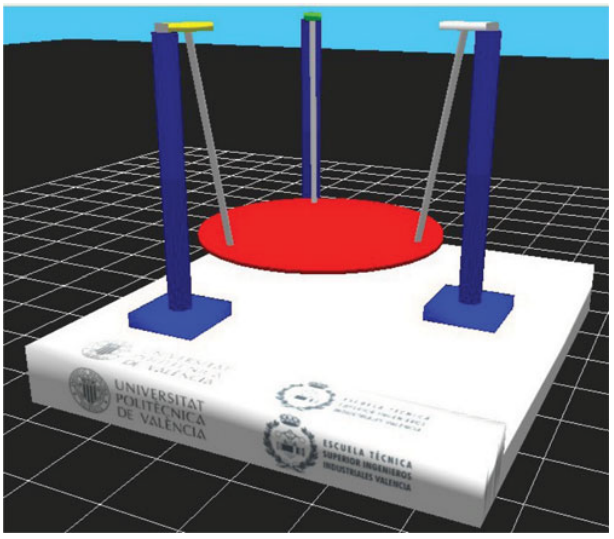



**Figure 5.17: Gazebo CAD model of the actual parallel robot**

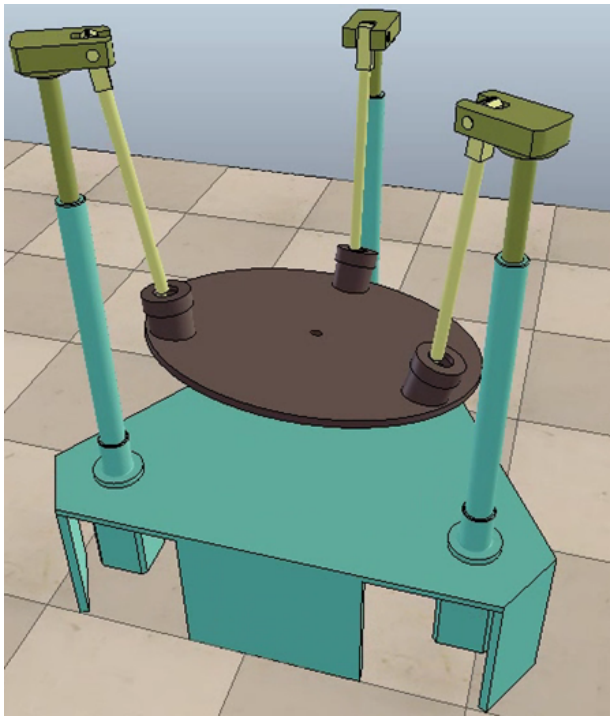


**Figure 5.18: V-REP CAD model of the actual parallel robot**

real time (using Orocos) at a frequency of 100 Hz. At each iteration (10 ms), the control action to apply to the three actuators is calculated. In addition to calculating and aplying the control action, the Orocos module is responsible for publishing the actual position of the robot (reading straight from the encoders) at each period on an ROS topic, making use of the Orocos-ROS integration.

At the same time, the module responsible for generating the trajectory for the parallel robot, receives from a ROS topic the information about the accelerometer and buttons from a remote device. This information is obtained, by means of a ROS message sent by the remote device (either a Wiimote or SmartPhone, for instance). This component, depending on the parameters of the ROS message received, modifies the desired reference

at the parallel robot. Because the sending of messages to this component is performed through a network, the generation of the reference is not achieved only with the value obtained from the remote device, but also with a local reference. So, when this module receives a data message, it modifies the current reference. This is critical as, if a problem occurs with the network or the remote device (Wiimote), the robot would remain in a stable state.

Meanwhile, in the client computer, there are two ROS nodes with different functionality. The first ROS node has the function of communicating with the Bluetooth device (Wiimote or Smartphone) and to publish the message on a ROS topic with information from the accelerometer and the pressed buttons. The frequency with which this ROS node runs is 100Hz, being its only function to fill and send the message with the remote device settings, in this case, a Wiimote. More specifically, the message sent from the WiiMote can be seen in the table 5.4.

| Type | Name |
|---|---|
| bool | button_a |
| bool | button_plus |
| bool | button_minus |
| bool | button_up |
| bool | button_down |
| bool | button_left |
| bool | button_right |
| bool | button_home |
| int32 | roll |
| int32 | pitch |

**Table 5.4: Message populated from the remote device**

The second ROS node reads from a topic the values of each of the three ro-

bot's legs actual position and sends them to the simulation/visualization framework. In this case, the visualization of the parallel robot CAD model has been performed using V-REP (Fig. 5.18) and Gazebo (Fig. 5.17). It has been done very easily, as both V-REP and Gazebo have the ROS integration available, so just subscribing to the topic, all the data is available for any purpose.

Using this model, a displaying in real-time of the parallel robot is performed. It should be noted that the values read by the node that implements the virtual model come directly from the real robot sensors. Thus, if an error occurs in the control or any unexpected failure, in this virtual model will still be seen exactly the current position of the actual robot. Furthermore, since the model only takes 3 integer data (one for each joint), the bandwidth required for their communication is very small. The ROS node only publishes the real values from each of the prismatic joint to the ROS network. Then, V-REP only has to subscribe to that topic and, taking advantage of kinematics, dynamics and collision detection modules, the motion of the CAD model can be represented.

| Network | Min (ms) | Max (ms) | Avg (ms) | S. Dev |
|---|---|---|---|---|
| Wired (inside) | 0.14 | 0.2 | 0.17 | 0.02 |
| Wireless (inside) | 1.5 | 299 | 21 | 54 |
| Wired (outside) | 44 | 46 | 44 | 0.1 |
| Wireless (outside) | 44 | 47 | 44 | 0.1 |

**Table 5.5: Connection latency between client and industrial computer**

It has been checked for proper function both wired and wireless networks,

and inside and outside the University network. The following Tab. 5.5 shows the connection latency (or delay in milliseconds) between de laptop used for displaying, and the industrial computer.

The best results obtained have been with a wired connection inside the University network. The latency is very small in this case and, due to the laptop obtaining the variables every 10 ms (100 Hz), the displaying works perfectly. In the other hand, using the wireless network inside the University, the latency average is 21 ms, but the standard deviation is 54 ms. Despite having done several changes so as to receive data every 50 ms (20 Hz), sometimes, when there is a data delay, the modelled robot stops for an instant, and the displayed image is not as fluid as in a wired connection. Finally, when the laptop is outside the University network, the latency is less than 50 ms, and the standard deviation is nearly zero. This is because the traffic and policies of a University network are very different from a conventional network. As before, some changes have been done to receive data every 50 ms (20Hz). In this case, with a lower rate than with a wired connection inside the University, the displaying is correct and fluid.

## 5.6. Conclusions

Despite the potential advantages of parallel manipulators for lower limb rehabilitation robots, they have not been used as much as in other fields of robotics. Specifically, different approaches of parallel robots for ankle rehabilitation were described in the introduction.

A low-cost ankle rehabilitation robot has been presented in this chapter. The kinematics and dynamics of the rehabilitation robot were developed and validated both in simulation and experimentally, achieving a high degree of accuracy for tracking position and force. Furthermore, due to the kinematic configuration of the robot, it can be adapted to the two relevant axes in ankle rehabilitation.

Different types of ankle rehabilitation exercises have been implemented and tested. Because of the versatility and adaptability of the system, it has been possible to perform several exercises with patients with different physical characteristics without the need to program the robot. These exercises include both passive and active exercises and, in the case of active ones, resistive and assistive exercises. For the latter, force control strategies have been implemented.

The results obtained indicate that these rehabilitation exercises train the specific part of the ankle correctly, following the advice of the physiotherapist.

Taking advantage of a component-based middleware (Orocos), the entire control system has been implemented modularly. So, different parameters such as the stiffness constant or position reference, in addition to the type of exercise, can be changed for each patient with no need for the medical staff to have any knowledge of robotics, meaning that one of the main problems of installing a robot in a clinic has been solved satisfactorily.

Finally, a novelty application related to the display and teleoperation of

the robot has been implemented using Orocos-ROS integration. This solution takes advantage of both middlewares: the capability to provide hard real-time operation of Orocos and the high level tools and functionalities of ROS. The Orocos-ROS integration has also been tested and validated with the parallel robot, being able to tele-operate it using a usual Bluetooth device such as Wiimote, and displaying it in real-time with V-REP. This novel solution using Orocos - ROS - V-REP communication, provides portability, scalability and the possibility to execute all the scheme in a distributed way.

This way, any kind of exercise can be performed and the medical staff is not required to be physically present with each patient, being able to monitor the performance of the robot in real time through the CAD model simulation.

# Parte IV

## Conclusiones generales

# Capítulo 6

## Conclusiones y trabajos futuros

En la presente Tesis, como se ha podido comprobar, en la parte final de cada uno de los capítulos se han expuesto las principales conclusiones de cada trabajo individualmente. En esta sección, se comentarán todas ellas en conjunto, así como los trabajos futuros que se van a realizar a partir del estado actual.

En primer lugar, se ha demostrado la eficacia del desarrollo de controladores robóticos en forma de componentes software independientes y reusables, siguiendo las metodologías de la Ingeniería del Software Basada en Componentes. Además, el hecho de haber escogido Orocos como *middleware* de control en tiempo real y orientado a componentes, ha permitido la ejecución en paralelo de diversos controladores avanzados, distribuyendo la carga computacional de una forma mucho menos compleja. Por otro lado, al haber diseñado los controladores de forma modular, también ha permitido poder reusar partes comunes a la hora de realizar nuevos

controladores. Gracias a esto, el grado de madurez de los controladores implementados ha incrementado, permitiendo la realización de pruebas experimentales (incluso cambiando dinámicamente entre controladores en mitad de una ejecución) de forma mucho más eficiente, en un menor tiempo, y presentando todo el sistema una mayor robustez y fiabilidad.

En segundo lugar, el modelo cinemático y dinámico de un robot paralelo con tres grados de libertad ha sido desarrollado, partiendo de un trabajo previo que consistió en la identificación de parámetros dinámicos del propio robot. A partir del modelo cinemático y dinámico, se han diseñado controladores complejos de forma modular, los cuales han sido testados inicialmente en un entorno de simulación y, posteriormente, integrados en el sistema de control del robot real. A pesar de que el error de posición es realmente pequeño usando este tipo de controladores, se ha comprobado que, cuando el modelo dinámico presenta alguna incertidumbre, este error aumenta significativamente. Es por este motivo por el que un controlador adaptativo ha sido desarrollado permitiendo, además, el estudio sobre cómo afectan cada una de esas incertidumbres de forma independiente (cuerpo rígido, fricción o la dinámica de los actuadores). Mediante este control adaptativo, se ha conseguido que el sistema sea capaz de ajustar dinámicamente, durante la ejecución del mismo, los parámetros desconocidos para así minimizar el error de posición.

A continuación, la inclusión de un sensor de fuerza de seis grados de libertad (acoplado en la base del robot) en el sistema de control ha permitido la creación de dos controladores de fuerza diferentes. En primer lugar, se ha conseguido diseñar un controlador estrictamente de fuerza, en el

que el robot es capaz de seguir un patrón de fuerza (en el eje vertical). También, al realimentar el bucle de control con las lecturas de fuerza, se ha sido capaz de implementar un controlador híbrido, en el que la referencia de posición se calcula de forma dinámica en función de las lecturas de fuerza. En cuanto a los resultados obtenidos experimentalmente, éstos demuestran la gran precisión en el seguimiento de una referencia de fuerza, así como la integración de las lecturas de fuerza en el bucle de control, interfiriendo en la referencia de posición previamente establecida.

En último lugar, a partir de los controladores desarrollados a lo largo de este trabajo, se ha conseguido construir una aplicación completa, consistente en la rehabilitación de tobillos. En esta aplicación completa, diseñada conjuntamente con personal biomédico, se han recreado diferentes ejercicios, en función de la gravedad de la lesión. Concretamente, un primer ejercicio ha consistido en la rehabilitación pasiva, en la que el robot realiza un patrón de movimiento previamente definido. A medida que se realiza este ejercicio, la lectura del sensor de fuerza está siendo monitorizada en tiempo real, pudiéndose identificar cualquier anomalía, y cancelar el ejercicio en el peor de los casos. Un segundo ejercicio ha consistido en la rehabilitación activa-asistiva, en la que es el paciente el que, a través de las fuerzas (en los tres ejes) aplicadas al sensor de fuerza, interfiere en la referencia de posición de la plataforma, modificando la altura, *roll* y *pitch*. En última instancia, se ha diseñado un ejercicio activo-resistivo, el cuál está prescrito para el fortalecimiento del tobillo. En este caso, es el paciente el que tiene que contrarrestar, mediante las fuerzas aplicadas por él mismo al sensor de fuerza, la posición de referencia de la platafor-

ma (poniéndose de manifiesto el controlador híbrido fuerza/posición).

Gracias a la integración de los *middlewares* ROS y Orocos, se ha conseguido controlar y visualizar (mediante un modelo CAD) de forma remota y en tiempo real, el robot paralelo presentado, facilitando el uso del mismo desde cualquier lugar y dispositivo con conexión a la red. Todo esto ha sido posible debido al desarrollo modular de componentes, el cual ha permitido la ejecución de distintos tipos de ejercicios de una forma completamente *plug&play*, transparente para el usuario final, y sin un conocimiento previo necesario para la realización de estos ejercicios de rehabilitación.

En cuanto a trabajos futuros, el camino a seguir es el relacionado con la rehabilitación, debido a la aceptación por la comunidad científica de trabajos sobre esta temática usando el robot previamente descrito, como se puede demostrar en *MEBIOMEC* [172] y [173]. Por otro lado, siguiendo el enfoque de componentes software, uno de los retos consistirá en el diseño de controladores para un nuevo robot paralelo de dimensiones reducidas y con cuatro grados de libertad (además de *roll*, *pitch* y altura, se le ha añadido el *yaw*), en el que se está trabajando para tratar de desarrollar el modelo cinemático y dinámico. La inclusión de este grado de libertad adicional permitirá la realización de ejercicios de abducción y aducción en el tobillo, un tipo de movimientos que con el robot paralelo de tres grados de libertad no se puede conseguir, proporcionando así un nuevo tipo de ejercicio. También, se pretende realizar un modelo CAD de este nuevo robot y, mediante la integración de *middlewares*, ser capaz de visualizarlo en tiempo real (utilizando *V-Rep*, *Gazebo* o *Rviz*), analizando el ancho de

banda necesario, todo ello con el objetivo de la posibilidad de tele-operar a distancia.

Para concluir, el hecho de introducir un prototipo real en el sector médico, y más concretamente en el de la rehabilitación, implica el conocimiento de numerosos aspectos relacionados a la fisiología humana en los que hay que trabajar, así como los diferentes casos de uso que pueden presentarse en el día a día de un profesional de este ámbito. La consecución de este objetivo sería, sin duda, una oportunidad excelente para que la comunidad científica pueda introducir la robótica en un sector en el que aún hay mucho que aportar.

# Bibliografía


[1] G. Philipson, "A Short History of Software," *Management, Labour Process and Software Development: Reality Bites*, p. 13, 2004.

[2] M. Hally, *Electronic brains: stories from the dawn of the computer age*. National Academies Press, 2005.

[3] R. Nimtz, "Development of the law of computer software protection," *J. Pat. Off. Soc'y*, vol. 31, p. 3, 1979.

[4] Crnkovic Ivica, Stafford Judith, and Szyperski Clemens, *Software Components beyond Programming: From Routines to Services*. 2011.

[5] C. Szyperski, D. Gruntz, and S. Murer, *Component Software: Beyond Object-oriented Programming*. Pearson Education, 2002.

[6] R. N. Taylor, N. Medvidovic, and E. M. Dashofy, *Software Architecture: Foundations, Theory, and Practice*. Wiley Publishing, 2009.

[7] A. Shakhimardanov, N. Hochgeschwender, and G. K. Kraetzschmar, "Component models in robotics software," *Proceedings of the 10th Performance Metrics for Intelligent Systems Workshop on - PerMIS '10*, p. 82, 2010.

[8] D. Brugali and P. Scandurra, "Component-based Robotic Engineering Part I: Reusable building blocks," *Robotics Automation Magazine*, vol. 16, no. 4, pp. 84–96, 2009.

[9] L. M. Hernández Acosta, *Componentes reutilizables para software robusto y cooperativo*. PhD thesis, Universidad de Las Palmas de Gran Canaria, 2006.

[10] C. Fung, "Architecture paradigms and their influences and impacts on component-based software systems," *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the*, vol. 00, no. C, p. 10 pp., 2004.

[11] A. W. Brown and K. C. Wallnau, "The current state of CBSE," *IEEE Software*, vol. 15, no. 5, pp. 37–46, 1998.

[12] A. Brooks, T. Kaupp, A. Makarenko, S. Williams, and A. Orebäck, "Towards component-based robotics," *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 3567–3572, 2005.

[13] R. C. Martin, "Granularity," *C++ Report*, pp. 1–12, 1996.

[14] C. Szyperski, "Component technology - what, where, and how?," *25th International Conference on Software Engineering, 2003. Proceedings.*, pp. 684–693, 2003.

[15] D. Brugali and A. Shakhimardanov, "Component-Based Robotic Engineering (Part II)," *Robotics Automation Magazine, IEEE*, vol. 17, no. 1, pp. 100–112, 2010.

[16] H. Bruyninckx, M. Klotzbücher, N. Hochgeschwender, G. Kraetzschmar, L. Gherardi, and D. Brugali, "The BRICS component model: a model-based development paradigm for complex robotics software systems," *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pp. 1758–1764, 2013.

[17] M. Radestock and S. Eisenbach, "Coordination in evolving systems," pp. 162–176, 1996.

[18] A. Shakhimardanov and E. Prassler, "Comparative evaluation of robotic software integration systems: A case study," *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 3031–3037, 2007.

[19] R. Glass, "The software-research crisis," *IEEE Software*, vol. 11, pp. 42–47, nov 1994.

[20] H. Leung and Z. Fan, "Software Cost Estimation," *Information and Software Technology*, vol. 34, no. 10, pp. 307–324, 2002.

[21] T. Wijayasiriwardhane, R. Lai, and K. Kang, "Effort estimation of component-based software development - a survey," *IET Software*, vol. 5, no. 2, p. 216, 2011.

[22] I. Crnkovic and M. Larsson, “Challenges of component-based development,” vol. 61, pp. 201–212, 2002.

[23] G. Kotonya and I. Sommerville, “Requirements engineering with viewpoints,” *Software Engineering Journal*, vol. 11, no. 1, p. 5, 1996.

[24] Z. Pan, J. Polden, N. Larkin, S. Van Duin, and J. Norrish, “Recent progress on programming methods for industrial robots,” *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 2, pp. 87–94, 2012.

[25] M. Ang, L. Wei, and Lim Ser Yong, “An industrial application of control of dynamic behavior of robots-a walk-through programmed welding robot,” *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 3, no. April, pp. 2352–2357, 2000.

[26] S. Deng, Z. Cai, D. Fang, H. Liao, and G. Montavon, “Application of robot offline programming in thermal spraying,” *Surface and Coatings Technology*, vol. 206, no. 19-20, pp. 3875–3882, 2012.

[27] L. Sciavicco and B. Siciliano, *Modelling and control of robot manipulators*. Springer Science & Business Media, 2012.

[28] T. Brogårdh, “Present and future robot control development-An industrial perspective,” *Annual Reviews in Control*, vol. 31, no. 1, pp. 69–79, 2007.

[29] Orocos Toolchain, “http://www.orocos.org/wiki/orocos/toolchain/.”

[30] B. P. Gerkey, R. T. Vaughan, and A. Howard, “The Player / Stage Project : Tools for Multi-Robot and Distributed Sensor Systems,” *Proceedings of the International Conference on Advanced Robotics (ICAR 2003)*, no. Icar, pp. 317–323, 2003.

[31] A. Makarenko, A. Brooks, and T. Kaupp, “Orca: Components for Robotics,” *International Conference on Intelligent Robots and Systems*, pp. 163–168, 2006.

[32] S. Fleury, M. Herrb, and R. Chatila, “G eno M : A Tool for the Speci cation and the Implementation of Operating Modules in a Distributed Robot Architecture 1 Introduction 2 A Network of Modules,” *In International Conference on Intelligent Robots and Systems*, pp. 842—-848, 1997.

[33] C. Schlegel and R. Wörz, "The Software Framework SMARTSOFT for Implementing Sensorimotor Systems," *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No.99CH36289)*, vol. 3, pp. 1610–1616, 1999.

[34] C. Schlegel, "Communication patterns as key towards component interoperability," *Springer Tracts in Advanced Robotics*, vol. 30, pp. 183–210, 2007.

[35] H. Utz, S. Sablatnög, S. Enderle, and G. Kraetzschmar, "Miro - Middleware for mobile robot applications," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 4, pp. 493–497, 2002.

[36] G. Metta, P. Fitzpatrick, and L. Natale, "YARP: Yet another robot platform," *International Journal of Advanced Robotic Systems*, vol. 3, no. 1, pp. 043–048, 2006.

[37] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, and W.-K. Yoon, "RT-Component Object Model in RT-Middleware," *Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA 2005*, pp. 457–462, 2005.

[38] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Mg, "ROS: an open-source Robot Operating System," *Icra*, vol. 3, no. Figure 1, p. 5, 2009.

[39] D. Alonso, J. Á. Pastor, P. Sánchez, B. Álvarez, and C. Vicente-Chicote, "Generación automática de software para sistemas de tiempo real: Un enfoque basado en componentes, modelos y frameworks," *RIAI - Revista Iberoamericana de Automatica e Informatica Industrial*, vol. 9, no. 2, pp. 170–181, 2012.

[40] Object Management Group. OMG, "http://www.omg.org."

[41] T. Stahl, M. Voelter, and K. Czarnecki, *Model-Driven Software Development: Technology, Engineering, Management*. John Wiley & Sons, 2006.

[42] E. Estevez and M. Marcos, "Model-Based Validation of Industrial Control Systems," *IEEE Transactions on Industrial Informatics*, vol. 8, pp. 302–310, may 2012.

[43] M. L. Alvarez, I. Sarachaga, A. Burgos, E. Estevez, and M. Marcos, "A Methodological Approach to Model-Driven Design and Development of Automation Systems," *IEEE Transactions on Automation Science and Engineering*, pp. 1–13, 2016.

[44] T. L. Johnson, D. L. Kerwplman, and H. A. Sutherland, "GRAFCET and SFC as Factory Automation Standards Advantages and limitions," *American Control Conference*, pp. Page(s): 1725 – 1730, 1987.

[45] G. Cloutier and J.-J. Paques, "GEMMA, the complementary tool of the GRAFCET," in *Fourth Annual Canadian Conference Proceedings., Programmable Control and Automation Technology Conference and Exhibition*, pp. 12A1–5/1–10, 1988.

[46] E. Estévez, A. Sánchez-García, J. Gámez-García, J. Gómez-Ortega, and S. Satorres-Martínez, "A novel model-driven approach to support development cycle of robotic systems," *International Journal of Advanced Manufacturing Technology*, vol. 82, no. 1-4, pp. 737–751, 2016.

[47] G. Booch, I. Jacobson, and J. Rumbaugh, *The unified modeling language*. Unix Review, 1996.

[48] XML Metadata Interchange Specification, "http://www.omg.org/spec/XMI/," 2014.

[49] E. Estévez, A. S. García, J. G. García, and J. G. Ortega, "Aproximación Basada en UML para el Diseño y Codificación Automática de," *Revista Iberoamericana de Automática e Informática Industrial*, vol. 14, no. 1, pp. 82–93, 2017.

[50] A. Bubeck, F. Weisshardt, and A. Verl, "BRIDE - A toolchain for framework-independent development of industrial service robot applications," *ISR/Robotik 2014, Joint Conference of 45th International Symposium on Robotics and 8th German Conference on Robotics.*, pp. 137–142, 2014.

[51] H. Bruyninckx, "Open robot control software: the OROCOS project," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, vol. 3, pp. 2523–2528, IEEE.

[52] I. Bonev, "The true origins of parallel robots," *ParalleMIC*, 2003.

[53] J. Gwinnett, “Amusement device,” 1931.

[54] W. Pollard, “Position-controlling apparatus,” 1964.

[55] V. E. Gough and S. G. Whitehall, “Universal Tyre Test Machine,” *FISITA Proceedings*, pp. 117–137, 1962.

[56] D. Stewart, “A platform with six degrees of freedom,” *Proceedings of the institution of mechanical engineers*, vol. 180, no. 1, pp. 371–386, 1965.

[57] R. Aracil, R. Saltarén, J. Sabater, and O. Reinoso, “Robots paralelos: Máquinas con un pasado para una robótica del futuro,” *Revista iberoamericana de automática e informática industrial*, vol. 3, no. 1, pp. 16–22, 2006.

[58] J.-P. Merlet, *Parallel Robots*, vol. 208. 2006.

[59] G. Pritschow, C. Eppler, and T. Garber, “Influence of the Dynamic Stiffness on the Accuracy of PKM,” in *3rd Chemnitzer Parallelkinematik Seminar, Chemnitz, April*, pp. 23–25, 2002.

[60] J. Wang and O. Masonry, “On the accuracy of a Stewart platform. Part I. The effect of manufacturing tolerances,” *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 1, pp. 114–120, 1993.

[61] S. Besnard, W. Khalil, and D. Noë, “Calibration of parallel robots using two inclinometers Ecole Centrale de Nantes,” no. May, pp. 0–5, 1999.

[62] J. Tlusty, J. Ziegert, and S. Ridgeway, “Fundamental Comparison of the Use of Serial and Parallel Kinematics for Machines Tools,” *CIRP Annals - Manufacturing Technology*, vol. 48, no. 1, pp. 351–356, 1999.

[63] J. Wu, J. Wang, and Z. You, “An overview of dynamic parameter identification of robots,” *Robotics and Computer-Integrated Manufacturing*, vol. 26, no. 5, pp. 414–419, 2010.

[64] M. Díaz-Rodríguez, V. Mata, and N. Farhat, “Dynamic parameter identification of parallel robots starting from the measurement of joints position and forces,” vol. 32, no. 2, pp. 119 – 125, 2009.

[65] M. Díaz-Rodríguez, V. Mata, Á. Valera, and Á. Page, "A methodology for dynamic parameters identification of 3-DOF parallel robots in terms of relevant parameters," *Mechanism and Machine Theory*, vol. 45, no. 9, pp. 1337–1356, 2010.

[66] A. Zubizarreta, I. Cabanes, M. Marcos, and C. Pinto, "Control of parallel robots using passive sensor data," *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2398–2403, 2008.

[67] F. Pierrot, C. Reynaud, and A. Fournier, "DELTA: a simple and efficient parallel robot," *Robotica*, vol. 8, p. 105, apr 1990.

[68] C. Connolly, "ABB high-speed picking robots establish themselves in food packaging," *Industrial Robot: An International Journal*, vol. 34, no. 4, pp. 281–284, 2007.

[69] B. Aponso, S. Beard, and J. Schroeder, "The NASA Ames Vertical Motion Simulator - A Facility Engineered for Realism," *Royal Aeronautical Society Spring 2009 Flight Simulation Conference London*, vol. 1, no. June, pp. 3–4, 2009.

[70] T. P. Jones and G. R. Dunlop, "Analysis of rigid-body dynamics for closed-loop mechanisms–its application to a novel satellite tracking device," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 217, pp. 285–298, jun 2003.

[71] M. Wapler, V. Urban, T. Weisener, J. Stallkamp, M. Durr, and A. Hiller, "A Stewart platform for precision surgery," *Transactions of the Institute of Measurement and Control*, vol. 25, no. 4, pp. 329–334, 2003.

[72] D. V. Amin and L. D. Lunsford, "Volumetric Resection Using the SurgiScope®: A Quantitative Accuracy Analysis of Robot-Assisted Resection," *Stereotactic and Functional Neurosurgery*, vol. 82, pp. 250–253, feb 2005.

[73] M. J. Girone, G. C. Burdea, and M. Bouzit, "The Rutgers Ankle.$^{o}$rthopedic Rehabilitation Interface," *Proc. of the ASME Haptics Symp.*, vol. 67, pp. 305–312, 1999.

[74] C. E. Syrseloudis, I. Z. Emiris, T. Lilas, and A. Maglara, "Design of a simple and modular 2-DOF ankle physiotherapy device relying on a hybrid serial-parallel

robotic architecture," *Applied Bionics and Biomechanics*, vol. 8, no. 1, pp. 101–114, 2011.

[75] P. K. Jamwal, S. Q. Xie, S. Hussain, and J. G. Parsons, "An adaptive wearable parallel robot for the treatment of ankle injuries," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 1, pp. 64–75, 2014.

[76] R. Aracil, R. J. Saltarén, and O. Reinoso, "A climbing parallel robot," *IEEE Robotics and Automation Magazine*, vol. 13, no. 1, pp. 16–22, 2006.

[77] R. Aracil, R. Saltarén, and O. Reinoso, "Parallel robots for autonomous climbing along tubular structures," *Robotics and Autonomous Systems*, vol. 42, no. 2, pp. 125–134, 2003.

[78] D. Brugali, A. Brooks, A. Cowley, C. Côté, A. C. Domínguez-Brito, D. Létourneau, F. Michaud, and C. Schlegel, "Trends in Component-Based Robotics," in *Software Engineering for Experimental Robotics*, pp. 135–142, Berlin, Heidelberg: Springer Berlin Heidelberg.

[79] M. Vallés, J. Cazalilla, A. Valera, V. Mata, and A. Page, "Dynamic controllers implementation based on the OROCOS middleware | Implementación basada en el middleware OROCOS de controladores dinámicos pasivos para un robot paralelo," *RIAI - Revista Iberoamericana de Automatica e Informatica Industrial*, vol. 10, no. 1, 2013.

[80] J. Cazalilla, M. Valles, M. Diaz-Rodriguez, V. Mata, A. Soriano, and A. Valera, "Implementation of dynamic controllers using real-time middleware for a low-cost parallel robot," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 47, pp. 2084–2084, IEEE, may 2014.

[81] P. Clements and M. Shaw, ""The golden age of software architecturerevisited," *IEEE Software*, vol. 26, no. 4, pp. 70–72, 2009.

[82] J. F. Tang, L. F. Mu, C. K. Kwong, and X. G. Luo, "An optimization model for software component selection under multiple applications development," *European Journal of Operational Research*, vol. 212, no. 2, pp. 301–311, 2011.

[83] A. T. Campbell, G. Coulson, and M. E. Kounavis, "Managing Complexity: Middleware Explained.," *IT Professional*, vol. 1, p. 22, 1999.

[84] T. H. J. Collett, B. a. MacDonald, and B. P. Gerkey, "Player 2.0: Toward a Practical Robot Programming Framework," *In Proceedings of the Australasian Conference on Robotics and Automation*, p. 8, 2005.

[85] C. Côté, Y. Brosseau, D. Létourneau, C. Raïevsky, and F. Michaud, "Robotic software integration using MARIE," 2006.

[86] T. Ionescu and C. W. Stammers, "Standardisation of Terminology," *Ionescu, Theodor and Stammers, Charles W*, vol. 38, pp. 597—-1111, 2003.

[87] F. Pierrot, V. Nabat, O. Company, S. Krut, and P. Poignet, "Optimal design of a 4-DOF parallel manipulator: From academia to industry," *IEEE Transactions on Robotics*, vol. 25, no. 2, pp. 213–224, 2009.

[88] R. CLAVEL, "Delta, a fast robot with parallel geometry," *Proc. 18th Int. Symp. on Industrial Robots, Lausanne, 1988*, pp. 91–100, 1988.

[89] Y. Li and Q. Xu, "Design and development of a medical parallel robot for cardiopulmonary resuscitation," *IEEE/ASME Transactions on Mechatronics*, vol. 12, no. 3, pp. 265–273, 2007.

[90] L. Tsai, "Robot analysis: the mechanics of serial and parallel manipulators," 1999.

[91] D. H. Kim, J.-Y. Kang, and K.-I. Lee, "Nonlinear Robust Control Design for a 6 DOF Parallel Robot," *KSME International Journal, Vol. 13, No. 7, pp. 557-568,1999*, vol. 13, no. 7, pp. 557–568, 1999.

[92] K. Fu and J. K. Mills, "Robust Control Design for a Planar Parallel Robot," vol. 22, no. 2, p. 2007, 2007.

[93] S. D. Stan, R. Balan, V. Maties, and C. Rad, "Kinematics and fuzzy control of ISOGLIDE3 medical parallel robot," *Mechanika*, vol. 75, no. 1, pp. 62–66, 2009.

[94] H. Guo, Y. Liu, G. Liu, and H. Li, "Cascade control of a hydraulically driven 6-DOF parallel robot manipulator based on a sliding mode," *Control Engineering Practice*, vol. 16, pp. 1055–1068, sep 2008.

[95] H. Abdellatif and B. Heimann, "Advanced model-based control of a 6-DOF hexapod robot: A case study," *IEEE/ASME Transactions on Mechatronics*, vol. 15, no. 2, pp. 269–279, 2010.

[96] D. Chablat and P. Wenger, "Architecture optimization of a 3-DOF translational parallel mechanism for machining applications, the orthoglide," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 3, pp. 403–410, 2003.

[97] K. M. Lee and S. Arjunan, "A Three-Degrees-of-Freedom Micromotion In-Parallel Actuated Manipulator," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 5, pp. 634–641, 1991.

[98] J. Jalón and E. Bayo, "Kinematic and dynamic simulation of multibody systems," *Mechanical Engineering Series, Springer, ...*, p. 452, 1994.

[99] M. Grotjahn, B. Heimann, and H. Abdellatif, "Identification of Friction and Rigid-Body Dynamics of Parallel Kinematic Structures for Model-Based Control," *Multibody System Dynamics*, vol. 11, pp. 273–294, apr 2004.

[100] M. Díaz-Rodríguez, V. Mata, N. Farhat, and S. Provenzano, "Identifiability of the Dynamic Parameters of a Class of Parallel Robots in the Presence of Measurement Noise and Modeling Discrepancy," *Mechanics Based Design of Structures and Machines*, vol. 36, pp. 478–498, nov 2008.

[101] R. Ortega and M. W. Spong, "Adaptive motion control of rigid robots: A tutorial," *Automatica*, vol. 25, no. 6, pp. 877–888, 1989.

[102] B. PADEN and R. PANJA, "Globally asymptotically stable 'PD+' controller for robot manipulators," *International Journal of Control*, vol. 47, no. 6, pp. 1697–1712, 1988.

[103] J. Cazalilla, M. Vallés, V. Mata, M. Díaz-Rodríguez, and A. Valera, "Adaptive control of a 3-DOF parallel manipulator considering payload handling and relevant parameter models," *Robotics and Computer-Integrated Manufacturing*, vol. 30, pp. 468–477, oct 2014.

[104] M. Moradi Dalvand and B. Shirinzadeh, "Motion control analysis of a parallel robot assisted minimally invasive surgery/microsurgery system (PRAMiSS)," *Robotics and Computer-Integrated Manufacturing*, vol. 29, pp. 318–327, apr 2013.

[105] D. Pisla, B. Gherman, C. Vaida, M. Suciu, and N. Plitea, "An active hybrid parallel robot for minimally invasive surgery," *Robotics and Computer-Integrated Manufacturing*, vol. 29, pp. 203–221, aug 2013.

[106] J.-P. Merlet, "Still a long way to go on the road for parallel mechanisms," *Proc. ASME Int. Mech. Eng. Congress and Exhibition*, no. December, pp. 95–99, 2002.

[107] F. Paccot, N. Andreff, and P. Martinet, "A Review on the Dynamic Control of Parallel Kinematic Machines: Theory and Experiments," *The International Journal of Robotics Research*, vol. 28, no. 3, pp. 395–416, 2009.

[108] W. Khalil and E. Dombre, *Modeling, identification and control of robots*, vol. 56. 2004.

[109] Yang Zhiyong and Huang Tian, "A new method for tuning PID parameters of a 3 DoF reconfigurable parallel kinematic machine," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, pp. 2249–2254 Vol.3, IEEE, 2004.

[110] F. X. Wu, W. J. Zhang, Q. Li, and P. R. Ouyang, "Integrated Design and PD Control of High-Speed Closed-loop Mechanisms," *Journal of Dynamic Systems, Measurement, and Control*, vol. 124, no. 4, p. 522, 2002.

[111] C. Yang, Q. Huang, and J. Han, "Decoupling control for spatial six-degree-of-freedom electro-hydraulic parallel robot," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 1, pp. 14–23, 2012.

[112] P. L. Yen and C. C. Lai, "Dynamic modeling and control of a 3-DOF Cartesian parallel manipulator," *Mechatronics*, vol. 19, no. 3, pp. 390–398, 2009.

[113] W. Shang and S. Cong, "Nonlinear computed torque control for a high-speed planar parallel manipulator," *Mechatronics*, vol. 19, no. 6, pp. 987–992, 2009.

[114] S. Jiang, "Modeling, identification and control of a redundant planar 2-Dof parallel manipulator," *International Journal of Control Automation and System*, vol. 5, pp. 559–569, 2007.

[115] H. S. Kim, Y. M. Cho, and K.-I. Lee, "Robust nonlinear task space control for 6 DOF parallel manipulator," *Automatica*, vol. 41, pp. 1591–1600, sep 2005.

[116] H. Abdellatif, J. Kotlarski, T. Ortmaier, and B. Heimann, *Practical Model-based and Robust Control of Parallel Manipulators Using Passi vity and Sliding Mode Theory*. Robotics, 2010.

[117] M. Honegger, A. Codourey, and E. Burdet, "Adaptive control of the Hexaglide, a 6 dof parallel manipulator," in *Proceedings of International Conference on Robotics and Automation*, vol. 1, pp. 543–548, 1997.

[118] W. Shang and S. Cong, "Nonlinear adaptive task space control for a 2-DOF redundantly actuated parallel manipulator," *Nonlinear Dynamics*, vol. 59, pp. 61–72, jan 2010.

[119] W.-W. Shang, S. Cong, and Y. Ge, "Adaptive computed torque control for a parallel manipulator with redundant actuation," *Robotica*, vol. 30, pp. 457–466, may 2012.

[120] M. Diaz-Rodriguez, A. Valera, V. Mata, and M. Valles, "Model-Based Control of a 3-DOF Parallel Robot Based on Identified Relevant Parameters," *IEEE/ASME Transactions on Mechatronics*, vol. 18, pp. 1737–1744, dec 2013.

[121] M. Vallés, M. Díaz-Rodríguez, Á. Valera, V. Mata, and Á. Page, "Mechatronic Development and Dynamic Control of a 3-DOF Parallel Manipulator," *Mechanics Based Design of Structures and Machines*, vol. 40, pp. 434–452, oct 2012.

[122] D. S. BAYARD and J. T. WEN, "New class of control laws for robotic manipulators Part 2. Adaptive case," *International Journal of Control*, vol. 47, pp. 1387–1406, may 1988.

[123] J. Cazalilla, M. Vallés, Á. Valera, V. Mata, and M. Díaz-Rodríguez, "Hybrid force/position control for a 3-DOF 1T2R parallel robot: Implementation, simulations and experiments," *Mechanics Based Design of Structures and Machines*, vol. 44, pp. 16–31, apr 2016.

[124] J. Cazalilla, M. Vallés, A. Valera, V. Mata, and M. Díaz-Rodríguez, "Implementation of Force and Position Controllers for a 3DOF Parallel Manipulator," in *Mechanisms and Machine Science*, vol. 25, pp. 359–369, 2015.

[125] R. Cao, F. Gao, Y. Zhang, D. Pan, and W. Chen, "A New Parameter Design Method of a 6-DOF Parallel Motion Simulator for a Given Workspace," *Mechanics Based Design of Structures and Machines*, vol. 43, pp. 1–18, jan 2015.

[126] S. Zarkandi, "Kinematics and Singularity Analysis of a Parallel Manipulator with Three Rotational and One Translational DOFs," *Mechanics Based Design of Structures and Machines*, vol. 39, pp. 392–407, jul 2011.

[127] J. A. Carretero, R. P. Podhorodeski, M. A. Nahon, and C. M. Gosselin, "Kinematic Analysis and Optimization of a New Three Degree-of-Freedom Spatial Parallel Manipulator," *Journal of Mechanical Design*, vol. 122, no. 1, p. 17, 2000.

[128] J. N. Pires, G. Afonso, and N. Estrela, "Force control experiments for industrial applications: a test case using an industrial deburring example," *Assembly Automation*, vol. 27, no. 2, pp. 148–156, 2007.

[129] A. Garg, C. S. Vikram, S. Gupta, M. K. Sutar, P. M. Pathak, N. K. Mehta, A. K. Sharma, and V. K. Gupta, "Design and Development of In Vivo Robot for Biopsy," *Mechanics Based Design of Structures and Machines*, vol. 42, pp. 278–295, jul 2014.

[130] G. Zeng and A. Hemami, "An overview of robot force control," *Robotica*, vol. 15, no. 5, pp. 473–482, 1997.

[131] S. Bellakehal, N. Andreff, Y. Mezouar, and M. Tadjine, "Force/position control of parallel robots using exteroceptive pose measurements," *Meccanica*, vol. 46, pp. 195–205, feb 2011.

[132] N. Rosillo, A. Valera, F. Benimeli, V. Mata, and F. Valero, "Realtime solving of dynamic problem in industrial robots," *Industrial Robot: An International Journal*, vol. 38, no. 2, pp. 119–129, 2011.

[133] N. Farhat, V. Mata, A. Page, and F. Valero, “Identification of dynamic parameters of a 3-DOF RPS parallel manipulator,” *Mechanism and Machine Theory*, vol. 43, no. 1, pp. 1–17, 2008.

[134] J. J. Craig, “Introduction to Robotics: Mechanics and Control 3rd,” 2004.

[135] Spong, “Robot dynamics and control,” *Automatica*, vol. 28, no. 3, pp. 655–656, 1992.

[136] T. Yoshikawa, *Foundations of Robotics: Analysis and control*. MIT press, 1990.

[137] C. C. de Wit, B. Siciliano, and G. Bastin, *Theory of robot control*. Springer Science & Business Media, 2012.

[138] R. Volpe and P. Khosla, “A theoretical and experimental investigation of explicit force control strategies for manipulators,” *IEEE Transactions on Automatic Control*, vol. 38, no. 11, pp. 1634–1650, 1993.

[139] K. J. Astrom and R. M. Murray, *Feedback systems: an introduction for scientists and engineers*. Princeton university press, 2010.

[140] A. Valera, F. Benimeli, J. Solaz, H. De Rosario, A. Robertsson, K. Nilsson, R. Zotovic, and M. Mellado, “A car-seat example of automated anthropomorphic testing of fabrics using force-controlled robot motions,” *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 2, pp. 280–291, 2011.

[141] M. Vallés, J. Cazalilla, Á. Valera, V. Mata, Á. Page, and M. Díaz-Rodríguez, “A 3-PRS parallel manipulator for ankle rehabilitation: towards a low-cost robotic rehabilitation,” *Robotica*, pp. 1–19, mar 2015.

[142] Y. D. Patel and P. M. George, “Parallel Manipulators Applications - A Survey,” *Mordern Mechanical Engineering*, vol. 2, no. August, pp. 57–64, 2012.

[143] I. Díaz, J. J. Gil, and E. Sánchez, “Lower-Limb Robotic Rehabilitation: Literature Review and Challenges,” *Journal of Robotics*, vol. 2011, no. i, pp. 1–11, 2011.

[144] A. J. Del-Ama, A. D. Koutsou, J. C. Moreno, A. De-los Reyes, N. Gil-Agudo, and J. L. Pons, “Review of hybrid exoskeletons to restore gait following spinal cord

injury," *The Journal of Rehabilitation Research and Development*, vol. 49, no. 4, p. 497, 2012.

[145] G. Colombo, M. Joerg, R. Schreier, and V. Dietz, "Treadmill training of paraplegic patients using a robotic orthosis," *Journal of rehabilitation research and development*, vol. 37, no. 6, pp. 693–700, 2000.

[146] S. Hesse and D. Uhlenbrock, "A mechanized gait trainer for restoration of gait.," *Journal of rehabilitation research and development*, vol. 37, no. 6, pp. 701–708, 2000.

[147] M. Peshkin, D. A. Brown, J. J. Santos-Munné, A. Makhlin, E. Lewis, J. E. Colgate, J. Patton, and D. Schwandt, "KineAssist: A robotic overground gait and balance training device," in *Proceedings of the 2005 IEEE 9th International Conference on Rehabilitation Robotics*, vol. 2005, pp. 241–246, 2005.

[148] C. Schmitt, P. Métrailler, A. Al-Khodairy, R. Brodard, J. Fournier, M. Bouri, and R. Clavel, "The MotionMaker: a Rehabilitation System Combining an Orthosis With Closed Loop Electrical Muscle Stimulation," in *8 Vienna International Workshop on Functional Electrical Stimulation*, pp. 117–120, 2004.

[149] A. E. Q. Van Delden, C. E. Peper, G. Kwakkel, and P. J. Beek, "A systematic review of bilateral upper limb training devices for poststroke rehabilitation," 2012.

[150] K. Bharadwaj and T. G. Sugar, "Kinematics of a robotic gait trainer for stroke rehabilitation," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2006, pp. 3492–3497, 2006.

[151] G. B. Prange, M. J. a. Jannink, C. G. M. Groothuis-Oudshoorn, H. J. Hermens, and M. J. Ijzerman, "Systematic review of the effect of robot-aided therapy on recovery of the hemiparetic arm after stroke.," *Journal of rehabilitation research and development*, vol. 43, no. 2, pp. 171–184, 2006.

[152] E. Rocon, A. F. Ruiz, J. L. Pons, J. M. Belda-Lois, and J. J. Sánchez-Lacuesta, "Rehabilitation robotics: A wearable exo-skeleton for tremor assessment and suppression," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2005, pp. 2271–2276, 2005.

[153] J. A. Saglia, N. G. Tsagarakis, J. S. Dai, and D. G. Caldwell, "Control strategies for patient-assisted training using the ankle rehabilitation robot (ARBOT)," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 6, pp. 1799–1808, 2013.

[154] Y.-H. Tsoi, S. Q. Xie, and A. E. Graham, "Design, Modeling and Control of an Ankle Rehabilitation Robot," in *Design and Control of Intelligent Robotic Systems*, pp. 377–399, Berlin, Heidelberg: Springer Berlin Heidelberg.

[155] C. E. Syrseloudis, I. Z. Emiris, C. N. Maganaris, and T. E. Lilas, "Design framework for a simple robotic ankle evaluation and rehabilitation device.," *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, vol. 2008, pp. 4310–4313, 2008.

[156] J. S. Dai, T. Zhao, and C. Nester, "Sprained Ankle Physiotherapy Based Mechanism Synthesis and Stiffness Analysis of a Robotic Rehabilitation Device," *Autonomous Robots*, vol. 16, no. 2, pp. 207–218, 2004.

[157] J. Yoon, J. Ryu, and K.-B. Lim, "Reconfigurable ankle rehabilitation robot for various exercises," *Journal of Robotic Systems*, vol. 22, pp. S15–S33, sep 2006.

[158] C. E. Syrseloudis and I. Z. Emiris, "A parallel robot for ankle rehabilitation-evaluation and its design specifications," *8th IEEE International Conference on BioInformatics and BioEngineering, BIBE 2008*, 2008.

[159] Y. Fan and Y. Yin, "Mechanism design and motion control of a parallel ankle joint for rehabilitation robotic exoskeleton," in *2009 IEEE International Conference on Robotics and Biomimetics, ROBIO 2009*, pp. 2527–2532, 2009.

[160] C. Z. Wang, Y. F. Fang, S. Guo, and C. C. Zhou, "Design and kinematic analysis of redundantly actuated parallel mechanisms for ankle rehabilitation," *Robotica*, vol. 33, no. 2, pp. 366–384, 2015.

[161] F. Patanè and P. Cappa, "A 3-DOF parallel robot with spherical motion for the rehabilitation and evaluation of balance performance," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 19, no. 2, pp. 157–166, 2011.

[162] J.-J. E. Slotine and Weiping Li, "On the Adaptive Control of Robot Manipulators," *The International Journal of Robotics Research*, vol. 6, pp. 49–59, sep 1987.

[163] N. Sadegh and R. Horowitz, "Stability and Robustness Analysis of a Class of Adaptive Controllers for Robotic Manipulators," *The International Journal of Robotics Research*, vol. 9, no. 3, pp. 74–92, 1990.

[164] B. Siciliano and L. Villani, *Robot Force Control*, vol. 540. Springer Science & Business Media, 2012.

[165] N. Hogan, "Impedance Control: An Approach to Manipulation," *American Control Conference, 1984 IS - SN - VO -*, no. March, pp. 304–313, 1985.

[166] Ziren Lu and A. A. Goldenberg, "Robust Impedance Control and Force Regulation: Theory and Experiments," *The International Journal of Robotics Research*, vol. 14, pp. 225–254, jun 1995.

[167] R. Smits and H. Bruyninckx, "Composition of complex robot applications via data flow integration," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5576–5580, 2011.

[168] J. Chen, S. Siegler, and C. D. Schneck, "The three-dimensional kinematics and flexibility characteristics of the human ankle and subtalar joint–Part I.," 1988.

[169] M. Dettwyler, A. Stacoff, I. Kramers-De Quervain, and E. Stassi, "Modelling of the ankle joint complex. Reflections with regards to ankle prostheses," 2004.

[170] M. R. Safran, R. S. Benedetti, A. R. Bartolozzi, and B. R. Mandelbaum, "Lateral ankle sprains: a comprehensive review: part 1: etiology, pathoanatomy, histopathogenesis, and diagnosis.," *Medicine and science in sports and exercise*, vol. 31, no. 7 Suppl, pp. S429–37, 1999.

[171] E. Rohmer, S. P. N. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1321–1326, IEEE, nov 2013.

[172] Mebiobec, "https://mebiomec.ai2.upv.es/."

[173] A. Valera, M. Díaz-Rodríguez, M. Valles, E. Oliver, V. Mata, and A. Page, "Controller-observer design and dynamic parameter identification for model-based control of an electromechanical lower-limb rehabilitation system," *International Journal of Control*, pp. 1–13, sep 2016.