

BOTBLOQ: Ecosistema integral para el diseño, fabricación y programación de robots DIY

Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI)

EXPEDIENTE: IDI-20150289

Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020

ACRÓNIMO DEL PROYECTO: BOTBLOQ



Centro para el
Desarrollo
Tecnológico
Industrial



UNIÓN EUROPEA
Fondo Europeo de
Desarrollo Regional (FEDER)
Una manera de hacer Europa

ENTREGABLE E.4.3 PROCESO METODOLÓGICO PARA EL DISEÑO DE UN ROBOT MODULAR.

RESUMEN DEL DOCUMENTO

En el presente documento se define el proceso a seguir para el diseño de un robot modular en función de un conjunto de requisitos de entrada proporcionados por el usuario, incluyendo una descripción de las herramientas que deben ser creadas para la creación del diseño. Dentro de los distintos modos de diseño, nos hemos centrado en el problema de diseño automático, al ser el más novedoso y complejo de llevar a cabo.

Índice

1. Introducción	1
1.1. Motivación	1
1.2. Objetivos	3
2. Modos de diseño	5
2.1. Generación predefinida	5
2.2. Generación manual	5
2.3. Generación semi-automática	6
3. Metodología	9
3.1. Esquema de la metodología	9
3.2. Parámetros de entrada	10
3.3. Estructura abstracta del robot	10
3.4. Requisitos/parámetros físicos	11
3.5. Estructura física del robot	12
3.6. Controladores	12
3.7. Revisión del diseño	13
4. Ontología	14
4.1. Ontología para el diseño automático de robots	14
4.2. Generación de estructuras abstractas mediante ontologías	16
5. Generación de la estructura física	19
5.1. Parametrización de módulos	19
5.1.1. Optimización de módulos pasivos	20
5.2. Análisis dinámico de requisitos	22
5.2.1. Objetivos	23
5.2.2. Planteamiento	24
5.3. Caso de estudio: Generación automática de la estructura física de un manipulador	25
6. Generación automática de controladores	30
6.1. Generación automática de modelos de robots a partir de archivos Xacro parametrizables	31

6.2. Controlador para robots con ruedas	31
6.3. Controlador para robots serpiente	32
6.4. Controlador para robots manipuladores	32
6.5. Controlador para robots humanoides	34
7. Anexo A. Ontologías y representación del conocimiento	38
7.1. Conceptos de relevancia	38
7.2. Procesamiento de conocimiento	38
7.2.1. Ejemplo: KnowRob	39
7.3. Representación de Conocimiento: Ontologías	39
7.3.1. Lenguajes	39
7.3.2. SUMO	40
7.3.3. IEEE Standard Ontologies for Robotics and Automation . .	41
7.4. Knowledge reasoning: Métodos de extracción de conocimiento desde Ontologías	43
7.4.1. Consultas semánticas	43
7.4.2. Razonadores semánticos	43
7.4.3. Sigma	43
7.4.4. pySUMO	44
7.5. Conclusion	44
8. Anexo B. Optimización de eslabones	46
8.1. Formulación y resolución del problema	46
8.2. Programación matemática no lineal (PNL)	48
8.3. Casos de estudio. Solución para el cilindro hueco	49
8.3.1. Restricción adicional: mínimo espesor de impresión	55
9. Anexo C. Estudio dinámico de eslabones	61
10. Anexo D. Análisis de fallo de módulos pasivos	66
10.1. Criterio de Von Mises	66
10.2. Concentrador de tensiones	68
10.2.1. Diseño y cálculo de concentrador de tensiones	68
10.2.2. Simulación de concentrador de tensiones	71

1. Introducción

Este documento forma parte de la documentación asociada al Paquete de Trabajo número 4 (PT4) del proyecto BOTBLOQ. El PT4 tiene como objetivo el diseño, la fabricación y el control de robots modulares. En los anteriores entregables se han realizado:

E.4.1. Informe detallado del estado del arte. Este documento contiene una clasificación de los robots modulares preexistentes y un análisis de los requisitos exigibles a una arquitectura robótica modular con el fin de definir las funcionalidades que debe reunir para que resulte satisfactoria según los objetivos del proyecto.

E.4.2. Diseño de la arquitectura modular. En este entregable se presentaron los módulos confeccionados para la implementación de estructuras robóticas, incluyendo características mecánicas, eléctricas, electrónicas y lógicas. En él también se describen los modos de generación permitidos (manual, predefinido y semi-automático), y se muestran ejemplos de robots configurables (predefinidos) con esta arquitectura.

En el presente documento se definirán las etapas a seguir para el diseño y programación de un robot en función de un conjunto de requisitos de entrada proporcionados por el usuario, incluyendo una descripción de las herramientas creadas para la consecución del diseño, centrándonos en el problema de diseño automático, al ser el más novedoso y exigente.

1.1. Motivación

¿Cómo se diseña y controla un robot para que haga correctamente una determinada tarea? Este es el problema fundamental que se plantea en cualquier ámbito de la robótica (desde robótica industrial hasta robótica de servicio).

La manera tradicional para abordar este problema consiste en que ingenieros diseñen primero el *hardware*, es decir la estructura mecánica junto con los sensores y actuadores necesarios, y posteriormente diseñen el *software* o el control del robot. Como resultado de años de desarrollo, podemos encontrar diferentes configuraciones de robots (p.ej. manipuladores, robots con ruedas, humanoides,etc)

consideradas “correctas” para realizar las tareas encomendadas (p.ej. manufacturing, exploration, service, etc.)

Otra manera consiste en que, especificada una tarea, el diseño y/o el control del robot se obtenga “automaticamente”. Por ejemplo, existe un área de investigación denominada Evolutionary Robotics (ER) en donde mediante mediante algoritmos evolutivos se busca salvar las "limitaciones" de un diseño humano obteniendo mecanismos y controles complejos sin necesidad de tener un modelo explícito de ellos. A menudo se utilizan plataformas robóticas modulares para implementar estos diseños automáticos. Una completa revisión de esta línea de investigación en ER puede encontrarse en [16, 38]. Cabe separar dos líneas claras dentro de ER referentes a la obtención del diseño y control. Por un lado encontramos trabajos en los que dicho cálculo se realiza mediante simulaciones utilizando algoritmos evolutivos. En este caso los robots son posteriormente fabricados utilizando diferentes técnicas de prototipado. Por otro lado encontramos los *swarm self-assembly robots* [25] donde los propios robots se ensamblan automáticamente y aprenden a controlarse en entornos cambiantes. Para este tipo de robots se utilizan a menudo arquitecturas modulares, como es el caso de los Sambots [41].

En si, los robots modulares representan otra área de investigación de la robótica largamente estudiada desde finales de los 80 como se resume en [39, 21]. Los robots modulares presentan la ventaja de poder, en teoría, adoptar cualquier diseño deseado. Por eso, además de su uso en investigación (p.ej. Diseño automático) son utilizados en otro ámbitos como manipulación [10], y robótica educativa [20].

El proyecto BOTBLOQ pretende acercar la robótica a ámbitos donde hasta ahora no ha estado presente, creando herramientas hardware y software que permitan a usuarios de diferentes niveles el desarrollo de proyectos robóticos. Además, mediante la filosofía DIY (Do-It-Yourself) se pretende proporcionar a un colectivo de la sociedad lo más amplio posible la capacidad de diseñar, fabricar y programar robots para cualquier propósito, dentro de las limitaciones inherentes a nuestra plataforma.

El amplio colectivo objetivo nos obliga a dividir la plataforma en tres niveles de complejidad que se reflejan en tres modos de generación distintos, aunque toda la filosofía se ve integrada dentro de la misma metodología de diseño, fabricación y programación de robots. Aunque las estructuras robóticas modulares basadas en DIY tienen un gran auge entre los desarrolladores, existiendo algunas plata-

formas en centros de investigación y universidades de todo el mundo, éstas son, o bien demasiado concretas para un público muy determinado [3], o bien demasiado complejas de controlar y programar [41, 29, 39].

Gran parte del problema se debe a la falta de un diseño de alto nivel que incorpore las rutinas necesarias para el control de los robots no sólo desde un punto de vista procedural, sino también funcional. Por ejemplo, no sólo son necesarias las funciones de control que permitan mover los actuadores de los robots, sino otras funciones que les permitan realizar tareas más complejas, como caminar o dar una patada.

Por ello se hace necesario desarrollar una metodología integral de diseño que incluya tanto el diseño mecánico/eléctrico del dispositivo hardware, como elementos lógicos (rutinas de alto nivel) que permitan ejecutar las capacidades del robot abstrayendo las especificidades de la estructura física concreta (p.ej. una función 'caminar' que permita que un robot humanoide camine con independencia de algunos parámetros del robot como la longitud de sus piernas).

1.2. Objetivos

El objetivo principal de este documento es describir la metodología mencionada en el apartado anterior. Dicha metodología, detallada en la Sección 3, debe establecer:

- Los pasos a seguir en el proceso de diseño para cada modo de generación.
- La interacción que el proceso de diseño necesita con el usuario.
- Los elementos necesarios para fabricar y programar los robots diseñados.
- El proceso de fabricación y programación de los robots diseñados.

Esta metodología debe acompañarse de un sistema artificial inteligente capaz de diseñar robots conociendo los elementos disponibles que proporciona una arquitectura modular de robots. Por tanto, se deberán alcanzar los siguientes objetivos específicos:

1. A partir de las entradas del usuario es necesario determinar partes estructurales (atributos) y comportamientos (acciones) que el robot debe tener.

2. Compatibilidad de nuestro sistema con el estándar de Ontologías en Robótica y Automatización desarrollado por el IEEE (ORA).
3. Extracción de información desde la ontología (mediante búsquedas y consultas) y diseño del robot utilizando la información obtenida.
4. Interacción con el usuario en el caso de que haya ambigüedades (varias opciones posibles) en la generación del diseño y para realizar una parametrización del robot elegido (dimensiones, capacidades específicas, etc.).
5. Generación de archivos de las partes imprimibles del robot.
6. Creación de archivos de descripción del robot que sean utilizables por un entorno de control para la programación de las acciones que debe realizar.
7. Generación de controladores que puedan ser integrados en BOTBLOQ para ejecutar comportamientos en el robot (la interfaz de integración se definirá en el Entregable 4.6 de este mismo Paquete de Trabajo).

Para todo ello se utilizarán lenguajes de representación de conocimiento (SUKI [33]) y extracción de información (pySUMO [7]), generación paramétrica de elementos mecánicos (OpenSCAD [6]) y descripción de robots en entornos de control (concretamente *Robot Operating System*, ROS [8]).

2. Modos de diseño

Se permitirán tres modos de diseño¹ según el control que desee tener el usuario del proceso de creación del robot: generación predefinida, semi-automática (o paramétrica) y manual. De este modo se pretende hacer accesible la plataforma a un amplio espectro de público, desde alumnos de enseñanza primaria hasta investigadores en el campo de la robótica. Todos los diseños serán programables mediante la interfaz gráfica BOTBLOQ que está siendo desarrollada en el PT3 del presente proyecto, y que se basa a su vez en la IDE Bitbloq [2], la cual utiliza programación por bloques y está diseñada para uso educativo, pero también podrá ser programada directamente en interfaces de programación clásica similares a la IDE de Arduino. A continuación se definen en detalle los distintos modos de diseño mencionados.

2.1. Generación predefinida

El modelo completo del robot, incluyendo su estructura abstracta, su estructura física, estará definido de antemano. Junto con él, los bloques de programación de BOTBLOQ también estarán definidos de antemano (p.ej. en un robot industrial, mover el extremo en línea recta).

El usuario podrá elegir de una librería el tipo de robot que desea generar y, opcionalmente, proporcionar una serie de parámetros de configuración para determinar características escalables de ese tipo. Por ejemplo, en el caso de un robot serpiente se podría elegir el número de servos con que se desea construir.

2.2. Generación manual

Un usuario podrá crear desde cero un robot seleccionando los bloques básicos de diseño de una librería y ajustando sus parámetros manualmente (p.ej. bloque actuador). También podrá elegir elementos funcionales completos (p.ej. pierna de 5 servos). Para ello se generará una interfaz que permita realizar dicha construcción y seleccionar las propiedades de aquellos módulos que sean parametrizables, tales como los módulos pasivos (eslabones).

¹Aunque los modos de diseño ya fueron especificados en la Sección 3 del Entregable 4.2, serán definidos en más profundidad en este entregable.

Botbloq en este caso no creará bloques de programación asociados a funcionalidades del robot, sino que el propio usuario podrá realizar la programación desde bajo nivel (p.ej. mover articulación 2). Los bloques necesarios para programar los robots generados manualmente estarán incorporados a las librerías actuales de Bitbloq, (p.ej. Servo), incluidas cualesquiera nuevas funcionalidades debidas a sensores o actuadores adicionales que no formasen parte de las versiones actuales de Bitbloq y se hayan incorporado en el diseño de la arquitectura modular.

2.3. Generación semi-automática

En muchas ocasiones, los usuarios con pocos conocimientos fácilmente pueden imaginar qué capacidades les gustaría que tuviese su robot (un robot que sea capaz de jugar a algún juego, un robot que traiga el periódico, etc.). Sin embargo, aun incluso disponiendo de una arquitectura robótica modular que podría permitir su construcción, realizar el diseño del mismo no es una tarea al alcance de cualquiera. Es por ello que una parte de nuestro proyecto se centra en crear una aplicación que sea capaz de diseñar diferentes tipos de robots por sí misma en base a los deseos del usuario.

Es el modo de generación más complejo de desarrollar, puesto que el robot se diseñará completamente en base a las tareas que el usuario desee que realice el robot y a los requisitos que éste imponga en su funcionamiento. El usuario determinará, por ejemplo, que desea generar un robot que pueda jugar al fútbol. Para conseguirlo, la aplicación tendrá que interpretar el texto (escrito en lenguaje natural) y generar un robot utilizando la información inferida de ese texto. Esta información se pasará a través de una base de datos de conocimiento que determine las implicaciones morfológicas que tiene esa tarea:

- *Debe ser capaz de desplazarse*, por lo que necesitará una base móvil. El sistema inteligente daría la elección al usuario de crear un robot con ruedas o patas.
- *Debe poder golpear una pelota*, por lo que también necesitará un mecanismo de golpeo. En caso de ser un robot con patas, las propias patas podrían servir. Si se solicitó un diseño con ruedas, sería necesario añadir un mecanismo adicional.

- *Debe detectar el objeto que tiene que golpear*, por lo que además necesitará un sistema de percepción, ya sea un sistema de visión, un micrófono para detectar un sonido proveniente de la pelota o un sistema de localización basado en sensores ultrasonido, por ejemplo. Este sistema puede necesitar información adicional, como el color de la pelota, el sonido que produce o su tamaño para poder identificarla.

Todos estos términos deben estar incluidos en una ontología basada en el IEEE Standard of Robotics and Automation (ORA) definidos en el documento [13].

Además de la generación del robot, se añadirán una serie de parámetros/requisitos opcionales que puedan ser indicadas por el usuario en caso de así desecharlo, tales como la velocidad del movimiento del robot, o el tamaño que se desea tener. Una vez generado el robot, en BOTBLOQ se crearán bloques de tarea como programas de comportamientos de modo que el robot pueda ser programado de la manera más sencilla e intuitiva posible, y se asociarán esos bloques a la secuencia de acciones que define un comportamiento.

Los diseños paramétricos generados por nuestro sistema son optimizados en volumen mediante análisis de resistencia de materiales para luego ser fabricados con el menor coste de material mediante sistemas de prototipado, p.ej. impresoras 3D. En este sentido nuestro trabajo está más cercano a trabajos de diseño automático donde el resultado del algoritmo de diseño son nuevos modelos de estructuras físicas a ser fabricadas como por ejemplo en [36] que a los trabajos relaciones con self-assembly modular robots en el que los propios robots modulares se reconfiguran para obtener nuevos diseños funcionales como en [26].

Las imágenes de la Figura 1 muestran, a modo de ejemplo, robots reales construidos con nuestra arquitectura modular cuyo proceso automático de diseño y control será introducido en los siguientes apartados.

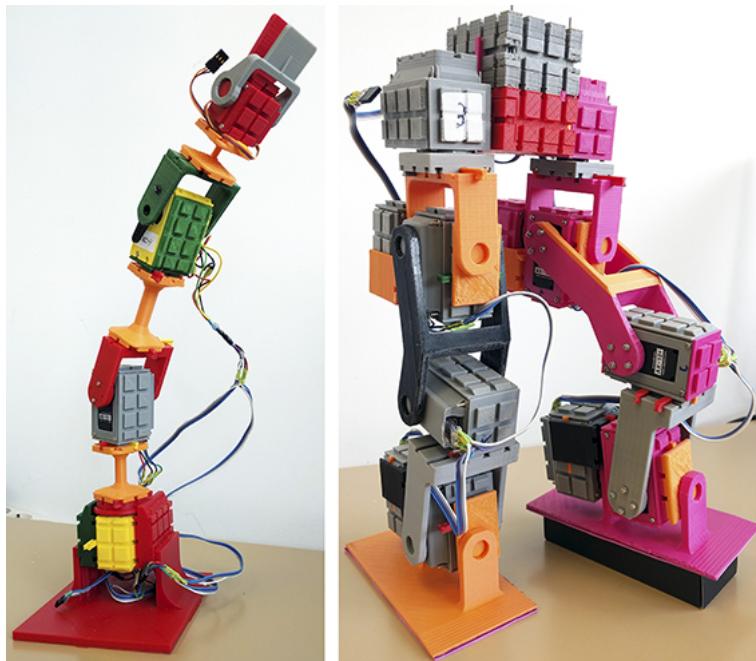


Figura 1: a) Manipulador de 3gdl. b) Robot bípedo de 12gdl.

3. Metodología

Esta sección describe en detalle el proceso metodológico utilizado para el método de generación semi-automática puesto que es el más complejo de desarrollar. Sin embargo también se indicará qué pasos de diseño se incluyen en cada modo de generación.

3.1. Esquema de la metodología

La metodología de diseño necesitará de una serie de herramientas que se mencionan a continuación y se describen en las diferentes secciones del documento. Estos son los pasos a seguir en la creación de un robot mediante nuestra arquitectura modular:

Definición de parámetros de entrada. Los parámetros de entrada o especificaciones diferirán dependiendo del modo de diseño elegido.

Creación de la estructura abstracta. Búsqueda de una configuración básica del robot que satisfaga las especificaciones (p.ej. humanoide, carro móvil o serpiente). La estructura abstracta consiste en una instancia de la ontología definida para la aplicación, en la cual se incluyen las partes funcionales del robot diseñado sin indicar aún explícitamente sus propiedades físicas cuantitativas.

Inclusión de requisitos y parámetros. Los parámetros son datos adicionales, generalmente opcionales ya que existen valores por defecto especificados, para la generación de la estructura del robot (p.ej. número de módulos de un robot serpiente), mientras que los requisitos son restricciones añadidas por el usuario que debe cumplir el robot (p.ej. velocidad mínima).

Creación de la estructura física. Los parámetros y requisitos indicados en el paso anterior se utilizan en la generación de la estructura física del robot, tanto en su descripción mediante ficheros URDF (Unified Robot Description Files) como en la generación de ficheros de modelado STL (STereo Lithography) que se utilizarán para imprimir las piezas físicas en impresoras 3D.

Creación de los controladores lógicos. Cada robot genera una serie de bloques preprogramados para realizar tareas complejas de un modo sencillo y fácilmente exportable a la aplicación BOTBLOQ.

Revisión del diseño por el usuario. Una vez finalizado el proceso el usuario puede decidir si está satisfecho con el diseño obtenido o si prefiere realizar algún cambio.

En las siguientes secciones del documento se detalla cada una de estas etapas, describiendo las herramientas desarrolladas para una correcta realización de las mismas.

3.2. Parámetros de entrada

Los parámetros de entrada son proporcionados por el usuario del sistema de diseño y sirven como punto de partida para la generación del robot. Dependiendo del modo de diseño, consistirán en:

- **Generación manual:** Puesto que en este caso el usuario diseña el robot completamente, tanto el hardware (estructura física del robot) como el software (descripción del robot y controladores), no es necesario especificar ningún parámetro de entrada en este modo.
- **Generación semi-automática:** El usuario indica las tareas/comportamientos que desea que el robot sea capaz de desempeñar (p.ej.: correr).
- **Generación predefinida:** En este caso el usuario se limita a seleccionar un robot de entre una galería de robots predefinidos. El resto del proceso de diseño es bastante limitado, puesto que estos robots, (salvo por algunos parámetros opcionales), están completamente definidos de antemano (p.ej. humanoide básico).

3.3. Estructura abstracta del robot

Para el modo de generación manual, no es necesario generar una estructura abstracta, puesto que el usuario definirá completamente la estructura del robot.

Podría seleccionarse sin embargo una estructura abstracta predefinida como apoyo para el diseño.

En el caso del modo de generación predefinido, la estructura abstracta viene impuesta por el robot elegido de la galería de robots predefinidos. En algunos casos (p.ej. número de ruedas de un carro móvil) puede haber alguna ambigüedad que se resuelve en el siguiente apartado de introducción de requisitos o parámetros.

Sin embargo, para generar la estructura abstracta del robot en el modo semi-automático necesitamos deducir qué robot es el que puede realizar la tarea o comportamiento requeridos a nuestro robot. Para ello se ha creado una ontología basada en el estándar definido en [13], el cual sienta las bases para la construcción de ontologías en el campo de la Robótica y la Automática. Dicha ontología se halla definida en el lenguaje SUO-KIF, y describe la taxonomía de los robots que pueden ser construidos mediante nuestra arquitectura modular y de sus partes funcionales. Igualmente describe la taxonomía de las acciones que pueden ser llevadas a cabo por un sistema robótico y las dependencias que tienen dichas acciones con respecto a partes funcionales del mismo.

La ontología se describe en detalle en la Sección 4, donde se muestran varios ejemplos de estructuras abstractas (instancias de la ontología) de robots predefinidos, mientras que una introducción a los conceptos relacionados con las ontologías se puede encontrar en el Anexo 7.

3.4. Requisitos/parámetros físicos

Una vez se dispone de una estructura abstracta, se determinan parámetros físicos necesarios para la definición de los componentes imprimibles mediante sistemas de prototipado rápido.

Requisitos. Especifican una restricción que condiciona la estructura mecánica/eléctrica del robot. Ejemplos de requisitos son:

- Velocidad a la que debe desplazarse un carro móvil.
- Máxima carga transportada por un manipulador industrial.
- Altura del escalón que debe ser capaz de subir un robot humanoide.

Parámetros. Especifican un valor necesario para definir completamente la estructura física del robot. Tienen valores por defecto que se aplican en caso de no ser especificados. Ejemplos de parámetros son:

- Longitud de las piernas de un humanoide.
- Número de módulos del cuerpo de un robot serpiente.
- Número de ejes de un carro móvil.

La sección 5.2 describe el efecto de los requisitos dinámicos sobre el robot y cómo se determina una solución válida con los módulos disponibles en nuestra arquitectura.

3.5. Estructura física del robot

La creación de la estructura física del robot involucra la creación de dos tipos de elementos:

- Ficheros URDF de descripción de los robots y sus características físicas.
- Ficheros STL de modelado 3D para impresión de las piezas mecánicas.

La Sección 5.1.1 explica cómo se diseñan los eslabones pasivos en base a un proceso de minimización del volumen de material utilizado, mientras que la Sección 6.1 describe el proceso de generación de los modelos software del robot (en formato URDF).

3.6. Controladores

La creación de controladores para generar el movimiento en los robots es un problema arduo y con una casuística tremadamente extensa. Nuestra metodología propone la creación de controladores parametrizables para cada uno de los robots de las configuraciones básicas, de modo que los comportamientos puedan describirse a partir de las acciones básicas de movimiento/medida que pueda desarrollar cada robot, basándonos en el conjunto de sensores y actuadores de que disponga.

Para ello, nuestro sistema utiliza por un lado el potencial de ROS [8] y de herramientas ya existentes en él, como puede ser el paquete *MoveIt!*, a la vez que

incluye otros métodos propios (p.ej. caminada de robots con patas de diferentes tamaños) y que generan unos controladores basados en las librerías de cinemática y dinámica (KDL) que incorpora ROS. Como alternativa a este diseño de control parametrizable, nuestro enfoque podría ser utilizado con configuraciones obtenidas por algoritmos evolutivos en un futuro.

3.7. Revisión del diseño

Una vez mostrada la estructura completa del robot y sus bloques de programación, se le plantea la posibilidad al usuario de realizar una revisión del diseño. En caso de no estar satisfecho con el mismo, el usuario puede volver al comienzo del proceso para corregir aquellas características que no sean de su agrado.

4. Ontología

Las ontologías, en un sentido computacional, consisten en especificaciones formales y explícitas de conceptualizaciones [23], y proporcionan conceptos y relaciones suficientes para articular modelos de situaciones concretas en un dominio dado. Consecuentemente, para construir modelos reales necesitamos una ontología que detalle la conceptualización compartida por los expertos en el campo de la robótica.

Recientemente se ha definido un estándar del IEEE denominado IEEE Standard Ontology for Robotics and Automation (ORA) [13]. Define una ontología básica que permite la representación, el razonamiento y la comunicación de conocimiento dentro del dominio de Robótica y Automatización (R&A), y se ha construido a partir de la ontología de alto nivel denominada SUMO (*Suggested Upper Merged Ontology*), la cual se halla definida en lenguaje SUO-KIF y puede ser consultada en [30]. ORA contiene el núcleo de la ontología, denominado CORA (Core Ontology for Robotics and Automation), que incluye los conceptos fundamentales en el dominio de la Robótica y la Automatización, así como sus definiciones, atributos, restricciones y relaciones. Estos aspectos se usan para construir conceptos más específicos pertenecientes a subontologías (p.ej. los conceptos relativos a sistemas de referencia, orientación y posición se incluyen en la subontología RPOS). La construcción de ontologías basadas en ORA siguen una metodología particular (METHONTOLOGY) definida también en el estándar.

4.1. Ontología para el diseño automático de robots

La ontología utilizada en el proyecto BOTBLOQ, denominada ADROn (*Automatic Design of Robots Ontology*), ha sido desarrollada a partir del estándar ORA utilizando la metodología propuesta en el mismo. ADROn define conceptos y relaciones adicionales para funcionar como una herramienta para el diseño automático de robots utilizando información inferida del lenguaje natural. Por continuidad con el origen, los nuevos conceptos se han desarrollado en inglés.

Los principales conceptos de ADROn son *Structural Robot Part* y *Module*, los cuales son subclases de *Device* y *Artifact* en SUMO, respectivamente. Ambas pueden ser instancias de *Robot Part* en CORA. Una instancia de *Structural Robot Part* representa cualquier dispositivo o conjunto de dispositivos que es parte de la

estructura del robot y juega un papel importante en un comportamiento concreto del robot.

Por ejemplo, *Robot leg* es una parte de la estructura de un robot y es esencial para caminar o correr. Una instancia de *Module* será cualquier artefacto (activo o pasivo) que sea parte del robot (p.ej. sensores, actuadores, etc.). La Figura 2 muestra las relaciones entre estos axiomas.

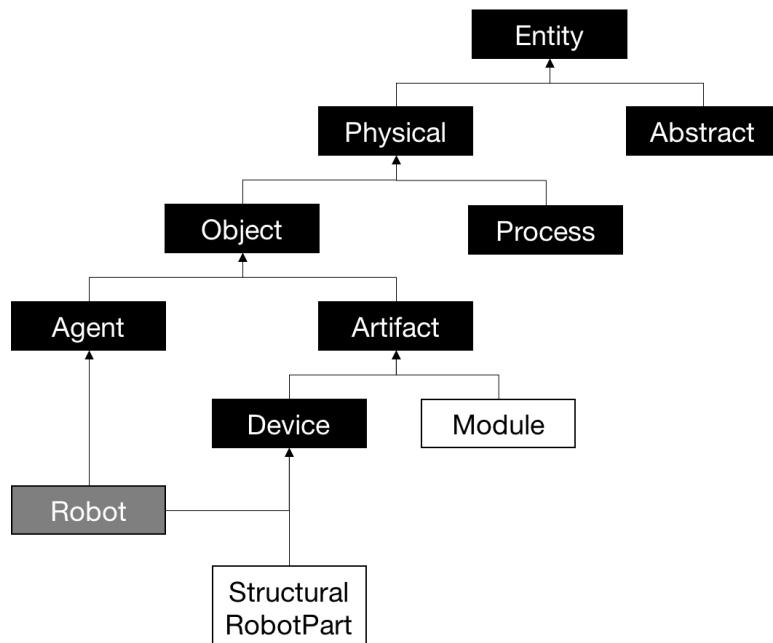


Figura 2: Taxonomía de los principales conceptos en ADROn (en blanco) y su relación con otras ontologías (SUMO, bloques negros, y CORA, cajas grises).

ADROn también incluye otros axiomas que aún no están cubiertos por el estándar, pero que son de utilidad para nuestro propósito. Algunos de ellos entran en las siguientes categorías:

- *Robot types* según el entorno en el que el robot va a desarrollar su actividad (p.ej. *stationary robot*, *ground robot*, *aerial robot*, *underwater robot*, etc.).
- *Robot processes* (p.ej. *robot communicating*) y *subprocesses* (p.ej. *robot grasping*, *robot walking*, etc.) los cuales complementan los procesos ya existentes en CORA.

Esta ontología está en continuo desarrollo, por lo que los conceptos y las relaciones siguen siendo extendidas para añadir nuevas casuísticas al diseño automático.

4.2. Generación de estructuras abstractas mediante ontologías

De acuerdo a ADROn, un robot consiste de una o más *Structural Robot Parts* y cada una de ellas tiene un *Module* o un conjunto de ellos. La Figura 3 muestra una instancia donde podemos ver un *Robot* (humanoide), una de sus *Structural Robot Parts* (una pierna); y algunos de los *Modules* (activos y pasivos) que constituyen la pierna.

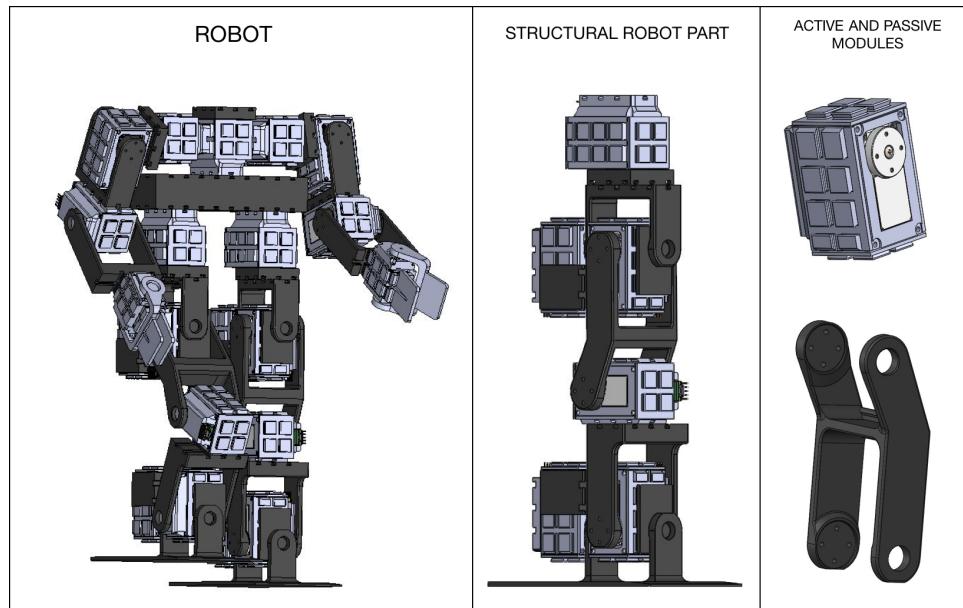


Figura 3: Instancias de los principales axiomas de ADROn relacionados con el concepto fundamental de CORA (*Robot*).

Nuestro sistema utiliza la información extraída de la descripción de las funcionalidades del robot (atributos, capacidades, etc.) que los usuarios escriben en lenguaje natural para describir las tareas que los robots deben realizar. Las características más importantes son los **behaviors** (comportamientos), los cuales se relacionan con las capacidades del robot y se definen como “el proceso o conjunto de procesos que son realizables por el robot”.

ADROn incluye la definición de cada módulo de la arquitectura modular descrita en el entregable E4.2 del proyecto, así como cada comportamiento que un robot puede exhibir y las relaciones entre comportamientos y las *Structural Robot Parts* requeridas para su realización.

La generación conceptual de un robot es un proceso de tres (opcionalmente cuatro) pasos:

1. Primero, dado un conjunto de comportamientos por el usuario, ADROn determina, mediante consultas, las *Structural Robot Parts* necesarias para ejecutar esos comportamientos.
2. Tras ello, nuestro sistema busca en una base de datos de configuraciones qué robots tienen dichas partes.
3. (Opcional) Si existen varias configuraciones con dicho comportamiento, se realizan una serie de cuestiones al usuario para desambiguar el resultado y seleccionar una configuración.
4. Por último, el sistema crea la estructura abstracta (conceptual y adimensional) de un robot capaz de tener o realizar esos comportamientos y proporciona ese esquema al generador de la estructura física, según se explica en la Sección 5.

A continuación mostramos un ejemplo de generación de un robot que tiene la capacidad de caminar (*walk*).

- En primer lugar, el usuario solicita un robot que tenga el comportamiento de caminar.
- Un proceso de consulta en ADROn determina la necesidad de usar piernas.
- Una búsqueda en la base de datos de configuraciones obtiene todas las coincidencias (p.ej. humanoide, cuadrúpedo y hexápodo).
- El sistema pregunta al usuario si el robot caminante necesita manipular objetos. Suponiendo que el usuario responde afirmativamente, se desambigua el resultado, puesto que sólo el humanoide tiene definidos brazos para manipular (en caso contrario, deberíamos seguir realizando cuestiones hasta tener una sola configuración).

- El sistema responde generando la estructura abstracta de un humanoide: dos piernas, tronco, dos brazos y una cabeza. En este momento acaba el diseño conceptual y comienza la parametrización para obtener una estructura física.

5. Generación de la estructura física

Una vez se ha obtenido la estructura abstracta, es necesario concretarla en una estructura física. Para ello es necesario indicar una serie de parámetros o de requisitos que serán incluidos en la descripción de los módulos físicos para dar forma al robot conceptual. Las siguientes subsecciones ilustran el tratamiento que debemos realizar con ambos tipos de entradas: parámetros y requisitos.

5.1. Parametrización de módulos

En el entregable E4.2 se definió una arquitectura robótica modular que permite implementar cualquier configuración diseñada por nuestro sistema. Esta plataforma está basada en módulos activos y módulos pasivos parametrizables cuyas partes mecánicas pueden ser fabricadas mediante impresión 3D siguiendo la filosofía DIY. Las caras de los módulos activos y pasivos se unen entre sí mediante un sencillo sistema de pasadores tipo cola de milano.

La figura 4 muestra un ejemplo en donde se aprecian módulos activos, módulos pasivos y las uniones entre ambos.

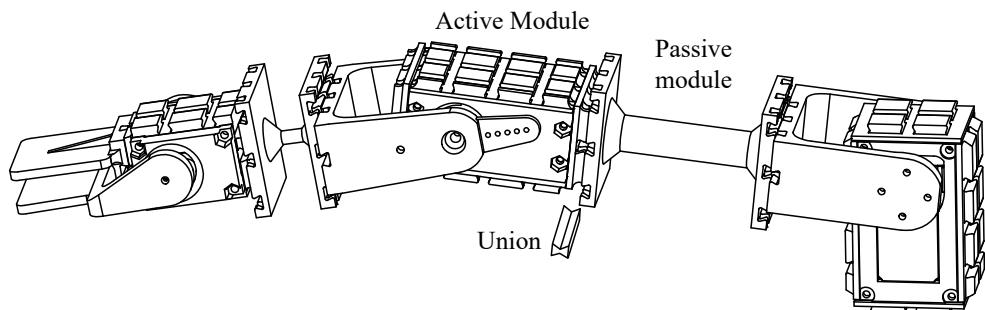


Figura 4: Modular Robot Architecture

Los módulos activos tienen una geometría predefinida y determinada por los componentes eléctricos/electrónicos que contienen, por lo que no son parametrizados en nuestra estructura, aunque sí que pueden variar en función de los requisitos solicitados.

5.1.1. Optimización de módulos pasivos

Los módulos pasivos son partes mecánicas que no contienen elementos electrónicos. Pueden usarse para interconectar módulos activos o incluirse para proporcionar nuevas funcionalidades a una estructura. El ejemplo más conocido es el de los eslabones de un brazo robótico, cuya longitud se puede ajustar para determinar el máximo alcance del brazo. Estos eslabones pueden adoptar geometrías variadas tales como ángulos rectos, siendo las barras rectas las más habituales.

Otro ejemplo de módulos pasivos son los pies, utilizados para incrementar la estabilidad de un robot caminante al incrementar la superficie de contacto con el suelo. Algunos de estos módulos pueden modificar propiedades de los robots al cambiar sus dimensiones. Por ejemplo, un robot con ruedas puede adaptar su máxima velocidad dependiendo del diámetro de sus ruedas, siempre que no se supere la potencia demandada del servo que la mueve.

En sistemas más complejos, como robots humanoides, el número de parámetros es mayor y están relacionados con la escalabilidad de un robot de dimensiones nominales. Algunos parámetros de estos sistemas son la longitud de los eslabones (p.ej. fémur y tibia) y el ancho de las caderas (según se menciona en la Sección 6.5), que cambia, por ejemplo, la habilidad para negociar obstáculos de diferentes alturas mientras se camina.

Cada configuración abstracta posee ciertos parámetros que pueden modificar sus capacidades y que pueden ser parametrizados por el usuario.

Caso de uso: eslabón de un manipulador industrial En este ejemplo estudiaremos la parametrización de los eslabones de un manipulador. Los eslabones pasivos extienden el espacio de trabajo de un brazo robótico dependiendo de su longitud, mientras que la inercia de su sección determina varias propiedades relacionadas con resistencia de materiales. En nuestro caso, la parametrización debe cumplir dos objetivos:

- Alcanzar el rango requerido por los requisitos indicados por el usuario.
- Evitar la deflexión (y/o torsión) de los eslabones.

El planteamiento del problema estático que sirve de base a la parametrización se muestra en la Figura 5, en el que se toma el eslabón como una viga empotrada en voladizo

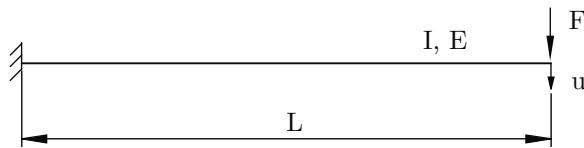


Figura 5: Diagrama de fuerzas de una viga sometida a flexión

Adicionalmente, puesto que estamos fabricando nuestros módulos con una impresora 3D, resulta razonable utilizar la mínima cantidad de materia prima posible que garantice ambos objetivos, de modo que se abarate el coste de fabricación. Para ello se ha planteado un problema de minimización de volumen de eslabones bajo ciertas restricciones en sus dimensiones, tales como la longitud de la viga, L , otras restricciones geométricas que dependen de la forma del eslabón, y una rigidez mínima exigible para evitar deflexión, la cual se puede relacionar con la máxima carga que debe transportar el manipulador.

$$\begin{aligned} & \underset{L}{\text{minimize}} && \text{Material} (\iff \text{Volumen}) \\ & \text{sujeto a} && \begin{cases} K \geq K_0, \\ \text{restricciones geométricas,} \end{cases} \end{aligned}$$

donde K es la rigidez de la barra, K_0 es la mínima rigidez requerida para evitar desplazamientos y las restricciones geométricas dependen de la sección concreta elegida para la barra y del entorno de impresión (p.ej. cada sistema de prototipado tendrá un grosor mínimo de impresión).

Este problema se ha resuelto para las diferentes secciones de la Figura 6 aplicando el método de Karush-Kuhn-Tucker para problemas de optimización no lineal [34].

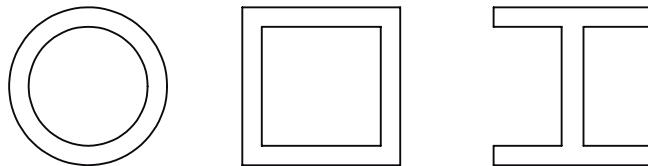


Figura 6: Secciones analizadas en la minimización de volumen

Según lo esperado, los resultados muestran que las secciones cuadrada y en 'I' reducen el material para la rigidez exigible en el plano vertical. Sin embargo han

sido descartadas para su uso en brazos robóticos debido a que sus eslabones rotan por efecto del giro de los motores, cambiando la dirección de aplicación de las fuerzas externas (p.ej. gravedad), lo que hace que dichas secciones se comporten peor en general (ya que sólo ocasionalmente se aplicará la fuerza sobre el plano para el que se ha optimizado su rigidez). Por ello se ha decidido utilizar eslabones huecos de tipo cilíndrico.

$$\begin{aligned} & \text{minimizar}_{L,r_2,r_1} && f = L(r_2^2 - r_1^2) \\ & \text{sujeto a} && r_2 \leq r_{2max} \quad r_2 - r_1 \geq e \\ & && L_{min} \leq L \leq L_{max} \quad (r_2^4 - r_1^4)L^{-3} \geq \tilde{K}_0, \end{aligned} \quad (1)$$

donde L es la longitud del eslabón, r_1 y r_2 son los radios interior y exterior, e es el menor espesor permitido por la impresora 3D, y \tilde{K}_0 es una constante relacionada con la rigidez y dependiente del módulo de Young, la inercia de la sección y la longitud del eslabón. La solución a este problema, que proporciona las dimensiones de los eslabones pasivos óptimos del manipulador, viene dada por

$$\begin{aligned} r_2 &= r_{2max}, \quad r_1 = \sqrt[4]{r_{2max}^4 - \tilde{K}_0}, \quad L = L_{min} \\ f &= \pi L \left(r_{2max}^2 - \sqrt{r_{2max}^4 - \tilde{K}_0} \right). \end{aligned} \quad (2)$$

Según se esquematiza en la Figura 7, esta solución será válida siempre que $r_2 - r_1 \geq e$. En caso contrario se establece que $r_2 - r_1 = e$ y los radios se obtienen de resolver el siguiente sistema de ecuaciones algebraicas:

$$\begin{aligned} r_2 - r_1 &= e \\ r_2^4 - r_1^4 &= \tilde{K}_0. \end{aligned} \quad (3)$$

5.2. Análisis dinámico de requisitos

Algunos requisitos de usuario pueden condicionar severamente la estructura mecánica/eléctrica del robot debido a las restricciones que añaden. En este trabajo se detallará el efecto de un requisito de carga transportada y de un tamaño del entorno de trabajo en el diseño físico de un manipulador industrial de 3 grados de libertad (gdl).

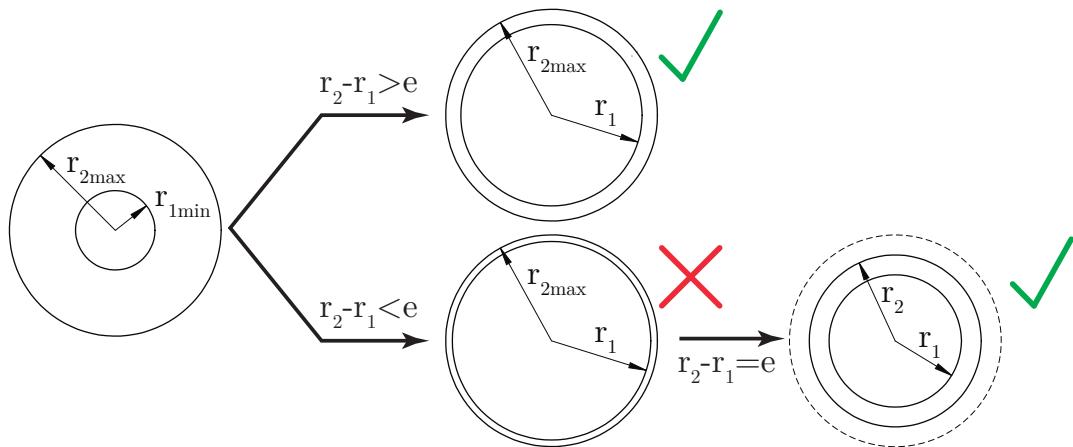


Figura 7: Solución óptima condicionada por la restricción de espesores de impresión

Para ello se ha realizado un estudio de la dinámica del manipulador así como de la peor situación para los actuadores con el fin de garantizar la funcionalidad del manipulador en todo el espacio de trabajo. Este estudio permite determinar tanto las dimensiones de los eslabones como el torque que deben desempeñar los servos en cada una de las distintas articulaciones.

Una vez definidas las solicitudes exigibles a cada uno de los eslabones y las longitudes que deben tener se utiliza el proceso definido en la Sección anterior para generar los eslabones optimizados en volumen para nuestra estructura física. Esto no sólo tiene beneficios económicos, sino también de ligereza y, por tanto, de ahorro energético en la operación del manipulador.

5.2.1. Objetivos

Los objetivos de este análisis son:

1. Encontrar expresiones adecuadas, para lograr relaciones entre la longitud del eslabón, cargas soportadas (incluyendo las fuerzas de inercia) y flecha máxima (deflexión). En estas expresiones se pueden determinar las variables que el usuario debe proporcionar (requisitos).
2. Garantizar que los servos de las articulaciones sean capaces de producir el torque necesario para los movimientos solicitados.

3. Asegurar la integridad de la estructura analizando los concentradores de tensiones de la misma.

5.2.2. Planteamiento

El problema que se va a resolver parte del mismo planteamiento de la optimización de la Sección 5.1.1, definido en la Figura 5, salvo porque en este caso añadimos las fuerzas de inercia a las fuerzas estáticas debidas a la carga en el extremo y al propio peso del eslabón.

Algunas hipótesis asumidas en este problema son:

- La sección más solicitada está en el empotramiento, ya que es donde se tiene el mayor momento flector. Esa sección ha sido la estudiada, ya que es el lugar donde produciría rotura por tensión.
- En el empotramiento, la viga tiene las condiciones de contorno de giro y desplazamiento iguales a 0.
- El desplazamiento vertical o flecha máxima es proporcional a las reacciones de la fuerza aplicada (en la imagen, llamada P).

Para obtener la relación entre la tensión en el empotramiento y las cargas dinámicas del eslabón se ha resuelto el problema de cálculo matricial establecido por la siguiente expresión:

$$F = K * u \quad (4)$$

donde F representa el vector de solicitudes (fuerzas y momentos), u es el vector de desplazamientos, y K es la matriz de rigidez que relaciona ambos términos.

Se ha discretizado la viga en un único elemento viga unidimensional con 2 nodos (uno en cada extremo), teniendo en total 6 gdl, 3 en cada nodo, como se puede ver en la Figura 8.



Figura 8: Elemento viga unidimensional con 3 gdl en cada nodo

Aplicando la ecuación (4) al problema descrito en la Figura 5, y resolviendo la ecuación matricial resultante podemos obtener las reacciones en el empotramiento

(R_x, R_y, M_z) , el desplamiento vertical y el giro en el extremo, (u, θ) , que provoca la carga P (se ha tomado como criterio de signos positivo hacia arriba) en función de: la longitud del eslabón L , el módulo de Young del material E , y el momento de inercia de la sección I . Estas reacciones nos servirán para calcular los torques máximos que debe realizar cada articulación, lo que limitará la elección de servos que debe incorporar el robot. El proceso de cálculo se indica en el Anexo 9.

Por otro lado, el cálculo de los valores máximos que tendremos en el empotramiento permitirá realizar un análisis de fallo (rotura) de los eslabones de un robot. Esto es necesario porque a los elementos optimizados en la Sección 5.1.1 se le añaden unas bases en los extremos para permitir la unión con otros elementos. Las aristas generadas en dichas uniones suelen ser concentradores de tensiones que pueden causar fallo en la estructura. Este análisis se detalla en la Sección 10.

5.3. Caso de estudio: Generación automática de la estructura física de un manipulador

El proceso que determina los parámetros óptimos para los módulos activos y pasivos depende de la estructura abstracta seleccionada. El caso de un manipulador se muestra en la Figura 9.

Una vez se ha generado la estructura abstracta del manipulador haciendo uso de la ontología, se realiza el siguiente proceso:

1. Primero se asigna un parámetro longitud para los módulos pasivos calculado en función del espacio de trabajo deseado.
2. Posteriormente, con un cálculo iterativo, basado en la evaluación de matrices jacobianas paramétricas, se determinan los pares que debe ejercer cada articulación y las fuerzas y pares que debe soportar cada eslabón. Esta evaluación dependerá de los requisitos indicados por el usuario (p.ej. carga máxima), así como de los módulos de la arquitectura (p.ej. pesos, distribución de masas, dimensiones).
3. Se eligen los módulos activos que debe llevar cada articulación para poder cumplir con los requisitos de par obtenidos.

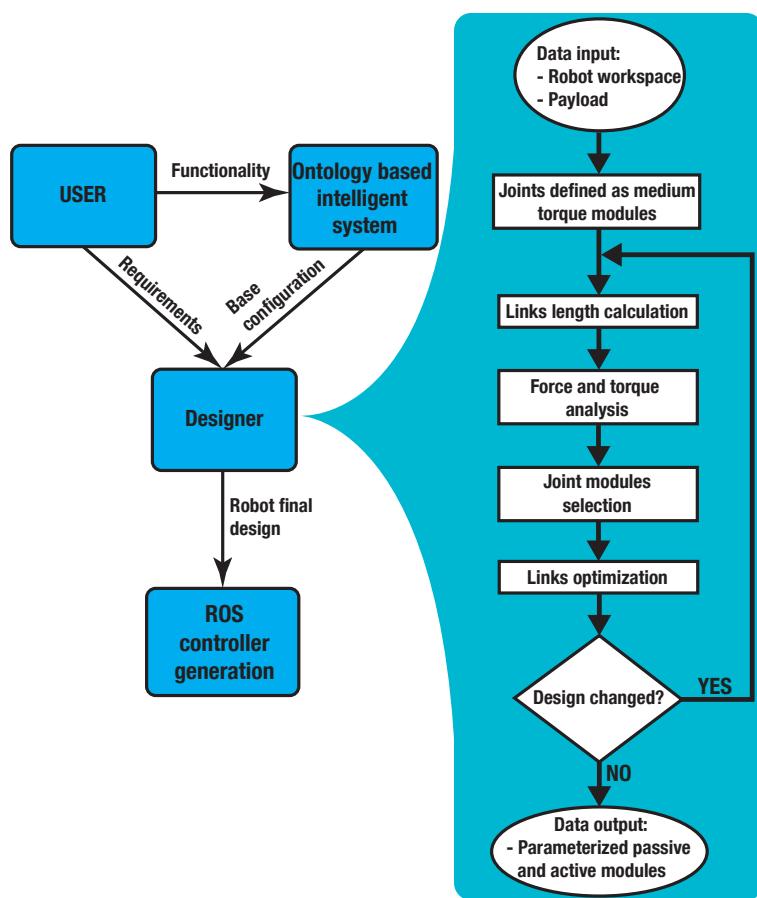


Figura 9: Proceso para la generación automática de un manipulador

4. Se diseñan los módulos pasivos haciendo uso de los resultados de optimización y análisis dinámico y de tensión descritos en las Secciones 8, 9 y 10.
5. El proceso realiza una nueva iteración si en la iteración anterior se ha generado/seleccionado un módulo pasivo o activo diferente, ya que las características del robot, (peso, inercia, etc.), habrán cambiado. La Figura 10 muestra un ejemplo de diseños distintos obtenidos ante distintos requisitos del usuario. Como puede observarse, aunque los robots A y B tengan la misma estructura abstracta (manipulador 3gdl), el diseñador automático determinó que la geometría de los módulos pasivos y el tipo de módulos activos sea diferente.



Figura 10: Robot A. (izq.) Dos módulos activos de torque alto y uno de torque medio. Robot B.(dch.) Un módulo activo de torque alto y dos de torque medio.

6. Por último se realiza la descripción formal del robot en un modelo URDF [8] de ROS. Para ello, partimos de las estructura abstracta programadas en Xacro [12] donde la parametrización viene implícita en la declaración de variables y el uso de bloques condicionales. El modelo URDF obtenido es utilizado por un sistema automático para la generación del controlador del robot en ROS tal como se expone en la siguiente sección.

Los procesos de diseño para otras configuraciones base no son mostrados aquí por cuestiones de espacio, pero, a excepción de pequeñas variaciones (p.ej. peso que debe ser capaz de soportar la pata de un robot con patas en lugar de peso que debe transportar un manipulador) siguen un esquema similar con cálculos iterativos de módulos pasivos y activos. Las Figuras 11 y 12, muestran, respectivamente, el diseño de dos robots humanoides y dos robots hexápodos generados mediante distintos requisitos de usuario.

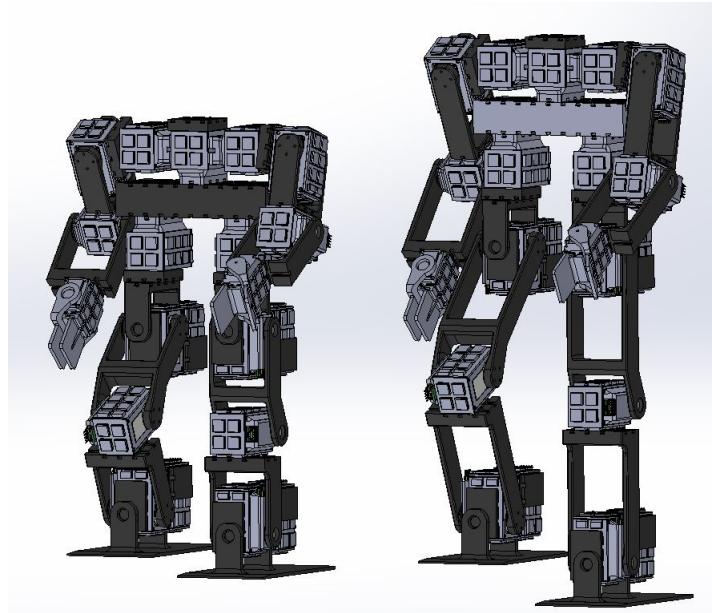


Figura 11: Dos parametrizaciones distintas de robots humanoides para distintos requisitos de usuario.



Figura 12: Dos parametrizaciones distintas de robots hexápodos para distintos requisitos de usuario.

6. Generación automática de controladores

El último paso en la generación automática de robots funcionales es la creación de un controlador para realizar los comportamientos requeridos por el usuario. Para cada configuración básica de la base de datos, un controlador parametrizable ha sido también diseñado dentro del entorno de trabajo de ROS.

Para el control de los robots generados se pretende usar ROS ya que es uno de los entornos de trabajo (framework) más usados para robótica hoy en día y simplificará la comunicación entre el robot y el PC, además de ser código abierto, lo que encaja perfectamente con la filosofía DIY. Para realizar el control del robot a través de ROS es necesario realizar la descripción del robot en el formato URDF, que es un lenguaje de marcas tipo XML (*eXtensible Markup Language*) para representar modelos de robots. Permite describir las partes mecánicas de un robot y sus características tanto cinemáticas como dinámicas. Para simplificar esta tarea nos apoyaremos en el lenguaje Xacro, según se explica en la Sección 6.1.

Una vez se dispone del fichero de descripción del robot, ROS cuenta con diferentes herramientas para la generación de controladores de movimiento. De ellas se han utilizado dos dentro de este proyecto:

- **MoveIt!** [5] es un paquete de ROS que permite realizar el control y la generación de trayectorias de robot manipuladores de manera asistida e intuitiva. Esta herramienta se usará para controlar el robot manipulador parametrizable.
- **KDL** (*Kinematics and Dynamics Library*) [11] es una librería de ROS escrita en lenguaje C++ que permite describir cadenas cinemáticas para realizar transformaciones cinemáticas directas e inversas, lo que es la base del controlador del robot. Utilizaremos esta librería para el control del robot bípedo.

El resto de la sección describe la generación de estos controladores para varias de las configuraciones básicas.

6.1. Generación automática de modelos de robots a partir de archivos Xacro parametrizables

Los archivos URDF, aunque necesarios para la descripción de los robots, son habitualmente bastante extensos y difíciles de editar manualmente. Para conseguir un esquema parametrizable se utilizan macros que generan estos URDF y que están escritas en lenguaje Xacro [12]. Este lenguaje admite declaración de variables, a través de las que se pueden modificar propiedades en la descripción del robot (fichero URDF) a partir de los parámetros que debe tener la estructura del robot, y que han sido, bien introducidos directamente por el usuario según se indica en la Sección 3.4, bien obtenidos durante la generación de la estructura física (Sección 5). Además, se permite el uso de bloques condicionales, lo que facilita el tratamiento de diferentes casuísticas en la generación. Otras ventajas de utilizar archivos Xacro son:

- Reducen el tamaño total del archivo URDF.
- Facilitan la lectura y mantenimiento de los archivos URDF.
- Crean módulos reutilizables para generar partes estructurales repetidas tales como piernas robóticas.

La generación de archivos URDF a partir de archivos Xacro es directa usando el paquete de ROS del mismo nombre.

6.2. Controlador para robots con ruedas

Nuestra base de datos contiene dos configuraciones de robots con ruedas [37]:

- **Diferencial:** Dos ruedas motrices independientes y paralelas y un apoyo omnidireccional (idealmente una esfera libre). Esta configuración no tiene deslizamiento en las ruedas.
- **Skid-steer:** Cuatro ruedas motrices independientes. La tracción es delantera y las dos ruedas traseras presentan deslizamiento en los giros.

Ambos robots son parametrizables de manera que se puede variar la separación entre las ruedas o el radio de éstas. Nuestro sistema de diseño automático

permite modelar y controlar robots de ambas configuraciones. Para conseguir esta parametrización de nuevo se ha usado un programa Xacro a partir del cual se generara el URDF, puesto que varias partes (ruedas) están repetidas en el mismo. Una vez definido el robot URDF el control se genera automáticamente utilizando el paquete de ROS llamado *diff_drive_controller* [4].

6.3. Controlador para robots serpiente

Los robot serpiente son altamente modulares ya que todo su cuerpo está compuesto por módulos iguales (osciladores) encadenados en dos orientaciones diferentes (vertical y horizontal). Los parámetros de esta configuración son el número de módulos y su tamaño. El lenguaje Xacro es una opción muy eficiente y sencilla de crear este tipo de robots ya que se puede utilizar una única macro que genere un módulo según el tamaño deseado y a continuación repetirlo tantas veces como módulos se requiera que tenga el robot. Un controlador válido para cualquier robot serpiente y que consiste en referencias sinusoidales desfasadas para cada módulo se ha implementado en ROS. Dependiendo de los parámetros de estas referencias, se pueden generar una gran variedad de movimientos [22].

6.4. Controlador para robots manipuladores

Existen dos tipos de configuraciones de robots manipuladores, según sean de 3 o de 6 grados de libertad. Puesto que se utilizan configuraciones de robots antropomórficos, el robot de 6gdl consiste en el de 3gdl, que permite alcanzar cualquier posición dentro del espacio de trabajo, al que se le añade una muñeca con otros 3gdl, que permite controlar la orientación con la que alcanzamos una posición. El modelo URDF del robot se genera de nuevo utilizando Xacro, ya que un manipulador consiste en la concatenación de varios eslabones, cada uno de los cuales puede ser definido mediante cuatro parámetros cinemáticos (parámetros de Denavit-Hartenberg), por lo que de nuevo es necesaria únicamente una macro para generarlo. Dicha macro también incorpora los parámetros dinámicos de los eslabones.

El controlador en ROS de esta configuración se ha realizado mediante la herramienta *MoveIt!*, que permite generar las trayectorias de los movimientos a partir del modelo URDF. Para usar *MoveIt!* es necesario realizar la configuración del

robot a través del asistente que proporciona la herramienta. Una vez configurado el robot, se genera un paquete que permite visualizar el robot modelado. Con el fin de que el paquete generado con *MoveIt!* acepte los robots parametrizables se han modificado los archivos para que en lugar de cargar el archivo URDF, use directamente el archivo xacro que genera el archivo URDF en tiempo de ejecución.

El control de bajo nivel (señales de control de actuadores) del robot se realiza mediante el paquete *ActionLib* [1] de ROS. Éste abre un canal de comunicaciones entre el PC (cliente) y el *hardware* embebido (servidor) en el robot, que consiste en un módulo Intel Edison capaz de ejecutar ROS. Este canal de comunicación nos permite tener realimentación continua de la trayectoria realizada por el robot.

La Figura 13 muestra la visualización en RViz [9] de los modelos URDF generados anteriormente y mostrados en la Figura 10. Las maniobras que se muestran corresponden a trayectorias implementadas mediante el controlador generado automáticamente para cada instancia del robot.

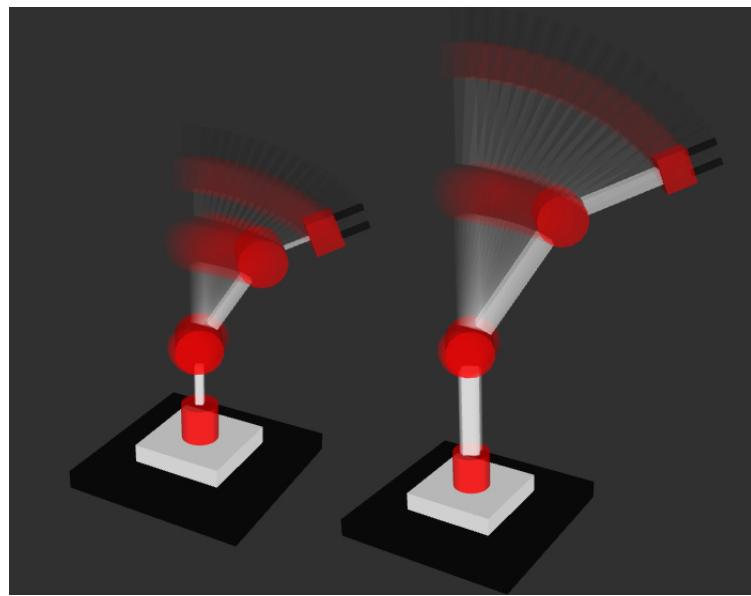


Figura 13: Visualización en RViz de maniobras definidas en **MoveIt!** para brazos robóticos con distinta parametrización.

6.5. Controlador para robots humanoides

En estos robots los módulos pasivos (p.ej. fémur y tibia) son parametrizables en longitud según se muestra en la Figura 14. Puesto que las cadenas cinemáticas de estos robots son más complejas que en un manipulador, el paquete **MoveIt!** no puede generar automáticamente el controlador, por lo que se utiliza la librería KDL para generar las referencias articulares necesarias para el movimiento a lo largo de una trayectoria. KDL contiene solvers genéricos para cadenas cinemáticas que permiten resolver tanto la cinemática directa como la cinemática inversa.

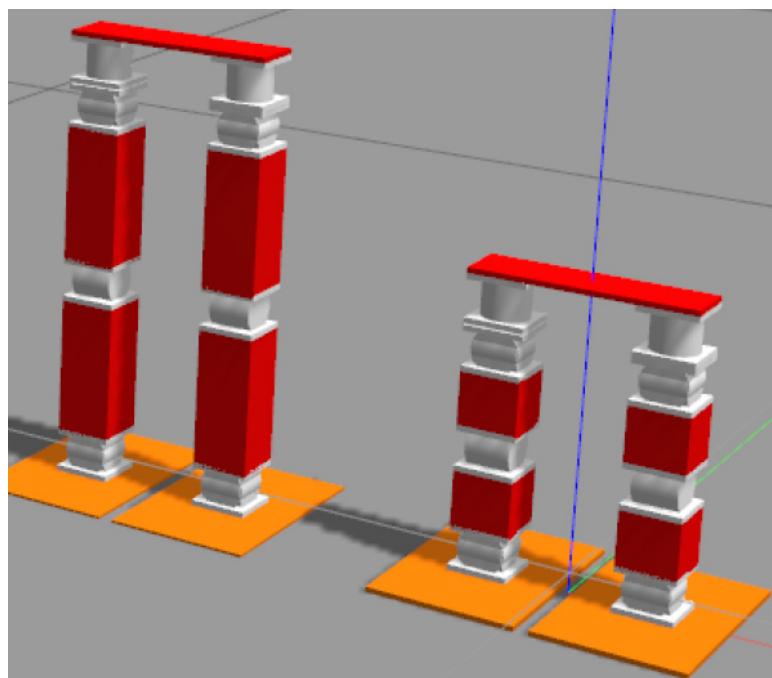


Figura 14: Piernas de dos robots humanoides con diferente parametrización.

El proceso para la generación del control automático es el siguiente:

1. Se ha utilizado una caminada estable ya definida en ROS para un robot concreto, (el robot Darwin [24] tiene un modelo implementado en ROS), para obtener un conjunto de referencias articulares que sirvan de base para la caminada de nuestro humanoide. La Figura 15 muestra estas referencias para la pierna derecha.
2. A continuación se han calculado los principales componentes frecuenciales de

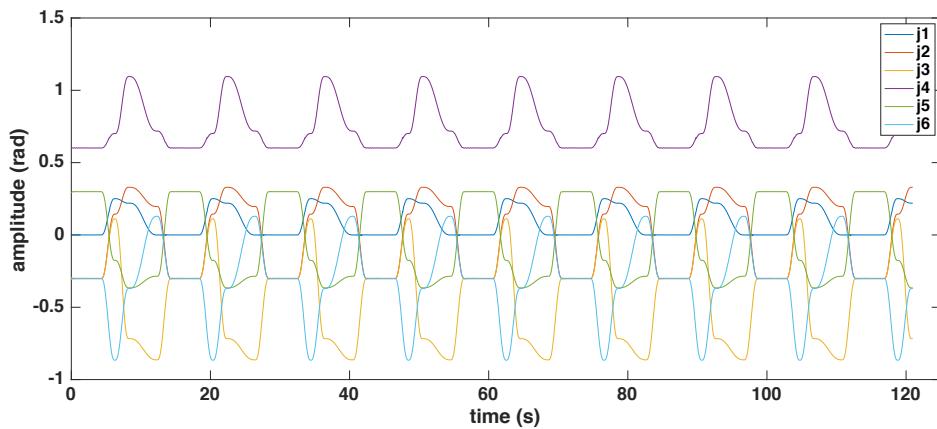


Figura 15: Referencias articulares de la pierna derecha durante varios pasos de una caminada estable.

esta trayectoria para cada articulación mediante el uso de una FFT². Estos valores han servido para crear un nuevo conjunto de referencias articulares de forma sintética, al que llamaremos caminada sintética, que puedan ser parametrizadas.

3. Para garantizar la estabilidad de la caminada ante cambios en los parámetros que definen las piernas del humanoide, se han calculado empíricamente unos mapas de conversión que determinan la relación entre el tamaño de los módulos pasivos de la pierna y los parámetros de la locomoción sintética. Por ejemplo, se ha determinado que, si las articulaciones dos y cinco están escaladas por un factor, la inclinación del cuerpo puede cambiarse para mantener el equilibrio, mientras que el tamaño del paso puede modificarse escalando las referencias de las articulaciones tres, cuatro y seis. La Figura 16 muestra estos mapas, los cuales resultan ser prácticamente lineales.
4. Por último, las referencias articulares para una instancia del robot se calculan escalando la caminada sintética del paso 2 por los factores obtenidos en los mapas de conversión del paso 3. Como ejemplo, la Figura 17 muestra las

²En este punto se han eliminado las zonas de descanso en las que la articulación se encuentra en reposo y la componente continua del movimiento, puesto que causaban distorsiones en los resultados. Estas características se vuelven a incluir posteriormente en la generación de la caminada sintética

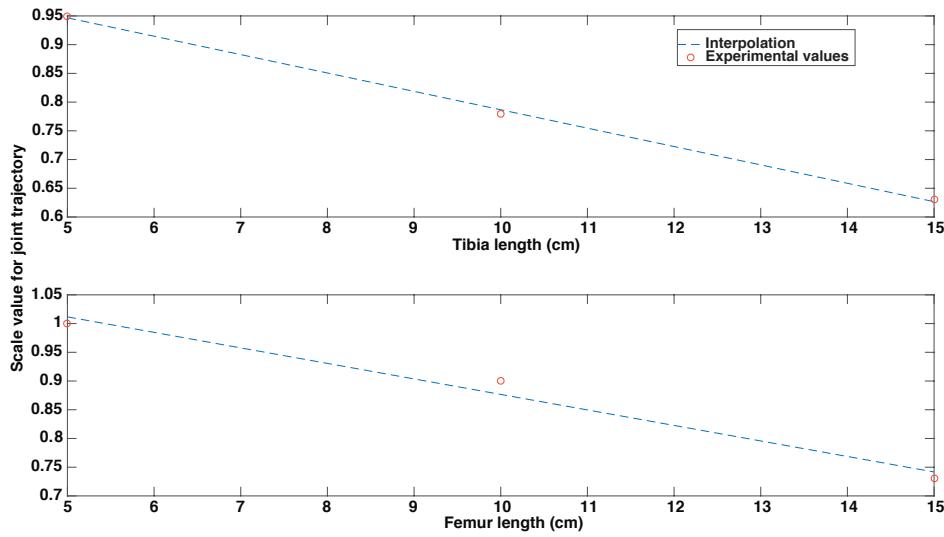


Figura 16: Mapas de conversión de caminada sintética de humanoides.

referencias de la articulación dos para los robots de la Figura 14, las cuales se obtienen a partir de la reconstrucción de la FFT según la siguiente ecuación

$$A_j = (-0,0320 \cdot x + 1,1067) \sum_{i=0}^N a_i \sin(\omega_i t + \phi_i) \quad (5)$$

donde N es el número de armónicos de la caminada sintética; a_i , ω_i y ϕ_i son los valores de la FFT obtenidos en el paso 2; y x es la longitud del módulo pasivo (tibia) en centímetros.

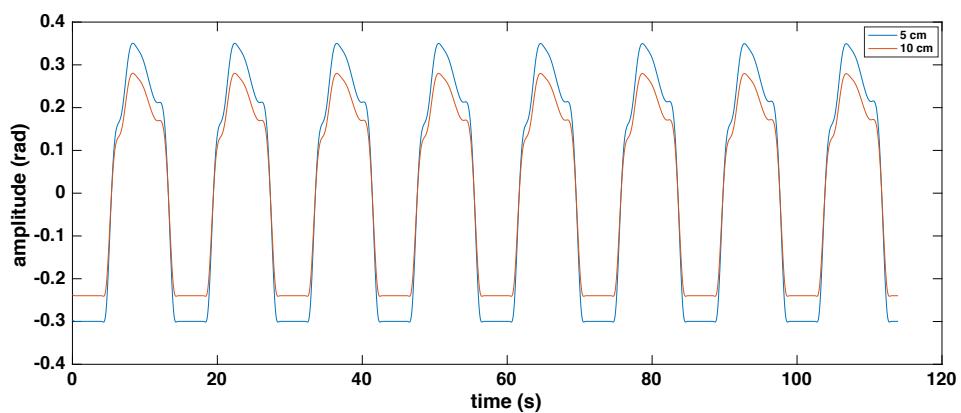


Figura 17: Referencias para la articulación 2 de los robots humanoides de la Figura 14.

7. Anexo A. Ontologías y representación del conocimiento

Este anexo muestra algunos de los conceptos relevantes asociados a la representación del conocimiento, haciendo hincapié en la representación mediante ontologías.

7.1. Conceptos de relevancia

Lenguaje Lógico Se trata de un término asociado a la lógica simbólica. La lógica simbólica no es más que un eslabón más en la evolución de la lógica tradicional en el que un conjunto de símbolos y reglas, sobre los que se fundamenta la construcción de predicados, permite la inferencia de información. Se debe tener en cuenta que esos predicados se hacen respecto a unas *Entidades* y dependiendo del orden del lenguaje, utilizando la definición matemática de "orden", nos permitirán una inferencia menor o mayor.

Lenguaje de primer orden Es un lenguaje lógico en el que los predicados permiten describir hechos respectivos a unas Entidades. Sin embargo, no podemos describir propiedades de esas Entidades o establecer relaciones entre ellas (esto sería posible utilizando un lenguaje de segundo orden). En la lógica de primer orden los predicados son entendidos como funciones matemáticas (una o varias entradas, una salida (consecuencia)).

7.2. Procesamiento de conocimiento

El procesamiento de conocimiento es el campo de la Inteligencia Artificial dedicado a representar información proveniente del mundo en un formato en el que pueda ser utilizado por un ordenador para resolver tareas complejas como interacción robot-humano. Incluye dos vertientes: Representación de conocimiento (redes semánticas, estructura, ontologías, etc.) e Inferencia o Razonamiento del conocimiento (motores de inferencia, teorema *prover* y clasificadores). Existe un gran número de métodos para representar y extraer conocimiento, aunque en este documento se van a desarrollar sólo algunas de esas opciones.

7.2.1. Ejemplo: KnowRob

KnowRob [40] es un sistema de procesamiento de conocimiento que combina representación de conocimiento (escrita en Prolog³) y métodos de razonamiento o inferencia que utilizan técnicas para adquirir y aumentar el conocimiento. Además, puede servir como *framework* para integrar información proveniente de distintas fuentes (robots, humanos, ordenadores, etc.). KnowRob combina una enciclopedia estática de conocimiento, conocimiento de sentido común, descripciones de tareas, modelos del entorno, información sobre objetos y sobre acciones observadas que han sido adquiridas desde varias fuentes (automatizadas manualmente, derivadas de la observación o importadas de la web). Además soporta diferentes mecanismos de razonamiento, probabilísticos y deterministas, clasificación y segmentación de métodos, interfaces para realizar búsquedas y herramientas de visualización, entre otros).

7.3. Representación de Conocimiento: Ontologías

Existen bastantes medios para representar conocimiento, en este documento nos centraremos en el uso de Ontologías. En un sentido computacional, las ontologías son un medio para expresar, de una manera formal, conocimiento sobre un determinado dominio de interés haciéndolo computable. Las Ontologías están compuestas por una taxonomía (conjunto de axiomas/conceptos correctamente estructurado) y un conjunto de relaciones y limitaciones entre todos esos axiomas.

7.3.1. Lenguajes

Hay un buen número de lenguajes comúnmente utilizados para representar conocimiento a través del uso de ontologías, entre ellos podemos citar: [27]: CycL, F-Logic, LOOM, KIF [19], Ontolingua, RDF(S) and OWL [28]. Sin embargo, en este documento se centrará la atención en dos de ellos, que serán utilizados en el desarrollo del trabajo: *Web Ontology Language (OWL)* and the *Knowledge Interchange Format (KIF)*.

³Lenguaje de programación lógica de propósito general utilizado en inteligencia artificial y lingüística computacional.

OWL Utilizado para definir ontologías en la Web. Una ontología escrita en este lenguaje describe un dominio a partir de clases, propiedades e individuos pudiendo incluir ricas descripciones de las características de los objetos. Está desarrollado a partir de otros lenguajes: *Resource Description Framework* (RDF) [14] y *RDF Schema* (RDFS), extendiéndolos semánticamente. Dentro de OWL encontramos tres sub-lenguajes, dependiendo del nivel de comprensión y expresividad: OWL-Lite, OWL-DL y OWL-Full.

SUO-KIF *Suggested Upper Ontology Knowledge Interchange Format* [33] es un lenguaje diseñado para la escritura e intercambio de conocimiento y fue desarrollado a partir de KIF para dar soporte a la definición de la ontología *Suggested Upper Merged Ontology* (SUMO) [31]. Pensado a priori como un lenguaje de primer orden lo que supone un buen compromiso entre la demanda computacional del razonamiento (inferencia) y la riqueza de la representación.



Figura 18: Comparación entre OWL and SUO-KIF dependiendo del nivel de razonamiento y expresividad.

7.3.2. SUMO

Suggested Upper Merged Ontology (SUMO) es una ontología de alto nivel que proporciona definiciones para términos de propósito general y que actúa como fundamento de otras ontologías más específicas. SUMO está escrita en el lenguaje

SUO-KIF y contiene todo el vocabulario contenido en *WordNet*⁴ [17]. SUMO es gratis y pertenece al IEEE. Las ontologías que la extienden están disponibles bajo Licencia General Pública *General Public License*⁵.

La raíz o principal categoría en SUMO es la *Entidad* la cual se divide en los conceptos *Físico* y *Abstracto*. El concepto de *Físico* representa las entidades que tengan extensión espacio-temporal, mientras que el de *Abstracto* describe entidades que no tienen o han sido elegidas para no tener extensión espacio-temporal.

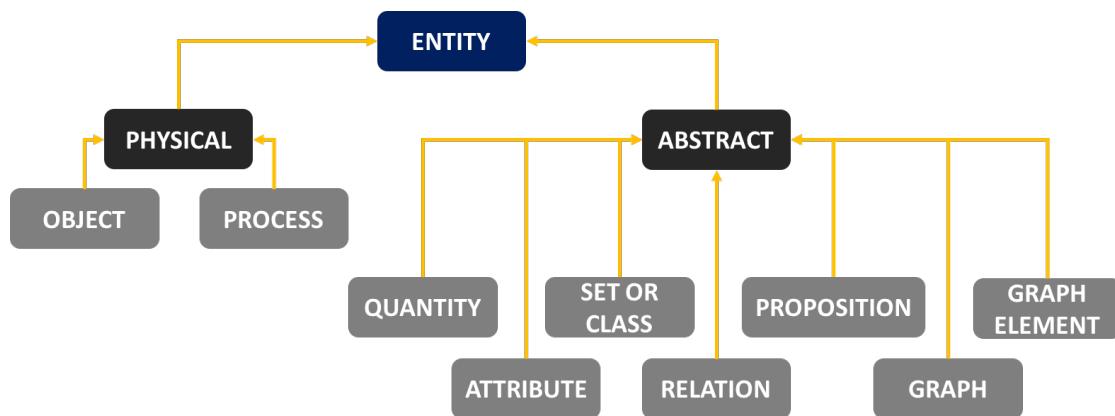


Figura 19: SUMO: Taxonomía básica.

7.3.3. IEEE Standard Ontologies for Robotics and Automation

Este estándar [13] define el núcleo de una ontología que permite la representación de conocimiento en el dominio de la Robótica y la Automática (R&A). Incluye conceptos generales como también sus definiciones, atributos, limitaciones y relaciones.

El estándar está escrito en SUO-KIF y usa algunos axiomas generales definidos en SUMO. Con la finalidad de cubrir tantos conceptos como sea posible, el estándar contiene varias ontologías:

- CORA: Ontología principal del estándar. Sienta la taxonomía básica, incluyendo axiomas relativos a aspectos generales en el dominio R&A.

⁴Extensa base de datos que contiene léxico escrito en Inglés.

⁵www.gnu.org/copyleft/gpl.html

- CORAX: Define conceptos que son demasiado generales como para formar parte de CORA y no están explícita o completamente definidos en SUMO.
- POS: Ontología sobre la posición, orientación y posicionamiento.
- RPARTS: CORA determina que cualquier dispositivo puede ser utilizado como parte de un robot. En este sentido, una parte de robot está considerada como el rol que juega un determinado dispositivo unido a un robot. RPARTS agrega algunos de los más generales y también específicos tipos de partes de robot, así como los requisitos que debe poseer un dispositivo para jugar el rol de parte de robot.

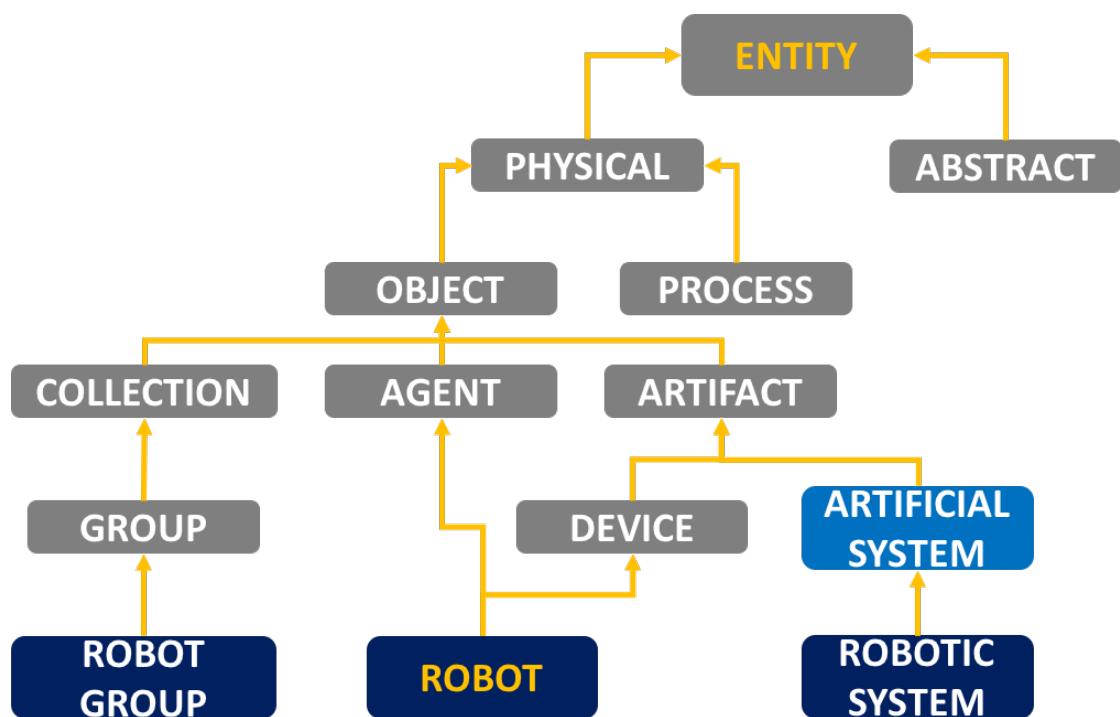


Figura 20: CORA: Taxonomía básica.

7.4. Knowledge reasoning: Métodos de extracción de conocimiento desde Ontologías

7.4.1. Consultas semánticas

Las consultas (*queries*) semánticas permiten realizar consultas y análisis de naturaleza asociativa y contextual. Además, posibilitan la recuperación de información implícita y explícita basada en la sintaxis, semántica e información estructural contenida en los datos. Están diseñadas para entregar resultados precisos o para responder preguntas más confusas y amplias a través de la comparación de patrones y razonamiento digital.

Las consultas semánticas trabajan con grafos, datos vinculados o *triples*⁶. Esto permite procesar las relaciones actuales entre la información e inferir las respuestas extraídas desde la red de datos. Esto está en contraste con la búsqueda semántica, que utiliza la semántica con textos desestructurados (como en el procesamiento de lenguaje natural).

7.4.2. Razonadores semánticos

Los razonadores semánticos son programas informáticos capaces de inferir consecuencias lógicas de un conjunto de hechos irrefutables o axiomas. La noción de razonador semántico generaliza el término "motor de inferencia", proporcionando un conjunto de mecanismos para trabajar más rico. La reglas de inferencia son normalmente especificadas por medio de un lenguaje ontológico y algunas veces de uno lógico-descriptivo. Muchos razonadores utilizan predicados lógicos de primer orden para realizar el razonamiento; la inferencia comúnmente procede concatenando información hacia delante y hacia atrás en el grafo.

7.4.3. Sigma

Sigma [32] es un entorno de desarrollo (IDE), de visualizado y depuración de predicados de primer orden. Trabaja con el lenguaje KIF y está optimizada para utilizar la ontología SUMO.

Sigma incluye un útil conjunto de herramientas para el trabajo relacionado con la ingeniería del conocimiento o tratamiento de datos. Entre ellos: búsqueda por

⁶En la notación de RDF, son recursos en el formato sujeto-predicado-objeto.

término y jerarquía, posibilidad de abrir archivos diferentes, capacidad de inferencia de primer orden, etc. Sigma está escrita en Java y, al igual que SUMO, está disponible bajo Licencia Pública General (GNU).

7.4.4. pySUMO

pySUMO⁷ es un IDE escrito en Python para Ontologías definidas en SUO-KIF. No permite inferir, sin embargo, sí que es posible hacer consultas semánticas directamente desde código escrito en Python y dado que todo el código de control de nuestro proyecto está escrito en este lenguaje, pySUMO es una herramienta interesante. Sin embargo, presenta algunas deficiencias ya que las consultas sólo funcionan con *triples* (sujeto-predicado-objeto), lo que es un problema cuando trabajas con predicados ternarios (con cuatro términos).

Existe una buena documentación de pySUMO, así que no tiene sentido explicarla en mayor profundidad. Un detalle relevante es que pySUMO utiliza Qt para la interfaz gráfica, por lo que en este documento se explicará cómo instalar este IDE.

7.5. Conclusion

En el futuro, sería deseable extraer información de la ontología utilizando motores de inferencia (el de Sigma u otro), pero por el momento (dado que la tarea requeriría mucho tiempo y recursos humanos), vamos a crear un sistema más simple utilizando pySUMO. Diseñaremos una aplicación, escrita en Python, que utilice pySUMO para extraer conocimiento de una ontología (proporcionando así la información suficiente para que nuestro sistema sea capaz de diseñar un robot de manera autónoma).

pySUMO nos permite utilizar la ontología escrita en SUO-KIF, lo que es genial porque necesitamos un lenguaje más expresivo que rápido. Sin embargo, el grupo de IEEE prefiere utilizar la ontología escrita en OWL, obteniendo así una mayor velocidad en la inferencia y la posibilidad de utilizar CORA desde KnowRob. La velocidad de cómputo es algo primordial en el control de robots, dado que ellos utilizan la ontología para el control, es normal que utilicen OWL. Quizás en un futuro sea recomendable hacer lo mismo, y utilizar la ontología no sólo para ge-

⁷pysumo.readthedocs.io/en/latest/

nerar robots, también para controlarlos (dotándolos de mayor inteligencia cuando actúen).

8. Anexo B. Optimización de eslabones

8.1. Formulación y resolución del problema

A continuación, se plantea detalladamente el problema expuesto en la Sección 5.1.1 para el perfil circular hueco, como el método de resolución que se utilizará para dicho problema.

Lo que se pretende es buscar las dimensiones óptimas de una viga con sección circular hueca, de la que se desconocen la longitud, los radios interno y externo y la rigidez, por lo que los parámetros de diseño serán:

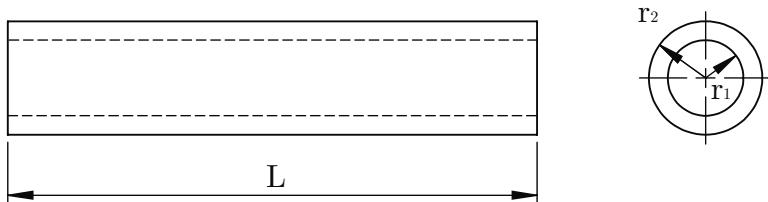


Figura 21: Esquema de una viga con sección circular hueca

$$\mathbf{X} = (r_1, r_2, L)$$

Puesto que lo que se quiere es minimizar es el volumen, la función objetivo será el volumen de una viga de sección circular hueca :

$$V(r_1, r_2, L) = \pi L(r_2^2 - r_1^2)$$

Es obvio que se puede simplificar la expresión a minimizar, puesto que π no influye en el cálculo del mínimo, por lo que no se incluirá en la función objetivo para facilitar los cálculos:

$$f(\mathbf{X}) = L(r_2^2 - r_1^2) \quad (6)$$

Los parámetros de diseño tienen diferentes restricciones. En cuanto al radio externo, no es deseable que la estructura tenga unas dimensiones excesivas, por lo que una restricción importante es el radio externo máximo que se desearía tener:

$$r_2 \leq r_{2max} \quad (7)$$

Otra restricción a considerar es el tamaño mínimo del hueco interior del cilindro. En este caso interesa que dicho hueco tenga un tamaño mínimo:

$$r_1 \geq r_{1min} \quad (8)$$

Se tendrá en cuenta que a la hora de la toma de datos, obviamente, el radio externo máximo debe ser siempre superior al radio interno mínimo.

$$r_{2max} > r_{1min}$$

Es lógico pensar que la longitud del módulo pasivo va a estar entre un valor mínimo y un valor máximo, por lo que se tendría las siguientes restricciones:

$$L_{min} \leq L \leq L_{max} \quad (9)$$

Por último, en cuanto a la rigidez, como se ha explicado anteriormente, se busca que si la estructura está sometida a flexión, la rigidez sea mayor o igual a una dada. En este caso, la rigidez a flexión de una viga de sección circular hueca viene dada por la expresión que viene a continuación.

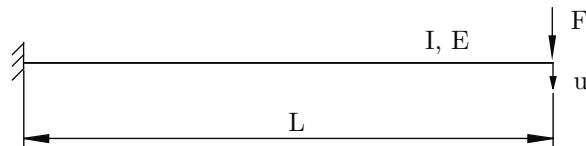


Figura 22: Viga sometida a flexión

Se recuerda que:

$$F = Ku$$

Expresión general de la rigidez a flexión de una viga: $K = \frac{3EI}{L^3}$

Inercia de una sección circular hueca: $I = \frac{\pi(r_2^4 - r_1^4)}{4}$

donde, r_1 es el radio interno y r_2 es el radio externo. Por tanto, la expresión final de la restricción quedaría de la siguiente forma:

$$K(r_1, r_2, L) = \frac{3\pi E(r_2^4 - r_1^4)}{4L^3} \geq K_o$$

Como se ha realizado con la función objetivo, en este caso, se puede simplificar mucho la inecuación, agrupando los términos de las variables conocidas, por lo que quedaría de la siguiente forma

$$(r_2^4 - r_1^4)L^{-3} \geq \tilde{K}_o \quad (10)$$

$$\text{con } \tilde{K}_o = \frac{4K_o}{3\pi E}$$

Recapitulando el planteamiento del problema, es decir, agrupando las ecuaciones (6), (7), (8), (9), (10), se tiene lo siguiente:

<p>Parámetros de diseño:</p> $\mathbf{X} = (r_1, r_2, L)$ <p>Minimizar:</p> $f(\mathbf{X}) = L(r_2^2 - r_1^2)$ <p>Sujeto a:</p> $r_2 \leq r_{2max} \quad L_{min} \leq L \leq L_{max}$ $r_1 \geq r_{1min} \quad (r_2^4 - r_1^4)L^{-3} \geq \tilde{K}_o$
--

Se observa que dicho problema de optimización plantea la minimización de una función no lineal (función objetivo), teniendo en cuenta una serie de restricciones de desigualdad no lineales, por lo que, debido a estas condiciones el problema planteado está inmerso en el proceso de resolución matemática denominado *Programación no lineal* o PNL (de aquí en adelante). En el siguiente apartado, se plantea la resolución de un problema general del tipo PNL.

8.2. Programación matemática no lineal (PNL)

La *Programación no lineal* o PNL [34] es el campo de la optimización matemática que se dedica a maximizar o minimizar una función objetivo con una serie de restricciones de igualdad y desigualdad, sujeto a una serie de variables y cuando dichas restricciones o la función objetivo no son lineales.

El método de Karush Kuhn Tucker proporciona las condiciones necesarias para que una solución sea óptima.

Problema estándar: minimizar $f(\mathbf{X})$ sujeto a $g_i(\mathbf{X}) \leq 0, h_j(\mathbf{X}) = 0$ donde $\mathbf{X} \in \mathbb{R}^p$, $f, g_i, h_j : \mathbb{R}^p \rightarrow \mathbb{R}$ (funciones escalares) con $i = 1, 2, \dots, n$ y $j = 1, 2, \dots, m$.

Teorema: Si \mathbf{X} es solución óptima del problema anterior, entonces necesariamente debe verificar las condiciones de optimalidad de Karush Kuhn Tucker (KKT) siguientes:

$$\begin{cases} \nabla f(\mathbf{X}) + \sum_{i=1}^m \mu_i \nabla g_i(\mathbf{X}) + \sum_{j=1}^m \lambda_j \nabla h_j(\mathbf{X}) = 0 \\ \mu_i g_i(\mathbf{X}) = 0, \quad i = 1, 2, \dots, n \\ \mu_i \geq 0, g_i(\mathbf{X}) \leq 0, h_j(\mathbf{X}) = 0, \quad j = 1, 2, \dots, m \end{cases}$$

Dichas condiciones son necesarias, pero no suficientes. Para que fuesen suficientes, y, por tanto, el problema tuviera solución única, las funciones y las restricciones involucradas tendrían que ser convexas.

8.3. Casos de estudio. Solución para el cilindro hueco

Los tres tipos de perfiles de sección constante que se pretende estudiar son:

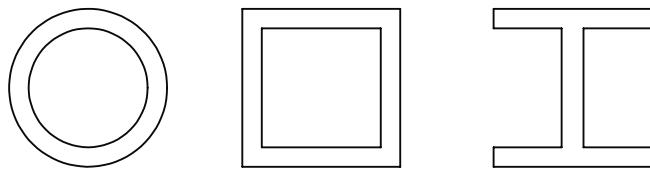


Figura 23: Secciones de perfil constante que se pretenden estudiar

Por motivos de brevedad, solamente se mostrará el caso de estudio que ha sido adoptado como solución, que es el del cilindro hueco mostrado en la Figura 21. A continuación se procede a su resolución mediante el método propuesto anteriormente.

Parámetros de diseño:

$$\mathbf{X} = (r_1, r_2, L)$$

Minimizar:

$$f(\mathbf{X}) = L(r_2^2 - r_1^2)$$

Sujeto a:

$$\begin{aligned} r_2 &\leq r_{2max} & L_{min} &\leq L \leq L_{max} \\ r_1 &\geq r_{1min} & (r_2^4 - r_1^4)L^{-3} &\geq \tilde{K}_o \end{aligned}$$

Han de verificarse las siguientes ecuaciones de optimalidad KKT, tal y como se ha indicado anteriormente, por lo que se tiene el siguiente sistema de ecuaciones e inecuaciones:

$$M = f(r_1, r_2, L) + \sum_{i=1}^5 \mu_i g_i(r_1, r_2, L) + \sum_{j=1}^5 \lambda_j h_j(r_1, r_2, L)$$

$$\begin{aligned} M = & [L(r_2^2 - r_1^2) + \mu_1(r_2 - r_{2max}) + \mu_2(r_{1min} - r_1) + \mu_3(L - L_{max}) + \mu_4(L_{min} - L) + \\ & + \mu_5(\tilde{K}_o - (r_2^4 - r_1^4)L^{-3})] \end{aligned}$$

$$\nabla M = 0$$

$$\begin{cases} \frac{\partial M}{\partial L} = (r_2^2 - r_1^2) + \mu_3 - \mu_4 + 3\mu_5(r_2^4 - r_1^4)L^{-4} = 0 \\ \frac{\partial M}{\partial r_2} = 2Lr_2 + \mu_1 - 4\mu_5r_2^3L^{-3} = 0 \\ \frac{\partial M}{\partial r_1} = -2Lr_1 - \mu_2 + 4\mu_5r_1^3L^{-3} = 0 \\ \mu_1(r_2 - r_{2max}) = 0 & \mu_2(r_{1min} - r_1) = 0 \\ \mu_3(L - L_{max}) = 0 & \mu_4(L_{min} - L) = 0 \\ \mu_5(\tilde{K}_o - (r_2^4 - r_1^4)L^{-3}) = 0 & \\ \mu_1, \mu_2, \mu_3, \mu_4, \mu_5 \geq 0 & \\ r_2 - r_{2max} \leq 0 & r_{1min} - r_1 \leq 0 \\ L - L_{max} \leq 0 & L_{min} - L \leq 0 \\ \tilde{K}_o - (r_2^4 - r_1^4)L^{-3} \leq 0 & \end{cases}$$

Se tienen 32 posibles casos debido a las ecuaciones de igualdad a cero, dichos casos son los que se muestran a continuación:

$(r_2^4 - r_1^4)L^{-3} = K_o$	$\mu_1 = 0$	$\mu_2 = 0$	$\mu_3 = 0$	$\mu_4 = 0$	(Caso I)
				$L = L_{max}$	(Caso II)
			$\mu_3 = 0$	$\mu_4 = 0$	(Caso III)
				$L = L_{min}$	(Caso IV)
				$\mu_4 = 0$	(Caso V)
				$L = L_{min}$	(Caso VI)
				$\mu_4 = 0$	(Caso VII)
				$L = L_{min}$	(Caso VIII)
				$\mu_4 = 0$	(Caso IX)
				$L = L_{min}$	(Caso X)
				$\mu_4 = 0$	(Caso XI)
				$L = L_{min}$	(Caso XII)
				$\mu_4 = 0$	(Caso XIII)
				$L = L_{min}$	(Caso XIV)
				$\mu_4 = 0$	(Caso XV)
				$L = L_{min}$	(Caso XVI)
				$\mu_4 = 0$	(Caso XVII)
				$L = L_{min}$	(Caso XVIII)
				$\mu_4 = 0$	(Caso XIX)
				$L = L_{min}$	(Caso XX)
				$\mu_4 = 0$	(Caso XXI)
				$L = L_{min}$	(Caso XXII)
				$\mu_4 = 0$	(Caso XXIII)
				$L = L_{min}$	(Caso XIV)
				$\mu_4 = 0$	(Caso XXV)
				$L = L_{min}$	(Caso XXVI)
				$\mu_4 = 0$	(Caso XXVII)
				$L = L_{min}$	(Caso XXVIII)
				$\mu_4 = 0$	(Caso XXIX)
				$L = L_{min}$	(Caso XXX)
				$\mu_4 = 0$	(Caso XXXI)
				$L = L_{min}$	(Caso XXXII)
$\mu_5 = 0$	$r_2 = r_{2max}$	$\mu_2 = 0$	$\mu_3 = 0$	$\mu_4 = 0$	(Caso I)
				$L = L_{max}$	(Caso II)
			$\mu_3 = 0$	$\mu_4 = 0$	(Caso III)
				$L = L_{min}$	(Caso IV)
				$\mu_4 = 0$	(Caso V)
				$L = L_{min}$	(Caso VI)
				$\mu_4 = 0$	(Caso VII)
				$L = L_{min}$	(Caso VIII)
				$\mu_4 = 0$	(Caso IX)
				$L = L_{min}$	(Caso X)
				$\mu_4 = 0$	(Caso XI)
				$L = L_{min}$	(Caso XII)
				$\mu_4 = 0$	(Caso XIII)
				$L = L_{min}$	(Caso XIV)
				$\mu_4 = 0$	(Caso XV)
				$L = L_{min}$	(Caso XVI)
				$\mu_4 = 0$	(Caso XVII)
				$L = L_{min}$	(Caso XVIII)
				$\mu_4 = 0$	(Caso XIX)
				$L = L_{min}$	(Caso XX)
				$\mu_4 = 0$	(Caso XXI)
				$L = L_{min}$	(Caso XXII)
				$\mu_4 = 0$	(Caso XXIII)
				$L = L_{min}$	(Caso XIV)
				$\mu_4 = 0$	(Caso XXV)
				$L = L_{min}$	(Caso XXVI)
				$\mu_4 = 0$	(Caso XXVII)
				$L = L_{min}$	(Caso XXVIII)
				$\mu_4 = 0$	(Caso XXIX)
				$L = L_{min}$	(Caso XXX)
				$\mu_4 = 0$	(Caso XXXI)
				$L = L_{min}$	(Caso XXXII)

De estos 32 casos, se han subrayado los 4 casos que cumplen las condiciones KKT.

Se tienen por tanto, 4 posibles casos, de los cuales dos de ellos tienen la solución con $L = L_{min} = L_{max}$, lo cual no tiene sentido, a no ser que $L_{min} = L_{max}$, en cuyo caso es como fijar a priori la longitud. Se omitirán dichos casos, puesto que se resolverá posteriormente el problema para una longitud fija. Quedan, por tanto, dos casos posibles como solución:

- $r_2 = r_{2max}, r_1 = \sqrt[4]{r_{2max}^4 - \tilde{K}_o L_{min}^3}, L = L_{min}$
- $r_2 = r_{2max}, r_1 = r_{1min}, L = L_{min}$

La función a optimizar es $f = (r_2^2 - r_1^2)L$, se ve claramente que L no influiría (ya que es L_{min} en ambos casos), r_2 tampoco (ya que en ambos casos es r_{2max}), por tanto, el valor para r_1 que hace mínimo el volumen es $r_1 = \sqrt[4]{r_{2max}^4 - \tilde{K}_o L_{min}^3}$

Por lo que, los valores que hacen mínimo el volumen son:

$$r_2 = r_{2max}, r_1 = \sqrt[4]{r_{2max}^4 - \tilde{K}_o L_{min}^3}, L = L_{min}$$

Sustituyendo el valor de \tilde{K}_o para el caso de la sección cuadrada hueca se tiene lo siguiente:

$$r_2 = r_{2max}, r_1 = \sqrt[4]{r_{2max}^4 - \frac{4K_o L_{min}^3}{3\pi E}}, L = L_{min}$$

(11)

Con los valores óptimos de las variables de diseño, se obtiene el volumen óptimo para una estructura a flexión con sección circular:

$$f = \pi L \left(r_{2max}^2 - \sqrt{r_{2max}^4 - \frac{4K_o L_{min}^3}{3\pi E}} \right)$$

(12)

Conclusiones: De la resolución de este problema, se pueden sacar varias conclusiones importantes:

- La restricción de $L \geq L_{min}$ se queda siempre activa porque el valor óptimo es alcanzado en la igualdad $L = L_{min}$.

Se ve que la solución óptima se da para el mínimo de longitud, esto es así puesto ya que se quiere minimizar el volumen, que depende directamente de la longitud, por lo que el volumen va a ser mínimo cuando la longitud sea la mínima posible.

- Se tiene que la restricción $K \geq K_o$ queda activa $K = K_o$.

Se recuerda la fórmula de la rigidez para el caso de una viga a flexión con sección circular hueca:

$$K(r_1, r_2, L) = \frac{3EI}{L^3} = \frac{3\pi E(r_2^4 - r_1^4)}{4L^3} \geq K_o$$

La rigidez va a aumentar si se tiene una inercia alta y una longitud baja, por esto la longitud va a tender a la mínima (ya que una longitud pequeña proporciona rigidez y reduce volumen). La inercia es lo que se va a 'ajustar' para reducir el volumen y a la vez la rigidez, por lo que se trata de buscar una inercia baja, es por esto, que el volumen mínimo se alcanza para la rigidez mínima, esto es, K_o .

- $r_2 = r_{2max}$ y $r_1 = \sqrt[4]{r_{2max}^4 - \tilde{K}_o L_{min}^3}$: En cuanto al radio externo, la restricción $r_2 \leq r_{2max}$ queda también activa, lo que se consigue con esto, es que al tener el radio externo máximo posible, se va a tener una alta inercia. De esta manera va a ser el radio interno el que se 'ajuste' buscando el mínimo volumen con el que se consigue la rigidez mínima indicada.

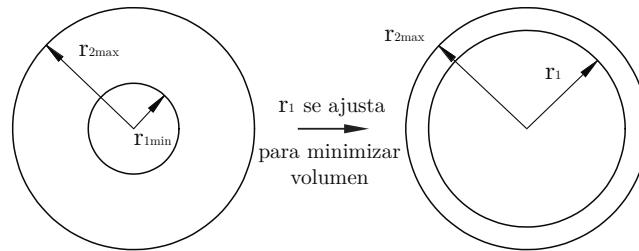


Figura 24: Forma en la que el diseño inicial se ajusta a la sección circular óptima

Teniendo en cuenta lo que se ha dicho en las conclusiones, de ahora en adelante, las restricciones de $L_{min} \leq L \leq L_{max}$ y $K \geq K_o$ serán remplazadas por otras de

igualdad, es decir, se tomarán siempre valores de longitud y rigidez conocidos $K = K_o$ y $L = L_o$.

8.3.1. Restricción adicional: mínimo espesor de impresión

Se puede poner el caso en el que se necesite muy poca rigidez en un viga de sección circular hueca con un gran radio externo. Lo que indica la solución óptima es que el radio externo va a ser el máximo y el radio interno se va a ajustar para conseguir esa rigidez (de tal manera que el volumen es el mínimo). En este caso, es fácil observar que se obtendría un radio interno prácticamente igual que el externo.

$$r_2 = r_{2max}, \quad r_1 = \sqrt[4]{r_{2max}^4 - K_o L^3} \quad (13)$$

Si la rigidez que se pide es muy pequeña, el radio interno tiene prácticamente al radio externo. Esto genera un gran problema, puesto que, como se recalca, a pesar de que matemáticamente sea correcto, no se puede fabricar un estructura que apenas tenga espesor.

Es por esto, que se tendrá que recurrir a un nuevo planteamiento, donde se impondrá una nueva condición: el perfil debe tener un espesor mínimo, con el objetivo de que sea posible fabricarlo en la realidad.

Los parámetros de diseño serían de nuevo los radios del perfil circular hueco (radio interno y externo):

$$\mathbf{X} = (r_1, r_2)$$

El volumen de una viga de sección circular hueca, que es lo que se quiere minimizar sería:

$$V(r_1, r_2) = \pi L(r_2^2 - r_1^2)$$

Se puede simplificar la expresión a minimizar para facilitar los cálculos:

$$f(\mathbf{X}) = (r_2^2 - r_1^2) \quad (14)$$

Las restricciones en este nuevo planteamiento serían tener un radio externo máximo, no se impondrá un radio interno mínimo, puesto que no tiene influencia en el cálculo del volumen óptimo (como se vio anteriormente en el caso circular).

$$r_2 \leq r_{2max} \quad (15)$$

La restricción de un espesor mínimo dado, para que sea factible fabricarlo, sería la siguiente:

$$r_2 - r_1 \geq e \quad (16)$$

Además, la estructura tiene que ir sujetada a una rigidez a flexión dada. De nuevo, la rigidez a flexión de una viga de sección circular hueca viene dada por la siguiente fórmula:

Rigidez a flexión de una viga: $K = \frac{3EI}{L^3}$

Inercia de una sección circular hueca: $I = \frac{\pi(r_2^4 - r_1^4)}{4}$

$$K(r_1, r_2) = \frac{3\pi E(r_2^4 - r_1^4)}{4L^3} = K_o$$

Agrupando términos de las variables conocidas, quedaría de la siguiente forma:

$$(r_2^4 - r_1^4) = \tilde{K}_o \quad (17)$$

Con $\tilde{K}_o = \frac{4L^3K_o}{3\pi E}$

Recapitulando el planteamiento del problema, es decir, agrupando las ecuaciones (14), (15), (16), (17) se tiene lo siguiente:

Parámetros de diseño: $\mathbf{X} = (r_1, r_2)$ Minimizar: $f(\mathbf{X}) = (r_2^2 - r_1^2)$ Sujeto a: $r_2 \leq r_{2max}$ $r_2 - r_1 \geq e$ $(r_2^4 - r_1^4) = \tilde{K}_o$
--

Como en cada uno de los casos que se han ido analizando, han de verificarse las ecuaciones de optimalidad KKT, por lo que se tiene el siguiente sistema de ecuaciones e inecuaciones:

$$M = \left[(r_2^2 - r_1^2) + \mu_1(r_2 - r_{2max}) + \mu_2(r_1 - r_2 + e) + \lambda((r_2^4 - r_1^4) - \tilde{K}_o) \right]$$

$$\nabla M = 0$$

$$\begin{cases} \frac{\partial M}{\partial r_2} = 2r_2 + \mu_1 - \mu_2 + 4\lambda r_2^3 = 0 \\ \frac{\partial M}{\partial r_1} = -2r_1 + \mu_2 - 4\lambda r_1^3 = 0 \\ \mu_1(r_2 - r_{2max}) = 0 \quad \mu_2(r_1 - r_2 + e) = 0 \\ \mu_1, \mu_2 \geq 0 \\ r_2 - r_{2max} \leq 0 \quad r_1 - r_2 + e \leq 0 \\ (r_2^4 - r_1^4) = \tilde{K}_o \end{cases}$$

Se tienen 4 posibles casos debido a las ecuaciones de igualdad a cero, dichos casos son los que se muestran a continuación:

$$\begin{cases} \mu_1 = 0 & \begin{cases} \mu_2 = 0 & \text{(Caso I)} \\ r_2 - r_1 = e & \text{(Caso II)} \end{cases} \\ r_2 = r_{2max} & \begin{cases} \mu_2 = 0 & \text{(Caso III)} \\ r_2 - r_1 = e & \text{(Caso IV)} \end{cases} \end{cases}$$

En el esquema anterior se han subrayado los 3 casos que serían válidos, a continuación se procede a buscar el caso óptimo.

De los 3 posibles casos, a continuación se muestran cuál se omitirá directamente por ser un caso muy restrictivo:

- Caso IV: $r_2 = r_{2max}$, $r_1 = \sqrt[4]{r_{2max}^4 - \tilde{K}_o} = r_{2max} - e$

Es prácticamente imposible que se cumplan esto puesto que sería mucha casualidad que dichos radios estuvieran sujetos a una rigidez dada y además mantuvieran un espesor también dado.

Quedan, por tanto, 2 casos posibles como solución:

- $r_2 = r_{2max}$, $r_1 = \sqrt[4]{r_{2max}^4 - \tilde{K}_o}$

Esta solución es la que había salido en el primero caso de estudio, similar a éste sólo que no se había considerado el espesor mínimo. Este caso sería el

óptimo siempre y cuando el espesor sea mayor al indicado. Si esto no fuera así, el caso óptimo sería el siguiente.

- Cuando $r_2 - r_1 = e$, $r_2^4 - r_1^4 = \tilde{K}_o$

Si el espesor de la solución anterior no supera el espesor mínimo indicado, ésta sería la solución óptima, asegurando que se cumple tanto espesor como rigidez.

Debido a esto, no se tendría una única solución válida para todos los supuestos, los valores que hacen mínimo el volumen serían:

$$\text{Si: } r_2 - r_1 \geq e$$

$$r_2 = r_{2max}, \quad r_1 = \sqrt[4]{r_{2max}^4 - \tilde{K}_o}$$

$$\text{Si: } r_2 - r_1 < e$$

Los radios que cumplen: $r_2 - r_1 = e$, $r_2^4 - r_1^4 = \tilde{K}_o$ Es decir, los siguientes:

$$r_2 = \frac{e}{2} - \frac{e^2}{12 \left(\frac{\tilde{K}_o}{8e} + \left(\frac{e^6}{1728} + \frac{\tilde{K}_o^2}{64e^2} \right)^{1/2} \right)^{1/3}} + \left(\frac{\tilde{K}_o}{8e} + \left(\frac{e^6}{1728} + \frac{\tilde{K}_o^2}{64e^2} \right)^{1/2} \right)^{1/3}$$

$$r_1 = -\frac{e}{2} - \frac{e^2}{12 \left(\frac{\tilde{K}_o}{8e} + \left(\frac{e^6}{1728} + \frac{\tilde{K}_o^2}{64e^2} \right)^{1/2} \right)^{1/3}} + \left(\frac{\tilde{K}_o}{8e} + \left(\frac{e^6}{1728} + \frac{\tilde{K}_o^2}{64e^2} \right)^{1/2} \right)^{1/3}$$

Donde el volumen óptimo quedaría:

$$f = \pi L(r_2^2 - r_1^2) \quad \text{Con los valores anteriores} \quad (18)$$

Conclusiones: De este caso de estudio se pueden sacar unas ideas generales muy interesantes:

- La solución óptima es la misma que para la viga con perfil circular hueco, $r_2 = r_{2max}$, $r_1 = \sqrt[4]{r_{2max}^4 - \tilde{K}_o}$, en el caso que la diferencia de radios proporcione una magnitud mayor o igual al espesor que se pide. Es lógico que salga este resultado, puesto que si el espesor es mayor al que se pide, no

haría falta estudiar el problema en cuestión. Como se dijo anteriormente, el radio externo será el máximo posible para obtener la mayor inercia posible, mientras que el radio interno es el que se ajustará de tal forma que hará disminuir volumen (y por tanto inercia) hasta ajustarse a la inercia pedida, minimizando así el volumen.

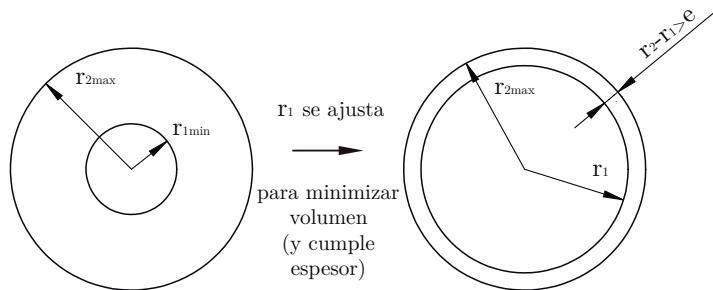


Figura 25: Sección circular óptima si cumple espesor

- Lo más interesante viene cuando la solución anterior no genera un espesor mayor o igual al que se exige en el problema. La solución es la siguiente: $r_2 - r_1 = e$, $r_2^4 - r_1^4 = \tilde{K}_o$. Los radios deben cumplir ambas ecuaciones, puesto que no se ajustan al espesor, el problema se soluciona manteniendo el espesor. De esta forma son dichos radios (que mantienen el espesor) los que se ajustan para conseguir la rigidez indicada, minimizando, por tanto, el volumen de la viga. A continuación, se muestra un dibujo que puede ayudar a la compresión de lo que está ocurriendo.

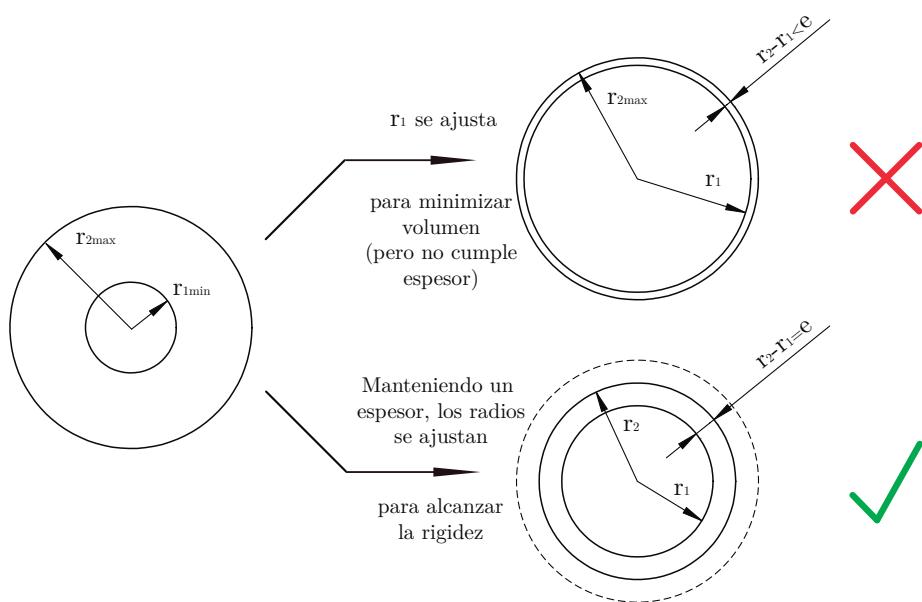


Figura 26: Sección circular válida si la solución óptima no cumple espesor

9. Anexo C. Estudio dinámico de eslabones

En la Sección 5.2 se explica cómo se relaciona la sección buscada para el eslabón para unas solicitudes debidas principalmente a una carga que provoca momento flector. En el caso real, esta carga está formada por las cargas estáticas que soporta el manipulador y las fuerzas de inercia debido a los distintos movimientos del manipulador. Hallar dichas cargas trae los siguientes problemas:

1. Las fuerzas de cada eslabón o elemento pasivo va a resultar de una composición de movimientos independientes, teniendo como grados de libertad la posición, velocidad y aceleración de cada servo, es decir 9 (3 en cada grado de libertad, 9 en total).
2. Debe buscarse la peor situación que solicite la estructura del manipulador, para así asegurarse que el diseño es válido para el correcto desempeño del manipulador.
3. Los torques del motor también deben ser conocidos, ya que el torque desempeñado por el motor es soportado por la estructura. También, al conocer el torque requerido, se puede seleccionar el manipulador que va a ser necesario, entre los distintos servomotores disponibles.

Para resolver la dinámica del manipulador, formada por composición de movimientos y teniendo varios grados de libertad se va a usar la formulación Newton-Euler [18].

Para ello, se han usado una serie de ecuaciones que parten de la segunda ley de Newton. Tras desarrollarse las ecuaciones de movimiento que provienen de Newton-Euler se llega a una serie de ecuaciones cinemáticas, basadas en el principio de d'Alembert, que describen el movimiento de los elementos del manipulador que se mueven respecto al sistema de referencia fijo de la base.

Las ecuaciones que se obtienen de resultado para la resolución de la dinámica son una serie de ecuaciones recursivas hacia adelante y hacia atrás.

- Las ecuaciones hacia adelante van a servir para describir la cinemática de cada elemento, que parte desde el sistema de referencia fijo de la base, que es el sistema inercial, referenciándose a los siguientes sistemas de referencias, uno en cada extremo del eslabón siguiente y referenciándose desde el anterior.

Esto tiene gran utilidad al partir el siguiente eslabón de la posición, velocidad y aceleración del anterior y añadiéndose las velocidades y aceleraciones que les confieren los movimientos de los propios grados de libertad del siguiente eslabón.

- Luego están las ecuaciones recursivas hacia atrás que nos muestran la dinámica del manipulador, es decir, las fuerzas y torques que soporta cada eslabón sometidos a ese estado cinemático. La principal razón de que son ecuaciones para atrás, es que la carga y solicitudes del efecto final, además de sus fuerzas de inercia, son soportadas por el último eslabón. Luego las de este eslabón son soportadas por el anterior, así hasta que se llega al primer sistema de referencias, que es el de la base, donde se soportan todas las solicitudes del manipulador. Para ello es importante haber resuelto con anterioridad las ecuaciones hacia adelante, que permiten añadir las fuerzas de inercia y conocer las fuerzas necesarias para realizar esos movimientos.

Antes de empezar a explicar las ecuaciones recursivas con mayor profundidad, se debe explicar la matriz de rotación de los distintos sistemas de referencias ${}^f R_i$, que giran al vector por el que multiplican, de un vector en el sistema de referencias i (que es en el que está el vector sin multiplicar) al vector representado en el sistema de referencias f. Cada sistema de referencias está orientado de distinta forma para decir propiedades de la posición en el manipulador siguiendo las indicaciones de Denavit-Hartenberg, cuyos rasgos más significativos en este manipulador son:

1. Eje Z con sentido del movimiento de la articulación.
2. Eje X suele estar orientado en la dirección que lleva el eslabón longitudinalmente (cuando la articulación es axial). Si la articulación es radial, el eje Z está orientado en la dirección longitudinal del eslabón, y el $X_i = Z_{i-1} \times Z_i$.
3. El eje Y está posicionado para completar cada sistema de referencias como sistema dextrógiro.
4. Por último está el sistema de referencia del *end-effector* o efecto final, en el cual tiene el eje Z orientado en la dirección del último eslabón, el eje X en el sentido en el cual estaría la abertura de una pinza o mano, y el eje Y con orientación para dotar al sistema de referencia de dextrógiro.

Se debe de tener en cuenta, que el sistema de referencias inercial sera el 0, y el nº para nombrar al sistema de referencia va aumentando según transmite el movimiento cada grado de libertad en orden ascendente.

La razón de usar estas rotaciones de los sistemas de referencias es debida a que las matrices de inercia, y los distintos parámetros físico-geométricos del manipulador como son vectores de posición entre las posiciones de distintas articulaciones y centros de masas no son variables si se toma solo el sistema de referencia inercial de la base. Por esa razón, los cálculos se simplifican si se llevan a cabo en cada eslabón desde el sistema de referencias pertinente y luego se multiplica por la matriz de rotación, obteniendo el deseado. Cabe decir que el producto de matrices de rotación como el del siguiente ejemplo, da lugar a una matriz de cambio de sistema de referencia del origen de la primera al final de la segunda como podemos ver en la siguiente ecuación:

$${}^a R_c = {}^a R_b {}^b R_c \quad (19)$$

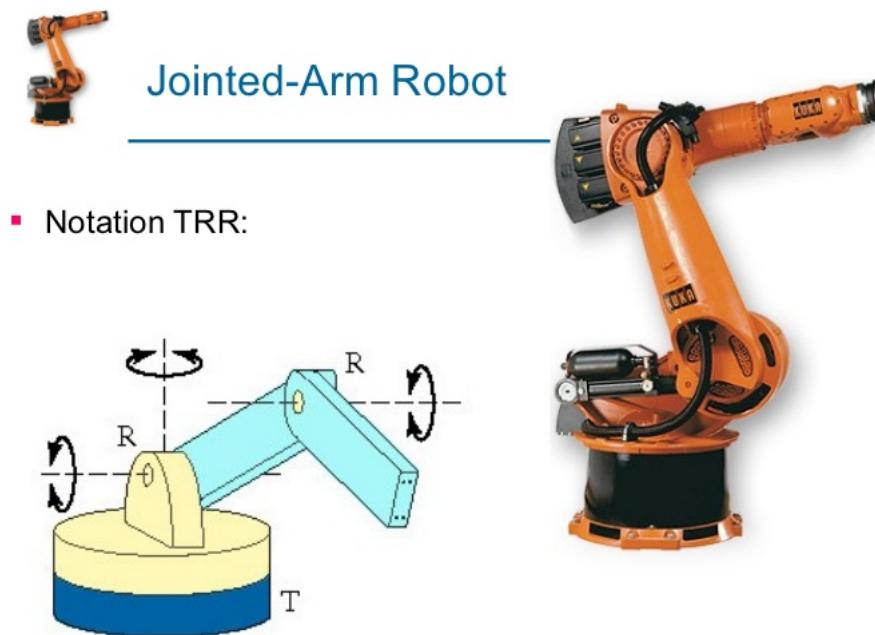


Figura 27: Manipulador 3 grados de libertad. Fuente: <http://slideplayer.com/>

Las ecuaciones hacia adelante en un sistema de n grados de libertad para

un manipulador como el que hemos estudiado, cuyas articulaciones son todas de rotación, son las siguientes:

- **Velocidad angular del elemento i** , ω_i , que depende de la velocidad angular del elemento anterior, ω_{i-1} ; y la velocidad angular de la articulación, \dot{q}_i que esta resaltado que tiene la dirección del eje z_0 en el sistema de referencias anterior, como vemos en la siguiente fórmula.

$${}^i R_0 \omega_i = {}^i R_{i-1} ({}^{i-1} R_0 \omega_{i-1} + z_0 \dot{q}_i) \quad (20)$$

- **Aceleración angular del elemento i** , $\dot{\omega}_i$, que depende de la velocidad, ω_{i-1} , y aceleración angular, $\ddot{\omega}_{i-1}$, del elemento anterior, la aceleración angular dada por la articulaci'on \ddot{q}_i , la velocidad angular de la articulaci'on anterior, \dot{q}_i , que van en el sentido z_0 del sistema de referencia anterior, $i - 1$, tal como se aprecia en la siguiente expresión.

$${}^i R_0 \dot{\omega}_i = {}^i R_{i-1} [{}^{i-1} R_0 \dot{\omega}_{i-1} + z_0 \ddot{q}_i + ({}^{i-1} R_0 \omega_{i-1}) \times z_0 \dot{q}_i] \quad (21)$$

- **Aceleración lineal en la articulación del elemento i** , \dot{v}_i , que depende de la aceleración angular, $\dot{\omega}_i$; la distancia entre la articulación y la anterior, p_i^* ; la velocidad angular, ω_i ; y la aceleración lineal de la articulación anterior, \ddot{v}_{i-1} , como se puede ver en la siguiente expresión.

$${}^i R_0 \dot{v}_i = ({}^i R_0 \dot{\omega}) \times ({}^i R_0 p_i^*) + ({}^i R_0 \omega_i) \times [({}^i R_0 \omega_i) \times ({}^i R_0 p_i^*)] + {}^i R_{i-1} ({}^{i-1} R_0 \dot{v}_{i-1}) \quad (22)$$

- **Aceleración lineal en el centro de masas del eslabón i** , \ddot{a}_i , que depende de la aceleración angular, $\dot{\omega}_i$; la distancia desde la articulación al centro de masas del eslabón, \bar{s}_i ; la velocidad angular, ω_i ; y la velocidad lineal en la articulación, \dot{v}_i , como se indica en la siguiente expresión.

$${}^i R_0 \ddot{a}_i = ({}^i R_0 \dot{\omega}_i) \times ({}^i R_0 \bar{s}_i) + ({}^i R_0 \omega_i) \times [({}^i R_0 \omega_i) \times ({}^i R_0 \bar{s}_i)] + {}^i R_0 \ddot{v}_i \quad (23)$$

Como se ve en estas ecuaciones hacia adelante, al empezar con $i = 1$ en la primera articulación se puede apreciar que se necesita los términos de velocidad

angular ω_0 , aceleración angular $\dot{\omega}_0$ que son iguales al vector nulo $\bar{0}$. Además, se necesita la aceleración lineal en la articulación \ddot{v}_0 :

$$\ddot{v}_0 = \begin{pmatrix} g_x \\ g_y \\ g_z \end{pmatrix} \quad (24)$$

$$|g| = 9,8062 \text{ m/s}^2 \quad (25)$$

Tras haberse explicado las ecuaciones hacia adelante y poder resolver la cinemática, debajo se procede a explicar las ecuaciones hacia atrás, que resuelven las fuerzas y momentos necesarios para la anterior cinemática se dé lugar en el manipulador. Al igual que en las ecuaciones hacia adelante, se va a resolver para un sistema de n grados de libertad. También se vuelve aclarar que para el caso de estudio, que es donde se aplican estas ecuaciones, todas las articulaciones son de rotación.

- **Fuerza en la articulación i , f_i** , que depende de la fuerza de la articulación posterior, f_{i+1} ; de la masa del eslabón i , m_i ; y de la aceleración lineal del centro de masas i , \bar{a}_i , como puede observarse en la expresión siguiente.

$${}^i R_0 f_i = {}^i R_{i+1} ({}^{i+1} R_0 f_{i+1}) + m_i^i R_0 \bar{a}_i \quad (26)$$

- **Momento en la articulación i , n_i** , en el que intervienen el momento en la articulación posterior, n_{i+1} ; la distancia entre la articulación i y la anterior, p_i^* ; la fuerza aplicada en la articulación posterior, f_{i+1} ; la distancia desde la articulación i al centro de masas del eslabón i , \bar{s}_i ; la masa del eslabón i , m_i ; la aceleración del centro de masas i , \bar{a}_i ; el momento de inercia del eslabón i desde su centro de masas, I_i ; la aceleración angular i , $\dot{\omega}_i$; la velocidad angular i , ω_i , como vemos en la siguiente expresión.

$$\begin{aligned} {}^i R_0 n_i = & {}^i R_{i+1} [{}^{i+1} R_0 n_{i+1} + ({}^{i+1} R_0 p_i^*) \times ({}^{i+1} R_0 f_{i+1})] + ({}^i R_0 p_i^* + {}^i R_0 \bar{s}_i) \times \\ & \times (m_i^i R_0 \bar{a}_i) + ({}^i R_0 I_i^0 R_i) ({}^i R_0 \dot{\omega}_i) + ({}^i R_0 \omega_i) [({}^i R_0 I_i^0 R_i) ({}^i R_0 \omega_i)] \end{aligned} \quad (27)$$

- **Torque necesario en motor de la articulación i , τ_i** , es el torque necesario en el motor para desarrollar el movimiento estudiado. Depende de el

momento en el eslabón i , n_i ; aunque debido al producto escalar con el eje +Z del sistema de referencia $i - 1$, solo nos interesa la componente del momento en dicha dirección. Además, también depende del coeficiente de amortiguamiento viscoso de dicha articulación, b_i , y de la velocidad de la articulación i que va a desempeñar el torque necesario, \dot{q}_i , de la forma que vemos siguiente fórmula.

$$\tau_i = (^i R_0 n_i)^T (^i R_{i-1} z_0) + b_i \dot{q}_i \quad (28)$$

Tras haberse explicado todo el procedimiento Newton-Euler para calcularse la dinámica, se ve que los últimos pasos del procedimiento obtenía los momentos y las fuerzas. Estas fuerzas (5.27) y momentos (5.28) son los aplicados en cada sistema de referencia. Entonces, con estos datos tenemos los esfuerzos aplicados en la sección más solicitada del eslabón, que son usados para conocer los parámetros de diseño del eslabón, como se vio en el apartado 4.

10. Anexo D. Análisis de fallo de módulos pasivos

Como paso último, en la parametrización de los eslabones óptimos se debe verificar la ausencia de fallo en la estructura del eslabón. Para ello se han aplicado teorías de fallo, en las que se usan las solicitudes máximas que tiene el eslabón, explicadas en la Sección 9.

10.1. Criterio de Von Mises

Para verificar que el elemento pasivo diseñado por los parámetros de actuación del manipulador no va a producir rotura, se le va a aplicar siempre una teoría de fallo. Para un caso como el que nos ocupa, de un eslabón sometido principalmente a flexión, se ha utilizado el criterio de Von Mises.

El criterio de Von Mises es un criterio que chequea si se produce fallo o no para materiales dúctiles midiendo la energía de distorsión. Para ello, utiliza la tensión de Von Mises, σ_{VM} .

$$\sigma_{VM} = \sqrt{(\sigma_1 - \sigma_2)^2 + (\sigma_1 - \sigma_3)^2 + (\sigma_2 - \sigma_3)^2} \quad (29)$$

En la ecuación anterior, las tensiones σ_1 , σ_2 y σ_3 son las tensiones principales del punto donde se va a calcular la tensión de Von Mises.

La tensión de Von Mises necesita conocer las tensiones del punto con mayor estado tensional del eslabón va a tener, para comprobar si falla o en cambio resiste el estado tensional. El criterio que se debe cumplir es el siguiente:

$$UTS \geq \sigma_{VM} \quad (30)$$

Tal como apuntaba la expresión (4.6), la tensión de rotura (UTS) del PLA es de 50 MPa.

Para calcular la tensión de Von Mises, en el caso que nos ocupa vamos a tener los momentos en cada articulación (5.28) y la fuerza aplicada en cada articulación (5.27). Estos datos son usados para calcular las tensiones aplicadas en la sección más solicitada, que es donde los momentos son mayores, tal como se indica en [15]. Los momentos utilizados tienen el aspecto de la siguiente imagen.

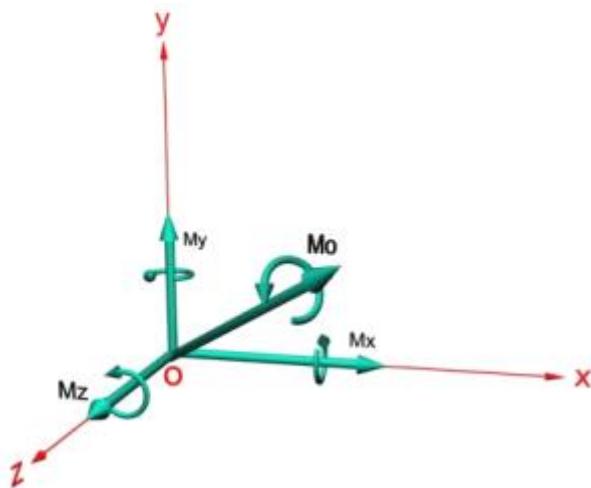


Figura 28: Momentos en el empotramiento. Fuente: <http://es.slideshare.net/>

Como se puede observar, hay 3 momentos: 1 momento torsor M_x y 2 momentos flectores M_y y M_z . Estos momentos son la descomposición del vector momento que se obtiene con la fórmula (5.28), denominado M en la figura 5, siendo cada uno de los momentos M_x , M_y y M_z una componente del vector (5.28). Se debe tener en cuenta que uno de los momentos flectores es el torque máximo que provee el motor, por lo que en esa dirección se sustituye, para así tener el momento máximo que

se puede dar en la peor situación, y se procede como en las siguientes ecuaciones para conseguir el momento flector con el que se calcula las tensiones para probar los criterios de fallo.

$$M_z = \tau_{max\ motor} \quad (31)$$

$$Momento\ flector \equiv M = \sqrt{M_z^2 + M_y^2} \quad (32)$$

$$Torsor \equiv T = M_x \quad (33)$$

Tras tenerse bien diferenciado los momentos flectores y torsores, además de obtener el momento usando el torque máximo dado por el motor en la peor situación, se va a proceder a calcular las tensiones. Para un caso como el que nos ocupa, en que se obtienen los momentos así, la ecuación (5.29) no va a ser utilizada, si no que es más útil usarla con las tensiones tangenciales y normales como en la siguiente expresión, en la que no considera que se tienen las tensiones principales.

$$\sigma_{VM} = \sqrt{\sigma_{xx}^2 + \sigma_{yy}^2 + \sigma_{zz}^2 - (\sigma_{xx}\sigma_{yy} + \sigma_{xx}\sigma_{zz} + \sigma_{yy}\sigma_{zz}) + 3(\tau_{xy}^2 + \tau_{yz}^2 + \tau_{xz}^2)} \quad (34)$$

En nuestro caso, las tensiones en el lugar de mayor estado tensional únicamente son σ_{xx} y τ_{xy} , que en este caso que se estudia, es debido al torsor, por lo que pasa a denominarse τ_r . Las tensiones σ_{xx} y τ_r se calculan de la siguiente forma:

$$\sigma_{xx} = \frac{|M| R}{I} + \frac{F_x}{A} \quad (35)$$

$$\tau_r = \frac{TR}{\frac{\pi}{2}(R^4 - r^4)} \quad (36)$$

Como se ve en las ecuaciones anteriores, el lugar en donde se produce el mayor estado tensional es en el radio exterior, ya que la tensión normal σ_{xx} es mayor cuanto más alejada de la fibra neutra, donde la tensión es 0. Tras tener los datos de las tensiones, la expresión (5.34) es usada para verificar lo que se expresa en la siguiente ecuación.

$$UTS \geq \sqrt{\sigma_{xx}^2 + 3\tau_r^2} \quad (37)$$

10.2. Concentrador de tensiones

10.2.1. Diseño y cálculo de concentrador de tensiones

Tras haberse mostrado la manera en que se va conocer los distintos parámetros para las dimensiones del eslabón óptimo, se conoce como encontrarse las tensiones

máximas del eslabón, se ha tenido que abordar el problema de la unión base del eslabón con la base o inicio del eslabón. El principal motivo por el que se ha de estudiar este caso es debido a que si la unión tiene un crecimiento de sección muy brusco, pudiéndose dar incluso una arista viva, ahí se produce un concentrador de tensiones [35], amplificando el estado tensional del se había hallado en el apartado anterior.

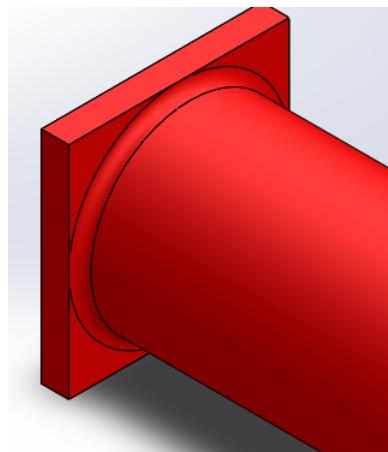


Figura 29: Redondeo de unión en la base del eslabón

Precisamente, la unión base del eslabón - eslabón es donde se da lugar la sección más solicitada y en donde se ha centrado el estudio de las fuerzas y las tensiones.

El concentrador de tensiones se muestra en las siguientes expresiones, siendo la primera de ellas para tensión normal y la siguiente para tensión cortante.

$$K_t = \frac{\sigma_{max}}{\sigma_{nom}} \quad (38)$$

$$K_{ts} = \frac{\tau_{max}}{\tau_{nom}} \quad (39)$$

Para abordar el cálculo de las tensiones, la ecuación (6.38) se utiliza aplicando los concentradores de tensiones, quedando de la siguiente forma:

$$UTS \geq \sqrt{(K_t \sigma_{xx})^2 + 3(K_{ts} \tau_r)^2} \quad (40)$$

Para conocerse el valor del concentrador de tensiones, se va a hacer uso de tablas, en las que se puede conocer el valor a través de varias variables. Las variables son la relación entre el diámetro del eslabón, d , el diámetro hasta el que se ensancha

con el redondeo , D , y el radio del redondeo , r . Las tablas pueden observarse a continuación, en las figuras 6 y 7, siendo la primera para la tensión creada por el momento flector, y la segunda, la tabla creada por el momento torsor.

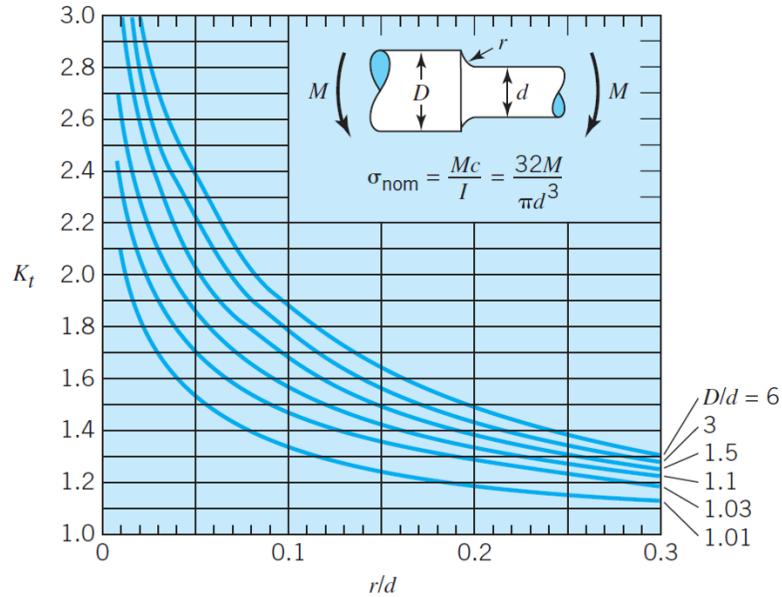


Figura 30: Momentos en el empotramiento. Fuente: Fundamentals of Machine Component Design

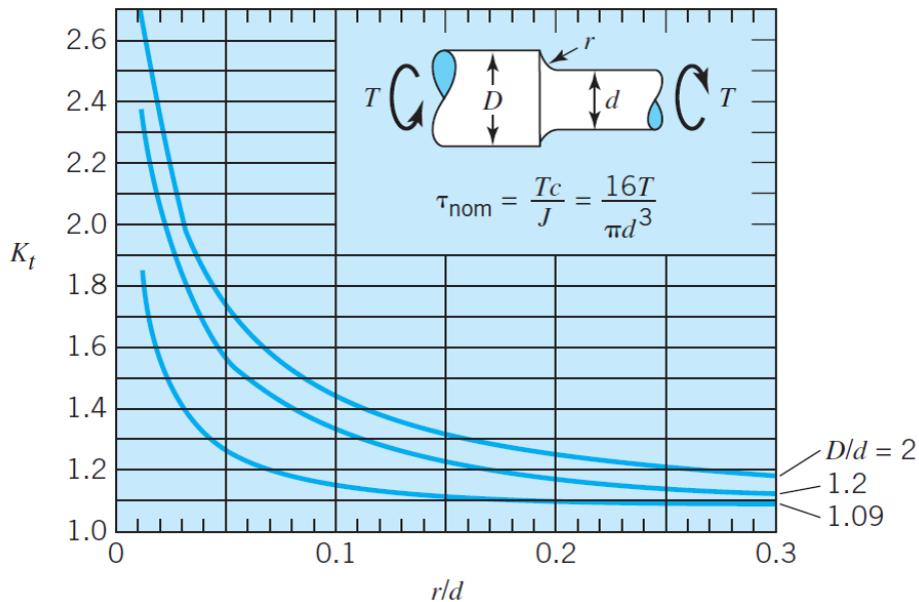


Figura 31: Momentos en el empotramiento. Fuente:Fundamentals of Machine Component Design

Como se ve en las tablas que el concentrador de tensiones depende del radio de redondeo, el redondeo debe de ser tangente tanto a la base del eslabón como al cilindro, para así evitar la aparición de un redondeo más pequeño que el redondeo y que se concentren las tensiones de formas inesperadas.

10.2.2. Simulación de concentrador de tensiones

Tal como se ha comentado antes, cada eslabón tiene al inicio una base (y al final) cuya utilidad es que tiene colas de milano para poder conectarse a otros elementos. La base es un cuadrado de 40x40 mm. En la ecuación (3.3) se ve que para optimizar los eslabones, se debía igualar el radio exterior R con un radio máximo fijado, en función del que fijemos se puede dar un redondeo mayor(r en las figuras 7 y 8). Ya que la información de las figuras 7 y 8 puede no representar casos como el estudiado en este caso, se va a modelar distintos eslabones, optimizándose como se ha explicado anteriormente, para ver que tensión se obtiene y si el concentrador de tensiones es acorde a lo que señalan las tablas.

Los siguientes parámetros se dan en todos los eslabones testados:

$$\text{Longitud } L = 100 \text{ mm} \quad (41)$$

$$E_{PLA} = 2,8 \text{ MPa} \quad (42)$$

$$\text{desplazamientos verticales } u = \frac{L}{500} \quad (43)$$

$$e = 2 \text{ mm} \quad (44)$$

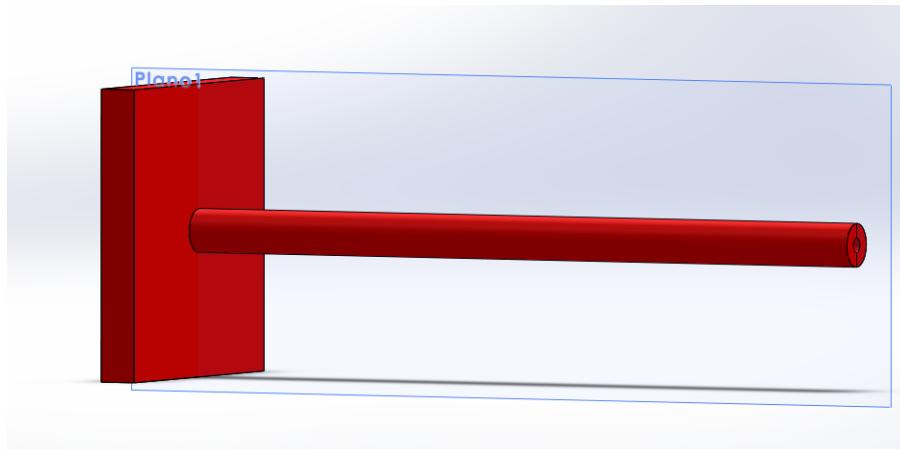


Figura 32: Eslabón para simulación de R3 y r1 con arista viva

1. Caso de viga m'as fina R=3 mm y r=1 mm en arista viva

En este caso, se ha simulado la viga más fina ya que el radio interior menor posible es 1 mm, y el espesor $e = 2\text{mm}$. El momento de inercia obtenido con dicha sección es el siguiente:

$$I = 6,283 \times 10^{-11} \text{ m}^4 \quad (45)$$

En la simulación se va a ajustar los datos para que en la optimización se obtenga dichos radios del eslabón. Para ello se debe aplicar la siguiente carga en el extremo:

$$P = 0,106 \text{ N} \quad (46)$$

El momento resultante es:

$$M = 1,06 \times 10^{-2} \text{ Nm} \quad (47)$$

Al calcularse las tensiones nominales máximas en el empotramiento, se obtienen las siguientes tensiones máximas en el radio exterior y en el interior:

$$\sigma_{Rext} = 504\,000 \text{ Pa} \quad (48)$$

$$\sigma_{Rint} = 168\,000 \text{ Pa} \quad (49)$$

Para obtener los resultados más exactos posibles, se ha realizado el malla-
do más fino que nos ha permitido la simulación en la zona alrededor del
redondeo. Los mallados son:

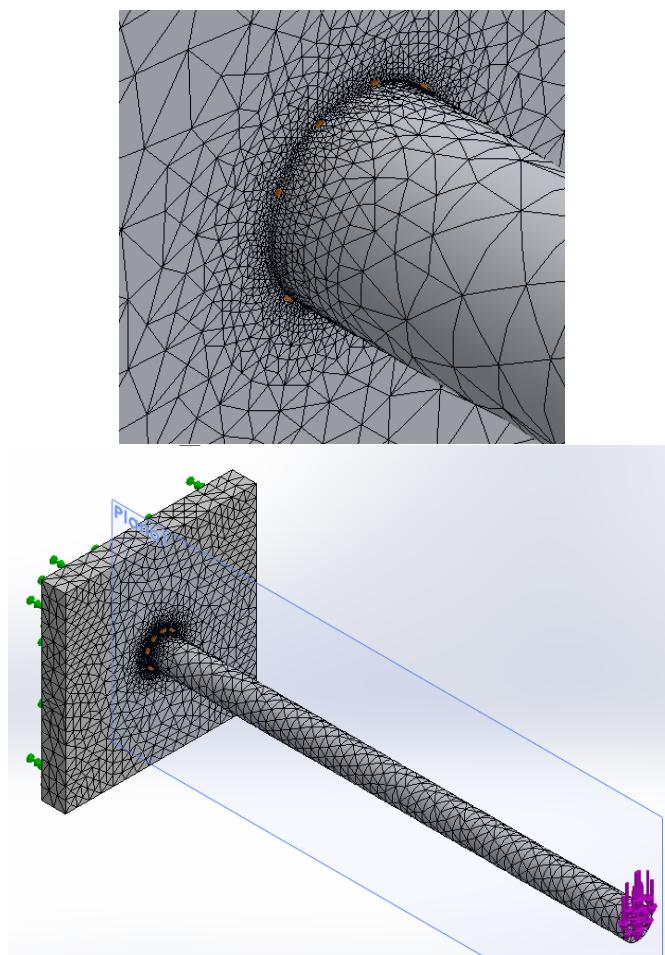


Figura 33: Mallado de la arista viva

El resultado de los desplazamientos se puede ver en la siguiente figura, que

es un poco mayor de lo definido, 2×10^{-4} quizás por la base, pero cumple bastante bien con las especificaciones de desplazamiento.

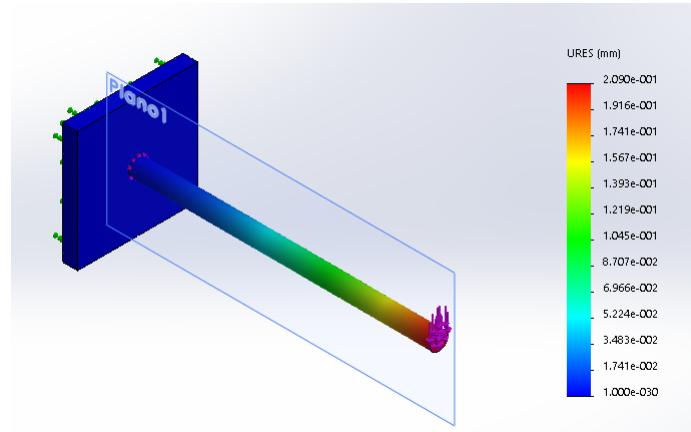


Figura 34: Desplazamientos en simulación de eslabón R3r1 con arista viva

En las siguientes imágenes se puede observar el estado tensional de la simulación, así como aparece marcado la máxima tensión en el radio exterior y en el interior.

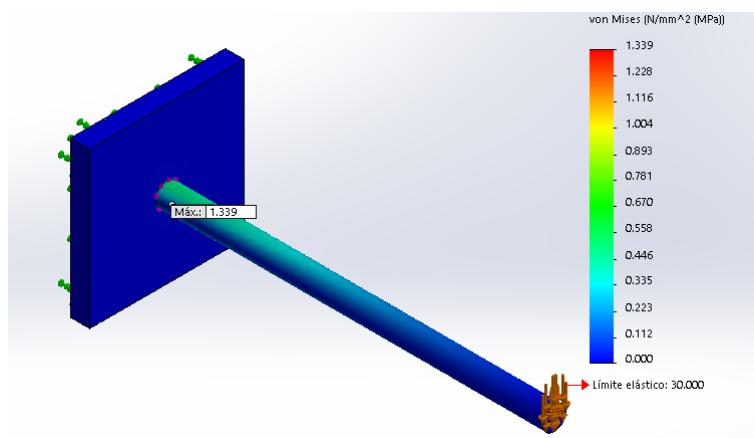


Figura 35: Estado tensional del eslabón con tensión max. en radio exterior

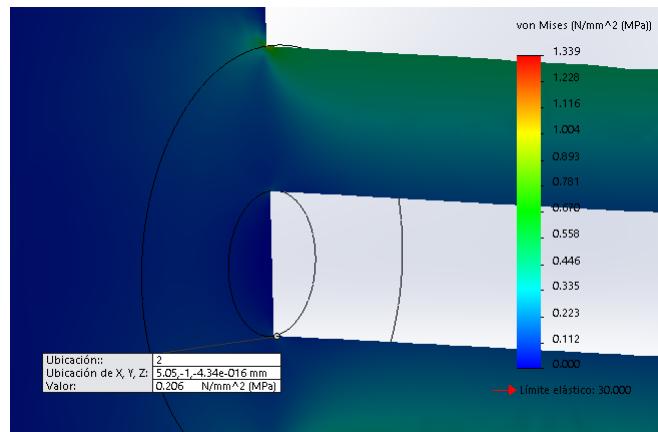


Figura 36: Tensión max. radio interior

En las figuras 12 y 13 se puede observar las tensiones máximas que son:

$$\sigma_R = 1\,135\,000 \text{ Pa} \quad (50)$$

$$\sigma_r = 206\,000 \text{ Pa} \quad (51)$$

Con estos resultados los concentradores de tensión quedaría en el radio exterior un $K_t = 2,25$ y en el radio interior $K_t = 1,23$. Para este caso, que se supone que el concentrador de tensiones sería el máximo, podemos interpretar que el programa puede tener fallo. Si se usase los datos de la figura 7, los concentradores de tensión serían de $K_t = 2,1$ pero no sería exacto ya que esta en la zona que esta haciendo asíntota con el eje Y, y solo hasta ahí se ha dibujado la línea y se debe suponer que seguiría incrementándose el valor del concentrador de tensiones, aunque es muy similar al hallado en la simulación. En el caso del radio interior, el concentrador usando la figura 7 sería un valor idéntico de $K_t = 2,2$, y en este caso vemos que no se asemeja al encontrado en la simulación.

2. Caso de viga más gruesa R=18 mm y r=15.5 mm con redondeo exterior de 2 mm e interior de 1 mm

En este caso se ha simulado el eslabón con el radio max fijado (3.2). Como se ha fijado el radio máximo de 18 mm, para minimizar el concentrador de tensiones y dar el redondeo mayor se debe de dar un redondeo exterior de 2

mm. En el interior se fija en 1 mm, por tener siempre una tensión menor. El resultado puede verse en la siguiente figura.

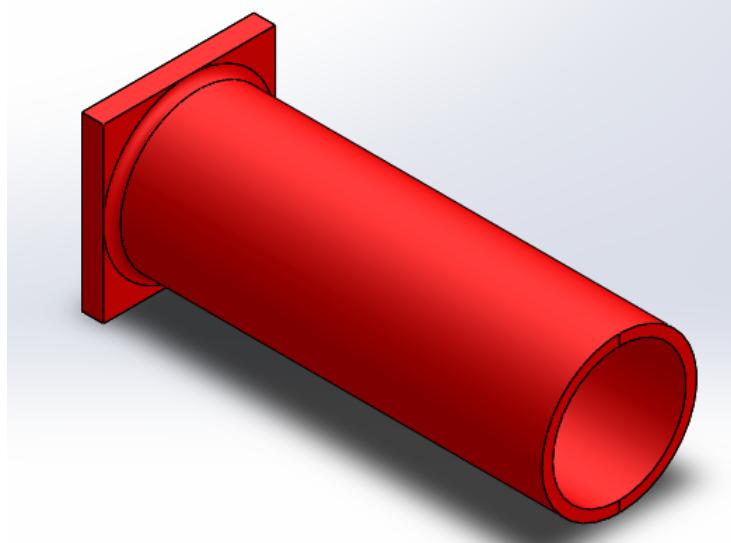


Figura 37: Eslabón de R18 y r15.5 con redondeos de 2 mm

La inercia del eslabón es:

$$I = 3,71 \times 10^{-4} \text{ m}^4 \quad (52)$$

Y la carga aplicada para que nos resulte optimizar con esta sección es:

$$P = 62,35N \quad (53)$$

Los desplazamientos, que tal como en la simulación anterior esta diseñada para que se produzca una flecha máxima de 2×10^{-4} , puede verse en la siguiente figura.

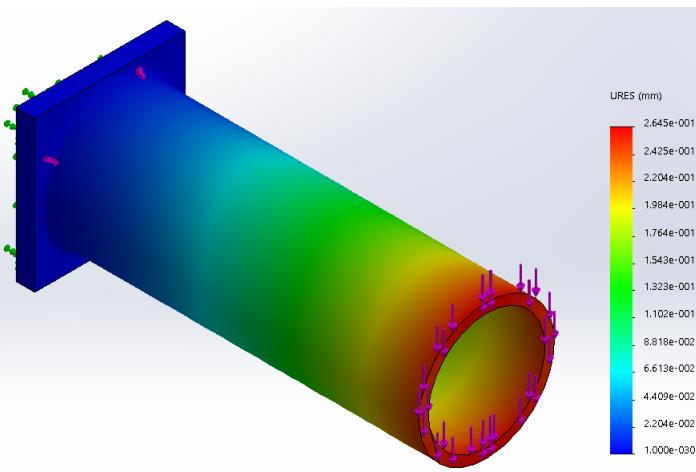


Figura 38: Desplazamientos en eslabón R18r15.5 con redondeos de 2 mm

Las tensiones nominales que se obtienen teóricamente son:

$$\sigma_R = 3\,024\,000 \text{ Pa} \quad (54)$$

$$\sigma_r = 1\,848\,000 \text{ Pa} \quad (55)$$

En las figuras siguientes se puede observar los resultados de la tensión en el radio exterior y en el radio interior.

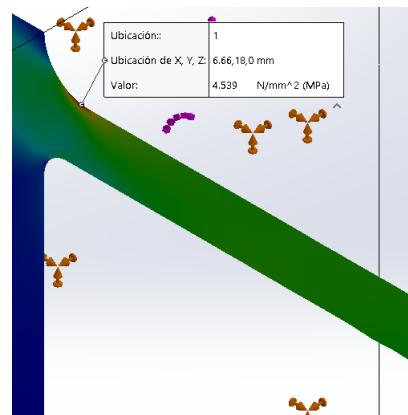


Figura 39: Desplazamientos en eslabón R18r15.5 con redondeos de 2 mm

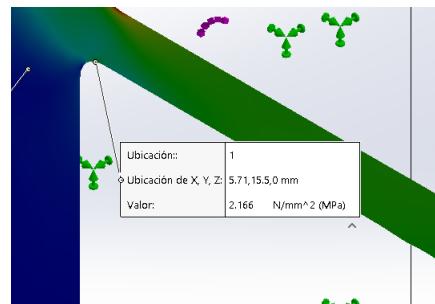


Figura 40: Desplazamientos en eslabón R18r15.5 con redondeos de 2 mm

Las tensiones máximas obtenidas en la simulación son las siguientes:

$$\sigma_R = 4\,539\,000 \text{ Pa} \quad (56)$$

$$\sigma_r = 2\,166\,000 \text{ Pa} \quad (57)$$

Los concentradores de tensiones en el radio exterior hallados en la simulación es en el radio exterior de $K_t = 1,5$ y en el radio interior de $K_t = 1,17$. Si se usan la figura 7 para obtener dichos concentradores de tensión se obtiene en el radio exterior un concentrador de $K_t = 1,7$. En el radio interior, usándose la figura 7, se obtiene un $K_t = 1,7$ aproximadamente.

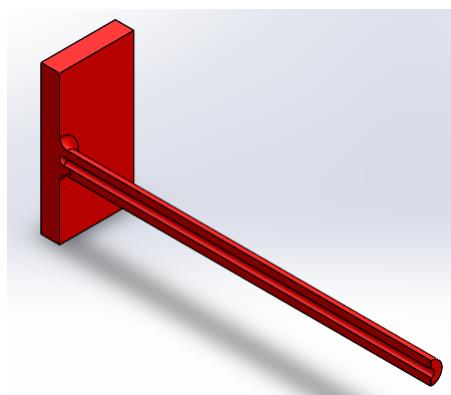


Figura 41: Desplazamientos en eslabón R18r15.5 con redondeos de 2 mm

3. Caso de viga más fina R=3 mm y r=1 mm con redondeo de 2 mm

Este caso, los redondeos se han seleccionado los mismos para el caso de la viga más gruesa, es decir, un redondeo exterior de 2 mm y un redondeo interior de 1 mm, lo que hace que en el interior sea semiesférico, como se puede ver en la figura 18. Aunque este caso, debido a que la base tiene más espacio y se podría dar mayores redondeos, minimizando el concentrador de tensión aun más, por los términos de los que depende según la figura 7, $\frac{D}{d}$, y $\frac{r}{d}$ el concentrador de tensiones va a dar valores más pequeños y no es necesario dar más.

Las solicitudes de la simulación son como el caso 1 aunque sin arista viva, por lo que la inercia (6.46), la carga (6.47) el momento en el empotramiento (6.48) y las tensiones nominales (6.49) y (6.50) son iguales.

Los resultados de desplazamiento de la simulación se ven en la siguiente figura.

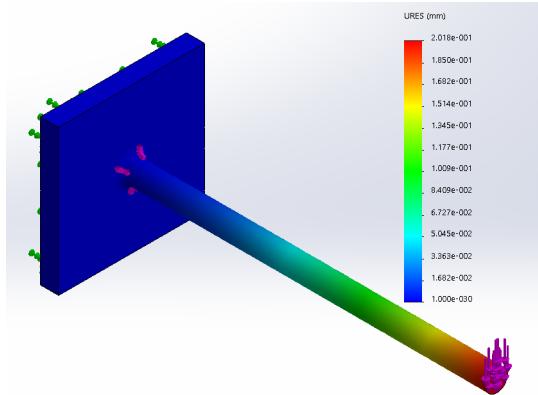


Figura 42: Desplazamientos en eslabón R3r1 con redondeos de 2 mm

Se puede observar que el desplazamiento esperado es un poco mayor, pero se puede considerar que cumple con lo esperado y que la simulación esta planteada correctamente.

Las tensiones que se obtienen de resultado de la simulación se puede ver en la figura 20 (radio exterior) y 21(radio interior).

$$\sigma_R = 598\,000 \text{ Pa} \quad (58)$$

$$\sigma_r = 147\,000 \text{ Pa} \quad (59)$$

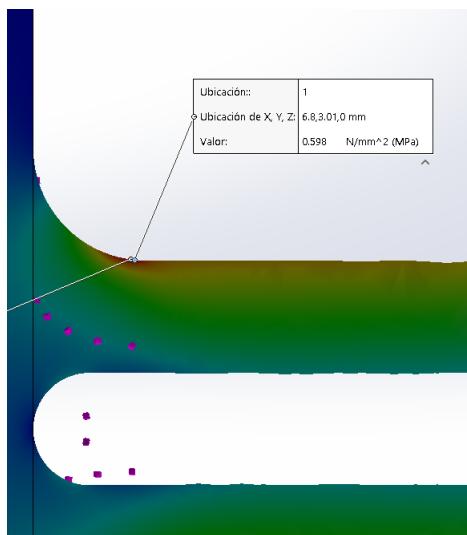


Figura 43: Tensión en el eslabón R3r1 con redondeo en el radio exterior

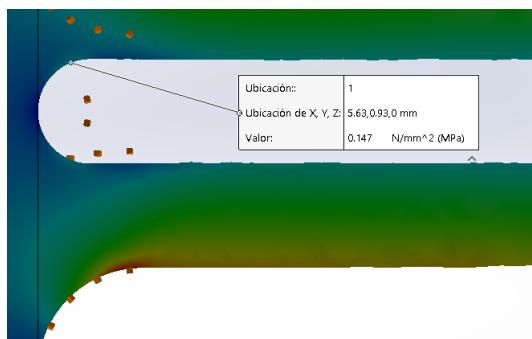


Figura 44: Tensión en el eslabón R3r1 con redondeo en el radio interior

Las tensiones máximas como resultado de la simulación son las siguientes:

Tras haberse comparado las tensiones de la simulación con las tensiones nominales teóricas (6.49)(6.50), se obtiene un concentrador de tensiones en el radio exterior de $K_t = 1,19$ y en el radio interior $K_t = 0,875$. Si se buscan los concentradores de tensiones con la figura 7, se obtiene un concentrador de tensiones en el radio exterior de $K_t = 1,3$ y en el radio interior $K_t = 1,3$. El resultado del concentrador de tensiones del radio interior que es menor que uno, y eso puede ser a que se debe tomar el menor diámetro como el diámetro de la sección y no el mayor y por eso es menor que 1.

Tras observar los resultados, se aprecia que en la tabla 7 se obtienen siempre mayores valores que en las simulaciones. Además, los mayores concentradores de tensiones se dan en la viga de mayor tamaño, dado que $\frac{r}{d}$ y $\frac{D}{d}$ llevan a los resultados más elevados. Como conclusión, se va a seleccionar como el valor fijo de concentrador de tensiones más alto para todos los casos, para poder asegurar la ausencia de fallo en la actuación de los eslabones.

Referencias

- [1] Actionlib ROS package. <http://wiki.ros.org/actionlib>. Accessed: 2016-09-1.
- [2] BitBloq. <http://bitbloq.bq.com/>. Último acceso: 2015-12-23.
- [3] Dash robot. <https://www.makewonder.com/>. Último acceso: 14-12-2016.
- [4] Diff_drive_controller ROS package. http://wiki.ros.org/diff_drive_controller. Accessed: 2016-09-1.
- [5] MoveIt! <http://moveit.ros.org/>. Último acceso: 2015-12-23.
- [6] OpenSCAD, the programmers solid 3D CAD modeller. <http://www.openscad.org/>. Último acceso: 2016-12-23.
- [7] pySUMO. <https://github.com/pySUMO/pysumo>. Último acceso: 2016-12-23.
- [8] Robot Operating System. <http://www.ros.org/>. Último acceso: 2015-12-23.
- [9] RViz ROS package. <http://wiki.ros.org/rviz>. Accessed: 2016-09-1.
- [10] Schunk modular robotic system. <http://mobile.schunk-microsite.com/en/produkte/products/dextrous-lightweight-arm-lwa-4d.html>. Accessed: 2016-09-1.
- [11] KDL ROS package. <http://wiki.ros.org/kdl>. Accessed: 2016-09-1.
- [12] Xacro ROS package. <http://wiki.ros.org/xacro>. Accessed: 2016-09-1.
- [13] IEEE standard ontologies for robotics and automation, 2015.
- [14] D. Beckett and B. McBride. Rdf/xml syntax specification (revised). *W3C recommendation*, 10, 2004.
- [15] J. Domingo Santillana. *Flexión. Tensiones*, chapter 5. E.P.S. Zamora, <http://ocw.usal.es/ensenanzas-tecnicas/resistencia-de-materiales-ingenero-tecnico-en-obras-publicas/contenidos/2008>.

- [16] S. Doncieux, J.-B. Mouret, N. Bredeche, and V. Padois. *Evolutionary Robotics: Exploring New Horizons*, pages 3–25. Springer Berlin Heidelberg, 2011.
- [17] C. Fellbaum. *WordNet*. Wiley Online Library, 1998.
- [18] K. Fu, R. González, and C. Lee. *Robotics: Control, Sensing, Vision and Intelligence*. McGraw Hill, 1987.
- [19] M. R. Genesereth, R. E. Fikes, et al. Knowledge interchange format-version 3.0: reference manual. 1992.
- [20] A. Golovinsky, M. Yim, Y. Zhang, C. Eldershaw, and D. Duff. Polybot and polykinetic system: a modular robotic platform for education. In *Proceedings 2004 IEEE International Conference on Robotics and Automation*, 2004.
- [21] J. Gonzalez-Gomez, H. Zhang, E. Boemo, and J. Zhang. Locomotion capabilities of a modular robot with eight pitch-yaw-connecting modules. In *9th international conference on climbing and walking robots*, 2006.
- [22] J. González Gómez. *Robótica Modular y Locomoción: Aplicación a Robots Ápodos*. PhD thesis, Universidad Autónoma de Madrid, 2008.
- [23] N. Guarino, D. Oberle, and S. Staab. *What Is an Ontology?*, pages 1–17. Springer Berlin Heidelberg, 2009.
- [24] I. Ha, Y. Tamure, and H. Asama. Development of open platform humanoid robot darwin-op. *Advanced Robotics*, 27, 2013.
- [25] H. Li, H. Wei, J. Xiao, and T. Wang. Co-evolution framework of swarm self-assembly robots. *Neurocomputing*, 148:112–121, 2014.
- [26] H. Li, H. Wei, J. Xiao, and T. Wang. Co-evolution framework of swarm self-assembly robots. *Neurocomputing*, 148:112–121, 2015.
- [27] A. Marchetti, F. Ronzano, M. Tesconi, and M. Minutoli. Formalizing knowledge by ontologies: Owl and kif. *Relatório apresentado L’Istituto di Informatica e Telematica (IIT). Consiglio Nazionale delle Ricerche (CNR). Italia*, 2008.
- [28] D. L. McGuinness, F. Van Harmelen, et al. Owl web ontology language overview. *W3C recommendation*, 10(10):2004, 2004.

- [29] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji. M-TRAN: self-reconfigurable modular robotic system. *IEEE/ASME Transactions on Mechatronics*, 7(4):431–441, dec 2002.
- [30] I. Niles and A. Pease. Towards a standard upper ontology. In *Proceedings of the international conference on Formal Ontology in Information Systems*, pages 2–9. ACM, 2001.
- [31] I. Niles and A. Pease. Towards a standard upper ontology. In *Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001*, pages 2–9. ACM, 2001.
- [32] A. Pease. The sigma ontology development environment. In *Working Notes of the IJCAI-2003 Workshop on Ontology and Distributed Systems*, volume 71, 2003.
- [33] A. Pease. Standard upper ontology knowledge interchange format. *Unpublished language manual. Available at <http://sigmakee.sourceforge.net>*, 2004.
- [34] P. Pedregal. *Introduction to Optimization*. Springer-Verlag New York, 2004.
- [35] W. Pilkey. *Peterson's Stress Concentration Factors*. John Wiley and Sons, 1999.
- [36] J. B. Pollack, H. Lipson, G. Hornby, and P. Funes. Three generations of automatically designed robots. *Artificial Life*, 7(3):215–223, Summer 2001.
- [37] R. Siegwart and I. R. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. Bradford Company, Scituate, MA, USA, 2004.
- [38] F. Silva, M. Duarte, L. Correia, S. M. Oliveira, and A. L. Christensen. Open issues in evolutionary robotics. *Evolutionary Computation*, 24:205–236, 2016.
- [39] A. Spröwitz, R. Moeckel, M. Vespignani, S. Bonardi, and A. Ijspeert. Room-bots: A hardware perspective on 3d self-reconfiguration and locomotion with a homogeneous modular robot. *Robotics and Autonomous Systems*, 62(7):1016–1033, jul 2014.

- [40] M. Tenorth and M. Beetz. KnowRob – A Knowledge Processing Infrastructure for Cognition-enabled Robots. Part 1: The KnowRob System. *International Journal of Robotics Research (IJRR)*, 32(5):566 – 590, April 2013.
- [41] H. Wei, Y. Cai, H. Li, D. Li, and T. Wang. Sambot: A self-assembly modular robot for swarm robot. In *2010 IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers (IEEE), may 2010.