UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA MECÁNICA

Modelado y simulación dinámica del robot industrial Staübli Unimation PUMA 560

Presentado por:

Marcos Simó Villanueva

Dirigido por:

Josep Lluís Suñer Martínez

Paloma Vila Tortosa

VALENCIA, JULIO 2018

Agradecimientos

Quisiera agradecer en primer lugar, el gran apoyo recibido por parte de mi familia durante todos estos cuatro años de carrera, el cual siempre me ha dado muchas fuerzas y energías para seguir adelante y conseguir todos mis logros.

Por otro lado, también me gustaría agradecer en gran medida el trabajo de apoyo y toda la ayuda que me ha brindado mi tutor Josep Lluís Suñer Martínez, sin el cual este trabajo no hubiera sido posible.

Por todo ello, muchas gracias

Marcos Simó Villanueva

Resumen del proyecto

El objetivo de este proyecto es el estudio del modelo cinemático directo e inverso, así como el dinámico inverso de un Robot PUMA 560 de seis grados de libertad.

Este estudio se basará principalmente en la comparación del modelo realizado mediante el software matemático MATLAB con la simulación realizada en el software ADAMS View.

Para comenzar, se han descargado los archivos CAD y se han importado al software de simulación ADAMS View, donde se ha ensamblado colocando los diferentes pares de revolución y *motions* entre los eslabones del brazo robótico PUMA 560. También se han establecido los parámetros másicos e inerciales relativos a cada barra del robot, siguiendo el criterio de PAUL 81.

Una vez con el robot correctamente definido en ADAMS View, se ha procedido a modelizar matemáticamente la cinemática y dinámica en MATLAB, con el objetivo de poder manipular de una forma sencilla las posiciones angulares de los servos para poder comparar los resultados con aquellos obtenidos de la simulación en ADAMS View.

Índice

1 Introducción

1.1 Estructura del proyecto

La estructura de este proyecto se divide en 10 capítulos que se enumeran y explican a continuación:

- El capítulo número uno es una introducción al proyecto, así como una breve explicación del robot seleccionado para el estudio.

- En el segundo capítulo se desarrolla una explicación de todos los elementos y herramientas matemáticas utilizados para realizar el estudio tanto de la cinemática como de la dinámica del brazo robótico.

- El tercer capítulo trata el estudio cinemático de posición, tanto directo como inverso.

- El cuarto capítulo también está destinado a la cinemática, pero en este caso se trata de la cinemática de movimiento.

- El quinto capitulo está enfocado a el cálculo de las trayectorias en el espacio de los nudos del robot.

- El sexto capítulo trata la dinámica de robots, en concreto el método para la resolución de la Dinámica Inversa del PUMA 560.

- El séptimo capítulo trata del desarrollo y resolución del proyecto, tanto procedimiento como soluciones obtenidas, así como los programas realizados para el cálculo y la simulación.

- El octavo capítulo muestra las conclusiones y futuros desarrollos obtenidos del proyecto llevado a cabo.

- En el noveno capítulo se expondrá el estudio económico del proyecto, considerando todos los aspectos necesarios para la realización de un proyecto técnico.

- En el décimo capítulo se incluye la bibliografía e información consultada, detallada de manera ordenada.

1.2 Objetivos

Este trabajo de fin de grado tiene como finalidad la modelización matemática de la cinemática directa e inversa, así como de la dinámica inversa para un robot PUMA 560, para compararlo posteriormente con la simulación dinámica de este y analizar los resultados.

Las diferentes partes del cuerpo del proyecto son las siguientes:

- Resolución del problema cinemático directo e inverso, de manera que se obtenga toda la información referente a la posición, velocidad y aceleraciones, del efector final.

- Modelado en MATLAB mediante la realización de programas en los cuales se implementarán los datos de los nudos, de modo que, como respuesta, se obtenga el estudio cinemático y dinámico del robot.

- Simulación en ADAMS View del robot. Previamente se definirán aspectos como las posiciones de los *Marker*s, ecuaciones de movimiento o medidas, de modo que obtengamos una simulación lo más parecida a la realidad y de gran precisión.

- Comparación y verificación y análisis de resultados. Una vez se tengan los resultados de MATLAB y de la simulación en ADAMS View, se procederá a el análisis y comparación de dichos resultados, llegando con ello a conclusiones.

1.3 Punto de partida

Primeramente, se ha realizado un pequeño estudio sobre las posibilidades de elección de diferentes tipos de robots industriales. Finalmente se llegó a la conclusión de que el robot escogido sería el PUMA 560, un robot clásico que salió a la luz por primera vez en los ochenta.

El motivo principal de la elección de dicho robot reside en que, debido a su largo tiempo en el mundo de la industria y la investigación, se pueden encontrar numerosas bases de datos, artículos y documentación de interés que permitirá realizar sin problema el estudio y modelización completa del mismo.

Por otra parte, el propio departamento de Ingeniería Mecánica y de Materiales es propietario de un prototipo, por lo que gracias a ello, sería posible la medición o la visualización de una forma más sencilla de su morfología y estructura.

Todos estos factores y la motivación por modelizar dicho robot con programas informáticos que hoy en día están a la vanguardia en el mundo de la ingeniería, son los que han decantado la decisión de escoger como robot para el caso de estudio el PUMA 560.

Una vez escogido el brazo robótico que se desea analizar, se debe establecer un punto de partida y una secuencia a seguir para poder realizar dicho proyecto.

1. Recolección de datos: Lo primero que se debe hacer antes de comenzar con el desarrollo del proyecto es recabar toda la información y características de nuestro robot, para ello se utilizará los datos que se pueden obtener de la literatura tanto en internet como de apuntes. De ella se obtendrán datos importantes como las dimensiones, posiciones de los servos, sistemas de referencia...

2. Parametrización DH: Una vez se haya recopilado la información necesaria, con papel y lápiz, se comenzará el estudio de parametrización mediante el método de Denavit & Hartenberg. Una vez se tenga este punto claro, se podrá comenzar a ensamblar el robot y colocar los distintos *motions* y *marker*s de manera correcta en el software de simulación.

3. Ensamblado del robot y correcta caracterización: Una vez se tenga completamente claro el punto anterior, se podrá proceder al ensamblado del robot, además se tendrán que colocar *marker*s conforme a DH, colocar los servos en los puntos indicados por el fabricante y establecer las barras según los parámetros inerciales y de posición deseados.

4. **Código MATLAB**: Con los pasos anteriores realizados, se generará un código en MATLAB que realice tanto el estudio de posición como el estudio diferencial y dinámico, de modo que se puedan variar los datos de posiciones, velocidades y aceleraciones en los nudos de una manera fluida para poder obtener los datos de distintas configuraciones de una manera fácil y sencilla.

Como cabe esperar, los puntos indicados anteriormente son sólo los hitos principales del proyecto, hay mucho trabajo de fondo, de recopilación de bibliografía, comprensión de las herramientas matemáticas y físicas. También hay puntos obviados como la selección de parametrización, generación de trayectorias etc.

1.4 Breve introducción al PUMA 560

El nombre del brazo robótico PUMA representa las siglas de “Programmable Universal Machine for Assembly, or Programmable Universal Manipulation Arm”. Este es un brazo robótico industrial creado por Victor Scheinman en la compañía robótica Unimation. Inicialmente desarrollado para General Motors, el PUMA estaba basado en diseños anteriores de Scheinman realizados en la Universidad de Standford.

Unimation fabricó dicho robot durante años hasta que finalmente fue comprada por la empresa suiza Stäubli en 1988. Nokia Robotics fabricó en torno a 1500 robots PUMA durante la década de los 80, llegando a ser el modelo mas popular entre los clientes de la época.

En 2002, la organización General Motors Controls, Robotics and Welding (CRW), donó el prototipo original del PUMA al Museo Nacional de Historia Americana.

El prototipo PUMA 560, consistía en un robot de seis grados de libertad con muñeca esférica, con una capacidad de carga máxima de entre 2.5 Kg a 4 Kg, con un peso global aproximado de 83 Kg, una repetitividad de +-0.1mm y unas velocidades máximas de carga de 500mm/segundo en línea recta con una carga de 2.5 Kg y de 470mm/segundo en línea recta con una carga de 4Kg.

El PUMA 560 es la rata de laboratorio de la investigación robótica, ha sido estudiado y usado en innumerables experimentos e industrias. Sin embargo, sigue siendo un reto reunir completamente todos los datos necesarios para construir debidamente un modelo, ya que, al haber sido frecuentemente estudiado, cada uno de los estudios ha extraído sus propios datos y conclusiones. Es por ello por lo que, en el desarrollo de este proyecto, se ha decidido elegir la solución de PAUL 81 debido a que es el estudio que se ha determinado más próximo a lo que podría ser la realidad.

2 Herramientas y métodos de localización espacial

2. 1 Introducción a la posición y orientación de un sólido rígido en el espacio

Un robot manipulador, es una cadena cinemática abierta, con varias barras, conectadas en serie por pares cinemáticos o nudos, los cuales están accionados por actuadores. Estos pares pueden ser de tipo revolución (pares R), o prismáticos (pares P). En alguno de los extremos de dicha cadena cinemática habrá una unión con la barra fija que constituirá la llamada base o soporte del robot, mientras que, en el extremo opuesto, podrá portar la herramienta o efector final encargada de desempeñar la función asignada al robot. Esta morfología genera que, gracias al desplazamiento relativo entre los distintos nudos que forman el robot, el efector final adopte una determinada posición y orientación en el espacio. En la robótica, es esencial la determinación de la posición y orientación de dicho efector final respecto a una referencia fija.

A lo largo de las sucesivas hipótesis, se va a considerar que las barras son rígidas y los pares cinemáticos que forman parte de la cadena abierta, no son flexibles. Esta hipótesis facilitará la posterior modelización del robot.

Los primeros pasos para analizar y modelizar un robot manipulador de cualquier tipo consisten en obtener matemáticamente una representación de la posición y la orientación en el espacio de su elemento terminal o efector final. El procedimiento que se va a seguir para hallar esta localización consiste en asignar un sistema de referencia cartesiano, trirrectangular y dextrógiro a cada uno de los objetos a representar, refiriendo su posición y orientación a un sistema de referencia fijo, previamente establecido.

2.2 Matrices de rotación

La forma más utilizada de expresar la orientación de un sólido rígido es representando mediante una matriz de rotación R de dimensión 3x3, los cosenos directores de los vectores unitarios del sistema de referencia local asignado a dicho sólido anteriormente respecto al sistema de referencia global que se haya establecido.

Dicha matriz de rotación cumple una serie de condiciones las cuales resultan de un alto interés y utilidad a la hora de efectuar cálculos. Las principales condiciones son que todas las columnas de dicha matriz cumplan la condición de ortogonalidad dos a dos y además tengan módulo unidad, por lo tanto, se demuestra que la inversa de la matriz de rotación es igual a su traspuesta, ($R^{-1} = R^{T}$).

Debe de tenerse en cuenta que la rotación de matrices no posee la propiedad conmutativa, por lo que el orden en el que se efectúe el producto de las matrices influirá en el resultado final obtenido, debido a que una vez que se ha efectuado una rotación, los ejes de los sistemas de referencia (inicial y final), ya no será coincidentes. Resulta por ello necesario establecer convenciones a la hora de efectuar rotaciones consecutivas.

2.3 Composición de rotaciones

Se denomina composición de rotaciones cuando un conjunto de matrices se multiplica en un orden determinado, representando como resultado, una secuencia de rotaciones alrededor de ciertas direcciones. Esta operación permite establecer la orientación de cualquier solido rígido en el espacio.

El conjunto de tres ángulos en el espacio ϕ, θ y Ψ, determina la orientación del sistema de referencia final respecto al inicial, estos ángulos se denominan ángulos de Euler.

Los ángulos de Euler, mostrados en la Figura 1, son independientes entre sí, ya que representan giros respecto a ejes mutuamente ortogonales. Estos ángulos, son la forma más eficaz y sencilla de representar la orientación de un sólido rígido en el espacio y pueden tomar diversas configuraciones:

-Ángulos de Euler ZXZ

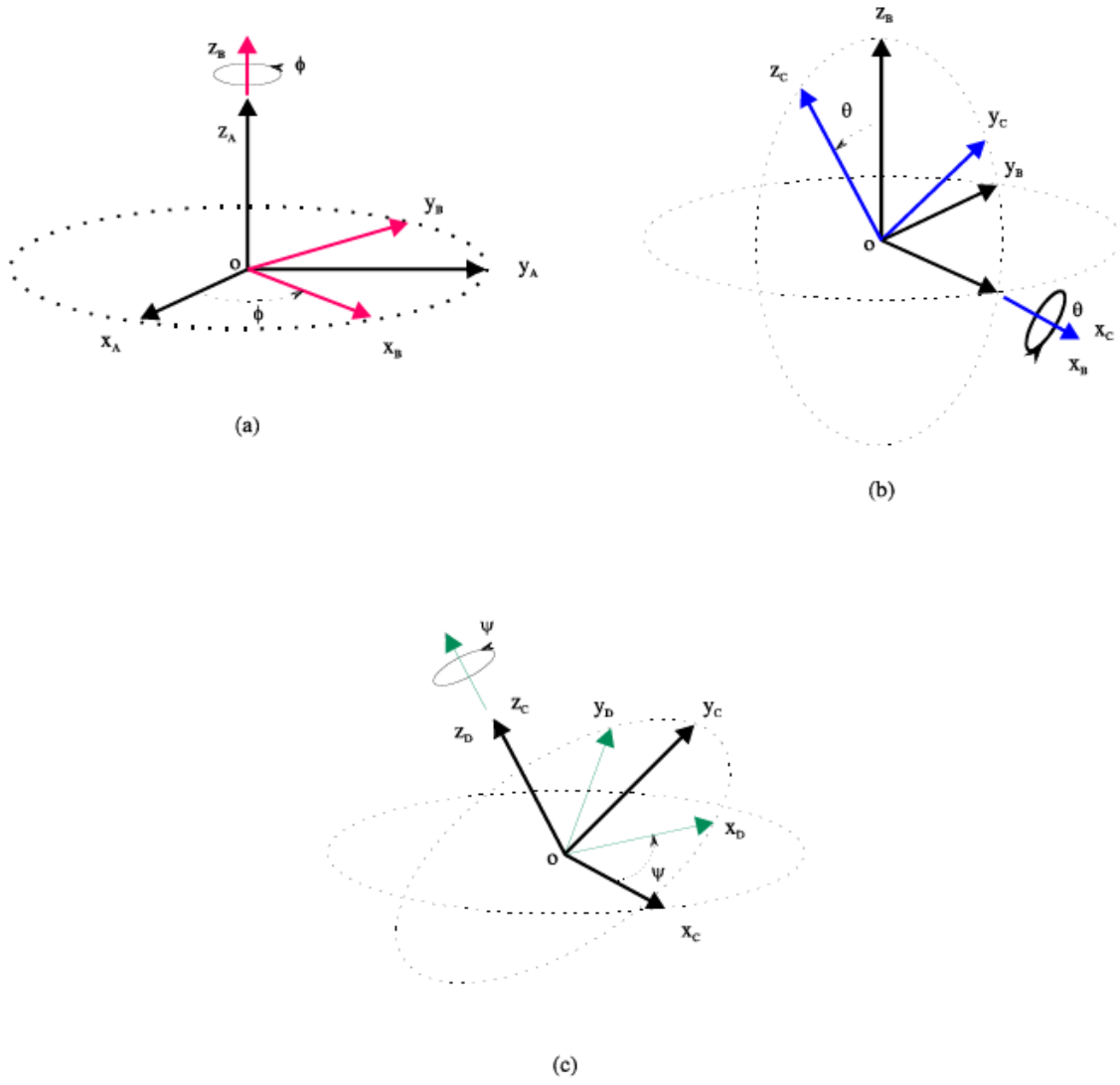


Figura 1: Ángulos de Euler ZXZ

Las tres matrices de rotación que representan los giros o transformaciones consecutivas son las siguientes:

-Primer giro en Z:

$$R_Z(\phi)=\begin{bmatrix}\cos(\phi) & -\text{sen}(\phi) & 0\\ \text{sen}(\phi) & \cos(\phi) & 0\\ 0 & 0 & 1\end{bmatrix} \quad (1)$$

-Primer giro en X:

$$R_X(\theta)=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos(\theta) & -\text{sen}(\theta)\\ 0 & \text{sen}(\theta) & \cos(\theta)\end{bmatrix} \quad (2)$$

-Segundo giro en Z:

$$R_Z(\psi)=\begin{bmatrix}\cos(\psi) & -\text{sen}(\psi) & 0\\ \text{sen}(\psi) & \cos(\psi) & 0\\ 0 & 0 & 1\end{bmatrix} \quad (3)$$

Con la combinación de las tres rotaciones se obtiene:

$$R(\phi,\theta,\psi)=R_Z(\phi)\cdot R_X(\theta)\cdot R_Z(\psi)=$$

$$=\begin{bmatrix}C\phi C\psi - S\phi C\theta S\psi & -C\phi S\psi - S\phi C\theta C\psi & S\phi S\theta\\ S\phi C\psi + C\phi C\theta S\psi & -S\phi S\psi + C\phi C\theta C\psi & -C\phi S\theta\\ S\theta S\psi & S\theta C\psi & C\theta\end{bmatrix} \quad (4)$$

Esta combinación de los ángulos de Euler es la más empleada, y la que se usará más adelante durante el desarrollo del trabajo.

-Ángulos de Euler ZYZ

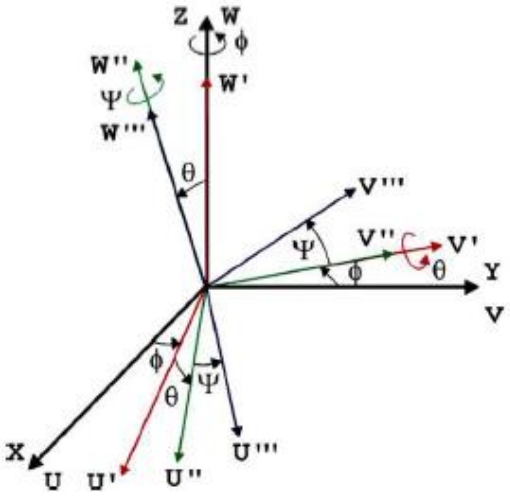


Figura 2: Ángulos de Euler ZYZ

Combinando las matrices anteriores esta vez cambiando el giro intermedio por un giro en Y se tiene:

$$R(\phi,\theta,\psi) = R_Z(\phi)\cdot R_Y(\theta)\cdot R_Z(\psi) =$$

$$= \begin{bmatrix} C\phi C\psi - S\phi C\theta S\psi & -C\phi S\psi - S\phi C\theta C\psi & S\phi S\theta \\ S\phi C\psi + C\phi C\theta S\psi & -S\phi S\psi + C\phi C\theta C\psi & -C\phi S\theta \\ S\theta S\psi & S\theta C\psi & C\theta \end{bmatrix} \quad (6)$$

-Roll, Pitch & Yaw (alabeo, cabeceo y guiñada)

Es otro conjunto de ángulos de Euler empleado sobre todo en el campo de la aeronáutica útil para describir la posición y orientación de un sólido rígido en el espacio. El giro en este caso corresponde a una rotación alrededor de eje Z, el cabeceo corresponde a una rotación alrededor del nuevo eje Y, por último, la guiñada corresponde al giro alrededor de un nuevo eje X.

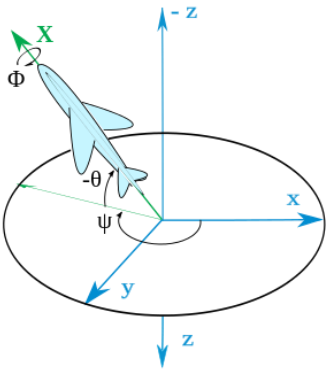


Figura 3: Alabeo, cabeceo y guiñada

Combinando las matrices de rotación se tiene:

$$R(\phi,\theta,\psi) = R_Z(\phi)\cdot R_Y(\theta)\cdot R_Z(\psi) =$$

$$\begin{bmatrix} \cos(\phi)\cos(\theta) & \cos(\phi)\sin(\psi)\sin(\theta) - \sin(\phi)\cos(\psi) & \cos(\phi)\cos(\psi)\sin(\theta) + \sin(\phi)\sin(\psi) \\ \sin(\phi)\cos(\theta) & \sin(\phi)\sin(\psi)\sin(\theta) + \cos(\phi)\cos(\psi) & \sin(\phi)\cos(\psi)\sin(\theta) - \cos(\phi)\sin(\psi) \\ -\sin(\theta) & \sin(\psi)\cos(\theta) & \cos(\psi)\cos(\theta) \end{bmatrix} \quad (7)$$

2.4 Transformación homogénea:

Para expresar la posición relativa de cualquier parte del robot, así como su orientación en el espacio, a parte de la rotación, la cual nos caracterizará la orientación, se debe emplear otro tipo más de transformación, la traslación.

Si se desea expresar la posición relativa de un sistema de referencia $\{O_B{-}X_BY_BZ_B\}$, respecto a otro $\{O_A{-}X_AY_AZ_A\}$, se debe caracterizar tanto la posición como la orientación. Se deberá considerar un vector de posición ${}^{A}r_{B}$ del origen O_B respecto al primer sistema de referencia.

Un vector cualquiera ${}^{B}r$ se puede transformar en otro ${}^{A}r$, mediante la expresión ${}^{A}r = {}^{A}R_B \cdot {}^{B}r + {}^{A}r_{B}$ donde ${}^{A}R_B$ representa la orientación del sistema de referencia $\{O_B{-}X_BY_BZ_B\}$ respecto al sistema $\{O_A{-}X_AY_AZ_A\}$.

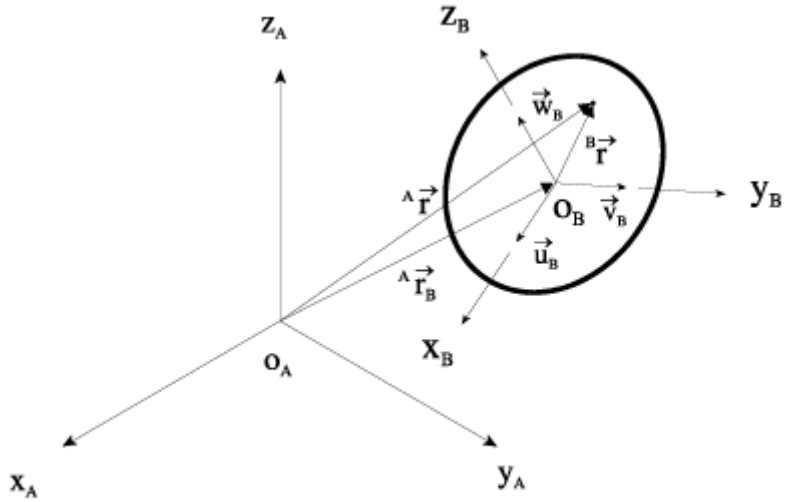


Figura 4: Transformación de coordenadas

Toda esta teoría, puede expresarse de forma compacta mediante la llamada Matriz de Transformación Homogénea (A, de dimensiones 4x4):

$$ {}^{A}A_{B} = \begin{bmatrix} & {}^{A}R_{B} & & \vdots & {}^{A}r_{B} \\ \cdots & \cdots & \cdots & \vdots & \cdots \\ 0 & 0 & 0 & \vdots & 1 \end{bmatrix} $$

O expresada de forma reducida:

$$ \mathbf{T} = \begin{bmatrix} \mathbf{R}_{3\times3} & \mathbf{p}_{3\times1} \\ \mathbf{f}_{1\times3} & \mathbf{w}_{1\times1} \end{bmatrix} = \begin{bmatrix} \text{Rotación} & \text{Traslación} \\ \text{Perspectiva} & \text{Escalado} \end{bmatrix} $$

Esta matriz permite cambiar el sistema de referencia respecto al cual se expresa la posición de un punto en el espacio, así como describir la relación entre dos sistemas de referencia en el espacio, mediante rotaciones y translaciones. Es importante recordar que como la multiplicación de matrices no sigue la propiedad conmutativa, el orden de las sucesivas rotaciones y traslaciones es importante.

$$ {}^{0}r_{e} = {}^{0}A_{1} \cdot {}^{1}A_{2} \cdot \ldots {}^{n-1}A_{n} \cdot {}^{n}r_{e} \qquad (8) $$

2.5 Modelización cinemática del brazo robot

Cadenas cinemáticas abiertas

Las herramientas de cálculo mencionadas en los apartados anteriores se pueden aplicar al modelo de un brazo robótico manipulador, el cual está formado por una serie de eslabones, solidos rígidos, los cuales están unidos mediante articulaciones, formando una cadena cinemática abierta. Usando el concepto de transformación homogénea, será posible situar respecto a un sistema de coordenadas fijo, tanto la posición como la orientación de cada una de las barras que componen dicho robot.

El movimiento relativo entre dos eslabones se consigue mediante el movimiento del par o articulación que los conecta. Un robot con n grados de libertad, estará constituido por n+1 barras (incluyendo la barra fija) unidas por n pares cinemáticos o nudos. Dichas barras se numerarán en orden ascendente, comenzando por la barra fija a la que se le asignará el número 0. Con lo que tenemos que el nudo i-ésimo conectará a las barras (i-1) e i.

Con la finalidad de poder expresar la posición de cada barra del robot con respecto de su anterior, se asignará a cada una de ellas un sistema de referencia local. Concatenando las diferentes transformaciones de cada barra de manera secuencial, se podrá expresar la posición del efector final respecto de la base fija del robot.

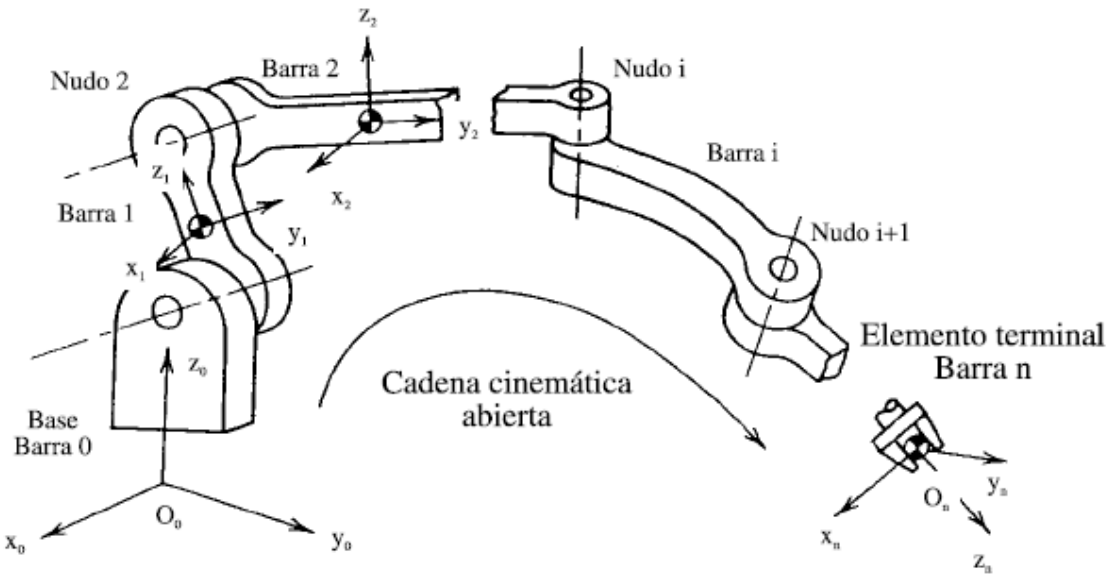


Figura 5: Cadena cinemática abierta

2.6 Notación de Denavit y Hartenberg

La notación de Denavit y Hartenberg, proporciona un método sistemático para asignar un sistema de referencia cartesiano trirrectangular y dextrógiro a cada barra que forma parte de brazo robot. Gracias a ello se podrán describir las relaciones cinemáticas entre elementos del manipulador mediante una serie de transformaciones homogéneas como ya se ha expresado anteriormente. Dicho procedimiento hace uso de un número mínimo de parámetros para definir completamente las relaciones cinemáticas, así mismo evita indefiniciones en estos casos de cadenas cinemáticas abiertas.

A continuación, se procederá a explicar cómo llevar a cabo la parametrización de Denavit y Hartenberg según el método de Paul.

Si se toman dos barras adyacentes (i-1) e i, unidas por el par cinemático i, el cual es de revolución, se tiene que el sistema de referencia i-1 está situado sobre el nudo i, mientras que el sistema de referencia de la barra i, estará colocado sobre la articulación i+1.

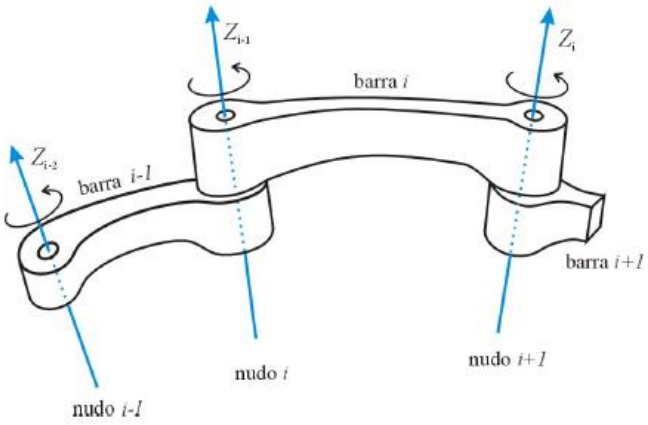


Figura 6: Notación DH criterio de Paul para un par de revolución

Por otro lado, los ejes Z de los sistemas de referencia coincidirán con los ejes característicos de los nudos sobre los que están situados, siendo los ejes de giro en el caso de un par de revolución, y el eje de deslizamiento en nudos prismáticos. Como consecuencia, tendremos que el eje Z_{i-1} corresponderá al eje de giro del nudo i, mientras que el origen del sistema de referencia i estará localizado en la intersección del eje Z_i con la normal común entre los ejes Z_{i-1} y Z_i la cual será nombrada como H_iO_i.

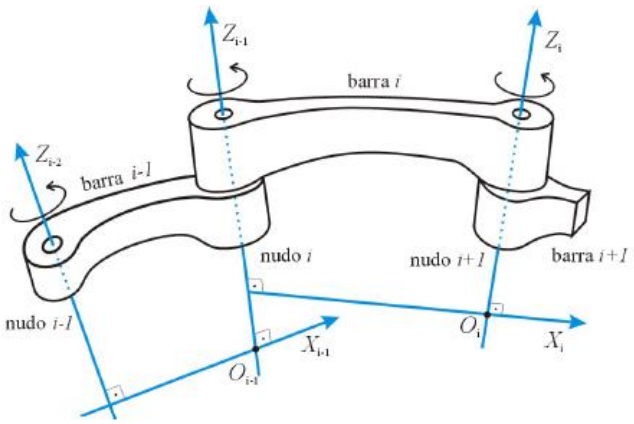


Figura 7: Notación DH eje X

Como se puede observar en la Figura anterior, el eje X coincidirá con la normal común entre los ejes Z situados en la misma barra.

Por último, será necesario definir el eje Y, de manera que se forme un sistema de referencia cartesiano, trirrectangular a derechas.

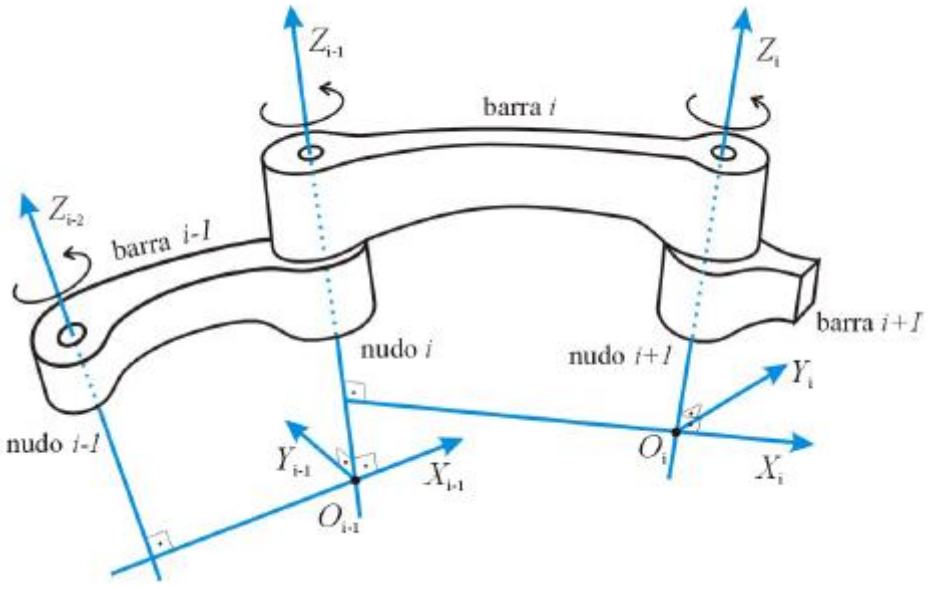


Figura 8: Notación DH eje Y

En el caso de que los ejes Z_{i-1} y Z_i sean paralelos o coincidentes, se usará cualquier recta que sea normal a ambos y en el caso de que dichos ejes se intersecten, el X_i será la recta que pase por el punto de intersección y sea perpendicular al plano que forman los ejes Z.

Para definir y localizar correctamente de dichos sistemas de referencia, serán necesarios 4 parámetros que nos establecerán la posición y orientación relativa entre dos barras adyacentes dichos parámetros son: [a_i, d_i, α_i, θ_i] de los cuales solo uno de ellos será variable. A dicho parámetro se le llamará variable de nudo.

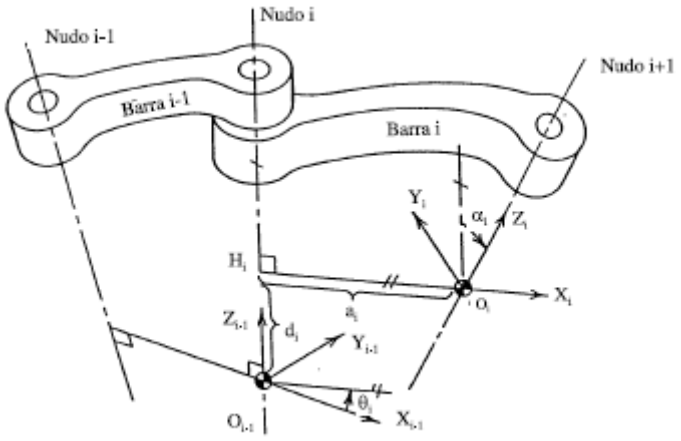


Figura 9: Parámetros DH

Cada parámetro indica lo siguiente:

a_i → Longitud de la normal común entre H_iO_i.
d_i → Distancia entre el origen O_{i-1} y el punto H_i.
α_i → Ángulo entre los ejes Z_{i-1} y Z_i, debe ser medido alrededor de X_i en sentido positivo.
θ_i → Ángulo comprendido entre X_{i-1} y la normal común H_iO_i medido en sentido positivo. alrededor de Z_{i-1} .

Los parámetros ai y αi, son constantes y vienen determinados por la geometría de la barra i. Los parámetros que podrán variar según el movimiento y el tipo de par del que se trate son di o θi. En pares de revolución será este último parámetro Theta el que varíe, representando el ángulo girado, mientras que en pares prismáticos será di el que varíe representando el espacio recorrido por el par prismático.

Una vez están asignados los sistemas de referencia a cada una de las barras del robot e identificados los parámetros de Denavit y Hartenberg, se procederá a formular la relación cinemática entre dos barras adyacentes mediante matrices de transformación homogénea.

Para hacer coincidir el sistema de referencia $\{O_i - X_i Y_i Z_i\}$ con el $\{H_i - X_i Y' Z_{i-1}\}$ deberán realizarse dos acciones: un giro de valor α_i alrededor del eje de las X_i junto con una traslación de valor a_i a lo largo del eje X_i .

$$ {}^{int}A_i = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & \cos(\alpha_i) & -\text{sen}(\alpha_i) & 0 \\ 0 & \text{sen}(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9) $$

Para conseguir que ambos sistemas de referencia coincidan y así poder definir la posición y orientación de una manera adecuada con respecto al otro, se realizará un giro de valor q_i alrededor del eje *Z'* y una traslación de valor d_i a lo largo del eje Z'.

$$ {}^{i-1}A_{int} = \begin{bmatrix} \cos(\theta_i) & -\text{sen}(\theta_i) & 0 & 0 \\ \text{sen}(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (10) $$

Mediante la combinación de dichas matrices de transformación homogéneas, se tendrá:

$$ {}^{i-1}A_i = {}^{i-1}A_{int} \cdot {}^{int}A_i = \begin{bmatrix} \cos(\theta_i) & -\cos(\alpha_i)\cdot\text{sen}(\theta_i) & \text{sen}(\alpha_i)\cdot\text{sen}(\theta_i) & a_i\cdot\cos(\theta_i) \\ \text{sen}(\theta_i) & \cos(\alpha_i)\cdot\cos(\theta_i) & -\text{sen}(\alpha_i)\cdot\cos(\theta_i) & a_i\cdot\text{sen}(\theta_i) \\ 0 & \text{sen}(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (11) $$

Dicha transformación representada gráficamente quedaría de la siguiente manera:

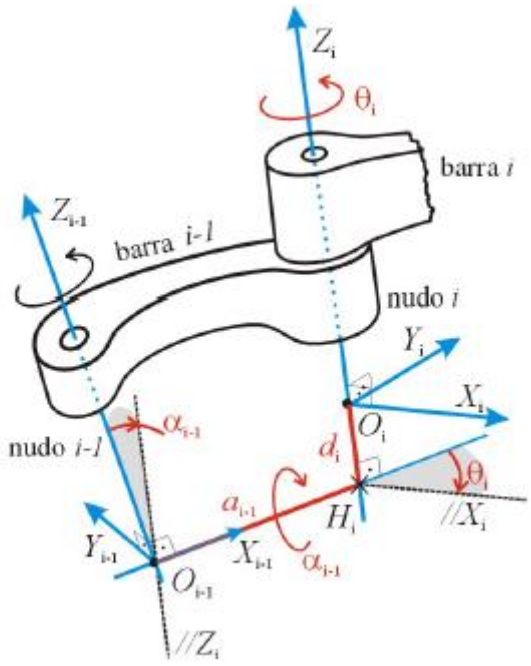


Figura 10: Aplicación método DH

Es importante remarcar que existen dos excepciones en el procedimiento de notación de Denavit y Hartenberg. Estas excepciones son la asignación de sistemas de referencia al elemento terminal y a la base, dado que en ambos casos no habrá perpendicular común. Para el caso del elemento terminal, el origen del sistema de referencia se considerará en cualquier punto de interés, mientras que el eje X del elemento terminal debe intersecar en ángulo recto con el del último nudo. En el caso de la base el origen del sistema de referencia se escoge de manera arbitraria según las preferencias, con la condición de que el eje Z de este ha de ser paralelo al del nudo uno.

3 Cinemática de Posición

La cinemática de robots estudia los movimientos de estos respecto a un sistema de referencia que se establece fijo. Dicho de otra forma, la cinemática describe de una manera analítica el movimiento o movimientos que el robot realiza en el espacio en función del tiempo, particularizado por las relaciones entre la posición y la orientación del extremo final del robot con los valores de sus coordenadas articulares.

Existen dos partes dentro de la cinemática. Una es el problema cinemático directo el cual se basa en la determinación de la posición y orientación del efector final del robot con respecto a un sistema de coordenadas fijo, todo ello conociendo los valores de las articulaciones, así como los parámetros geométricos del robot.

Por otro lado, está el problema cinemático inverso, el cual obtendrá las coordenadas de las articulaciones conociendo la posición y orientación del elemento terminal del robot.

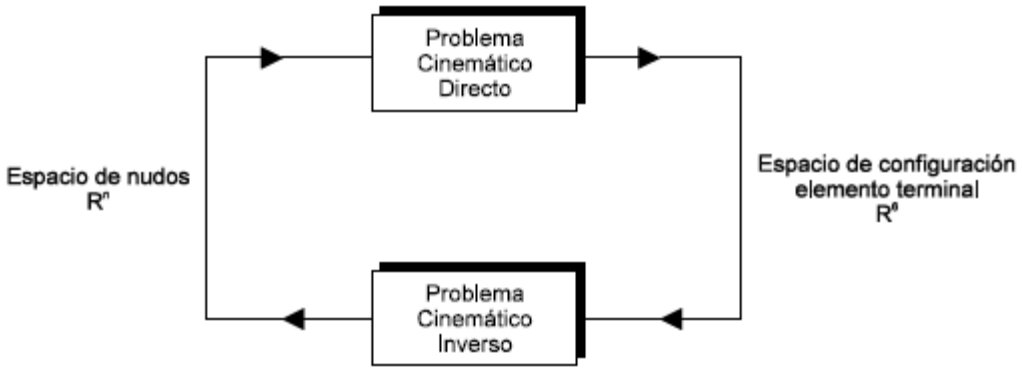


Figura 11: Cinemática directa, cinemática inversa

3.1 Problema Cinemático Directo

Como se ha expuesto en apartados anteriores, mediante la notación de Denavit y Hartenberg, es posible expresar la posición y la orientación del elemento terminal en función de las variables de los nudos, representadas de la siguiente forma: $q_i \equiv \theta_i$ en el caso que los nudos sean de revolución, (como es el caso de todos los nudos del robot PUMA 560).

Un robot con n grados de libertad, será modelado con una concatenación de (n+1) barras, cuyas localizaciones espaciales están expresadas a través de matrices de transformación homogéneas entre los diferentes sistemas de referencia locales.

Considerando las n transformaciones consecutivas sobre la secuencia de barras adyacentes, se podrá obtener la localización del elemento terminal con respecto al sistema de referencia fijo de la siguiente manera:

$$T = {}^{0}A_1(q_1)\,{}^{1}A_2(q_2)\cdots{}^{n-1}A_n(q_n) \qquad (12)$$

Donde la matriz de transformación homogénea T, describe la posición y orientación del elemento terminal con respecto a un sistema de referencia ligado a la base del manipulador. Esta expresión se denomina ecuación cinemática del robot y la matriz T como matriz de brazo robot, siendo esta única, por lo que solo existirá una posible solución para cada configuración.

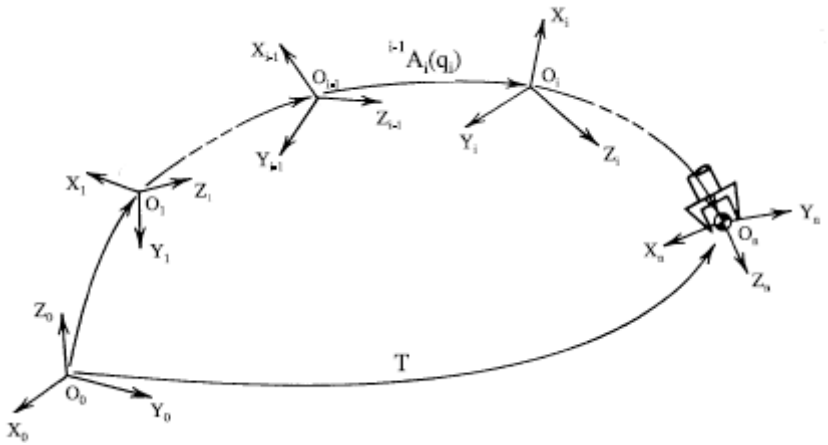


Figura 12: Localización elemento terminal

Se analizará el movimiento instantáneo del robot, es decir, determinar la posición, velocidad y aceleraciones del elemento efector final y de los nudos.

3.2 Problema Cinemático Inverso

El problema cinemático que trata de situar el elemento terminal en una posición y con una orientación predeterminada, se denomina Problema Cinemático Inverso. En este caso la ecuación cinemática debe resolverse en los desplazamientos de los nudos, dada la posición y orientación del efector final, representada mediante la matriz T. Una vez resuelta dicha ecuación cinemática, se podrá realizar el movimiento previsto del elemento terminal, asignando a cada uno de los nudos del robot, valores de los desplazamientos obtenidos anteriormente.

El Problema Cinemático Inverso, resulta de una mayor complejidad que el Directo. Mientras que en el Problema Directo se obtiene una única solución, que representa la localización del elemento terminal para un determinado conjunto de desplazamientos en los nudos, en el problema Inverso pueden existir diversas soluciones (valores de desplazamientos en los nudos), que sean compatibles con una misma localización del elemento terminal. Por otro lado, también se puede dar el caso de que para una determinada estructura de brazo robot, y en un determinado rango de posiciones y orientaciones del elemento terminal, puede no existir solución para dicho problema cinemático.

En el Problema Cinemático Inverso, la ecuación cinemática está formada por varias ecuaciones no lineales (trascendentales), con numerosas funciones trigonométricas, y no siempre es posible obtener soluciones explícitas de las variables de nudo en función de la localización deseada del elemento terminal. Esto hace que la resolución de este problema sea de un alto coste en cuanto a tiempo de computación se refiere.

La existencia de soluciones explícitas depende de la estructura cinemática del brazo del robot. Para que sea posible la resolución explícita de este problema para un robot con seis grados de libertad, (como es el caso del PUMA), se ha de cumplir la condición de Pieper.

La condición de Pieper dice: "Es condición suficiente para la resolubilidad explícita de un problema cinemático inverso, la existencia de tres nudos de revolución consecutivos cuyos ejes se intersecten en un solo punto para todas las posibles configuraciónes del brazo".

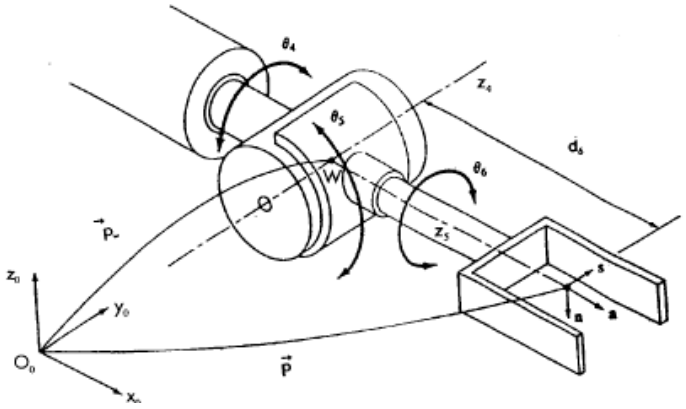


Figura 13: Muñeca esférica

Tres nudos dispuestos de ese modo, como se muestra en la Figura, constituyen una "muñeca esférica" (spherical wrist). Esta configuración se encuentra en la inmensa mayoría de los robots industriales.

La solución para el robot PUMA 560 vendría dada por:

$$p_{wx} = p_x - d_6 \cdot a_x$$
$$p_{wy} = p_y - d_6 \cdot a_y$$
$$p_{wz} = p_z - d_6 \cdot a_z$$

$$\lambda = \frac{\left(p_{wx} \cdot C_1 + p_{wy} \cdot S_1\right)^2 + p_{wz}^2 - a_3^2 - a_2^2 - d_4^2}{2 \cdot a_2} \qquad (13,14,15,16)$$

$$\theta_1 = \operatorname{atan2}\left(\frac{-p_{wx}}{p_{wy}}\right) + \operatorname{atan2}\left(\frac{\pm\sqrt{\left(p_{wx}^{\ 2} + p_{wy}^{\ 2} - d_2^2\right)}}{d_2}\right)$$

$$\theta_3 = \operatorname{atan2}\left(\frac{d_4}{a_3}\right) + \operatorname{atan2}\left(\frac{\pm\sqrt{\left(a_3^{\ 2} + d_4^{\ 2} - \lambda^2\right)}}{\lambda}\right)$$

$$\theta_2 = \operatorname{atan2}\left(\frac{\left(C_1 \cdot p_{wx} + S_1 \cdot p_{wy}\right)\cdot\left(-a_3 \cdot S_3 + d_4 \cdot C_3\right) - p_{wz} \cdot \left(a_2 + a_3 \cdot C_3 + d_4 \cdot S_3\right)}{\left(C_1 \cdot p_{wx} + S_1 \cdot p_{wy}\right)\cdot\left(a_2 + a_3 \cdot C_3 + d_4 \cdot S_3\right) + p_{wz} \cdot \left(-a_3 \cdot S_3 + d_4 \cdot C_3\right)}\right)$$

$$\theta_4 = \operatorname{atan2}\left(\frac{-S_1 \cdot a_x + C_1 \cdot a_y}{C_1 \cdot C_{23} \cdot a_x + S_1 \cdot C_{23} \cdot a_y - S_{23} \cdot a_z}\right) \quad \acute{o} \quad \theta_4 = \operatorname{atan2}\left(\frac{S_1 \cdot a_x - C_1 \cdot a_y}{-C_1 \cdot C_{23} \cdot a_x - S_1 \cdot C_{23} \cdot a_y + S_{23} \cdot a_z}\right)$$

$$\theta_5 = \operatorname{atan2}\left(\frac{\pm\sqrt{\left(C_1 \cdot a_y - S_1 \cdot a_x\right)^2 + \left(C_1 \cdot C_{23} \cdot a_x + S_1 \cdot C_{23} \cdot a_y - S_{23} \cdot a_z\right)^2}}{C_1 \cdot S_{23} \cdot a_x + S_1 \cdot S_{23} \cdot a_y + C_{23} \cdot a_z}\right)$$

$$\theta_6 = \operatorname{atan2}\left(\frac{C_1 \cdot S_{23} \cdot s_x + S_1 \cdot S_{23} \cdot s_y + C_{23} \cdot s_z}{-\left(C_1 \cdot C_{23} \cdot n_x + S_1 \cdot C_{23} \cdot n_y + C_{23} \cdot n_z\right)}\right) \quad \acute{o} \quad \theta_6 = \operatorname{atan2}\left(\frac{-C_1 \cdot S_{23} \cdot s_x - S_1 \cdot S_{23} \cdot s_y - C_{23} \cdot s_z}{C_1 \cdot C_{23} \cdot n_x + S_1 \cdot C_{23} \cdot n_y + C_{23} \cdot n_z}\right) \qquad (17\text{-}22)$$

La presencia de funciones trigonométricas inversas conduce a varias soluciones del problema cinemático inverso de posición:

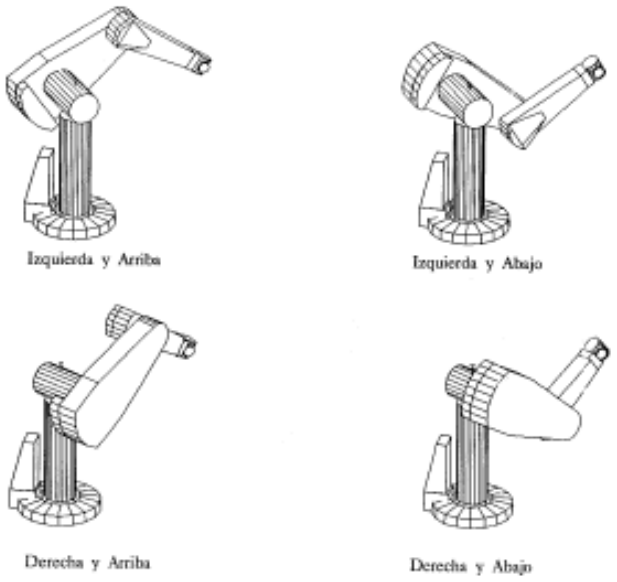


Figura 14: Posibles configuraciones del PUMA 560

4 Cinemática de Movimiento

En este apartado se abordará la relación existente entre la velocidad y aceleración con la que se mueve el elemento terminal junto con las correspondientes velocidades y aceleraciones experimentadas por los diversos elementos constitutivos del brazo robótico, principalmente los pares cinemáticos del mismo.

El movimiento que realiza el robot se define normalmente en el espacio cartesiano, sin embargo, el robot controla su movimiento en el espacio de los nudos, por tanto, será necesario implementar la relación que existe entre ambos movimientos.

4.1 Jacobiana del robot

Primeramente, será necesaria la definición de los siguientes vectores:

$$\dot{p} = \begin{bmatrix} \dot{r}_{O_n} \\ \dot{\phi}_n \end{bmatrix} \quad y \quad \dot{q} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_n \end{bmatrix} \longrightarrow \dot{p} = J \cdot \dot{q} \longrightarrow \vec{\dot{p}}_{6\times1} = J_{6\times n} \cdot \vec{\dot{q}}_{n\times1} \tag{23}$$

Los cuales se pueden ver representados en el efector final de un robot de la siguiente manera:

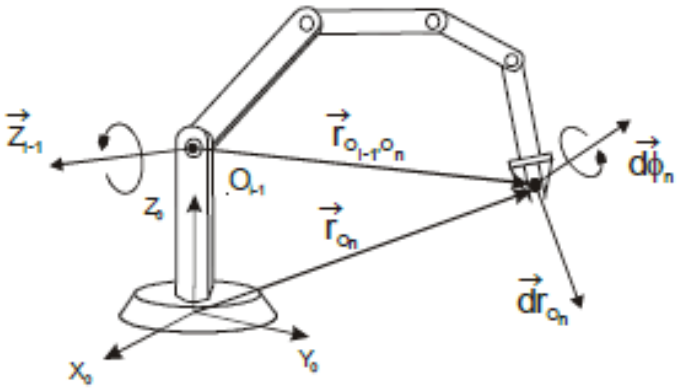


Figura 15: Representación vectores translación y rotación infinitesimal efector final

Estos vectores, representan la translación infinitesimal del elemento terminal por medio del vector tridimensional *dr*, mientras que la rotación infinitesimal se indica mediante $d\phi$ también de tres dimensiones. Estos vectores, es necesarios definirlos para poder aplicar el método de Whitney, el cual está basado en la identificación de las columnas de la matriz jacobiana, ya que la obtención de dicha matriz mediante el método de diferenciación no constituye un método computacionalmente eficiente.

Basándose en dicho método de identificación de columnas de la matriz jacobiana y aplicando el método de Whitney se obtiene esquemáticamente la siguiente matriz:

$$J_{6\times n}=\begin{matrix}\text{Nudo 1} & \text{Nudo 2} & & \text{Nudo n}\end{matrix}\begin{bmatrix}\vec{J}_{L_1} & \vec{J}_{L_2} & \cdots & \vec{J}_{L_n}\\ \vec{J}_{A_1} & \vec{J}_{A_2} & \cdots & \vec{J}_{A_n}\end{bmatrix}\begin{matrix}\text{velocidad lineal elemento terminal}\\ \text{velocidad angular elemento terminal}\end{matrix}$$

Hay que tener en cuenta que el número de filas del Jacobiano será igual al número de grados de libertad que se consideren en el espacio cartesiano, y el número de columnas será el número de grados de libertad que posea el robot.

Aplicando la ecuación previamente definida, la velocidad lineal del sistema de referencia del elemento terminal, vendrá definida por la siguiente expresión:

$$\dot{\vec{r}}_e=\vec{J}_{L_1}\cdot\dot{q}_1+\vec{J}_{L_2}\cdot\dot{q}_2+\cdots+\vec{J}_{L_n}\cdot\dot{q}_n \tag{24}$$

Por otro lado, la velocidad angular del mismo vendrá dada por la siguiente expresión:

$$\dot{\vec{\phi}}_e=\vec{J}_{A_1}\cdot\dot{q}_1+\vec{J}_{A_2}\cdot\dot{q}_2+\cdots+\vec{J}_{A_n}\cdot\dot{q}_n \tag{25}$$

Lo cual nos dará como resultado diferentes expresiones dependiendo del tipo de par cinemático que sea, si es un par prismático se obtendrá la siguiente expresión:

$$\begin{aligned}&\vec{J}_{L_i}\cdot\dot{q}_i={}^0\vec{z}_{i-1}\cdot\dot{q}_i\Rightarrow\vec{J}_{L_i}={}^0\vec{z}_{i-1}\\&\vec{J}_{A_i}\cdot\dot{q}_i=0\qquad\Rightarrow\vec{J}_{A_i}=0\end{aligned} \tag{26}$$

Mientras que si se trata de un nudo de revolución se tendrá la siguiente expresión:

$$\begin{aligned}&\vec{J}_{L_i}\cdot\dot{q}_i={}^0\vec{\omega}_i\times{}^0\vec{r}_{O_{i-1},O_e}=\left({}^0\vec{z}_{i-1}\cdot\dot{q}_i\right)\times{}^0\vec{r}_{O_{i-1},O_n}=\left({}^0\vec{z}_{i-1}\times{}^0\vec{r}_{O_{i-1},O_n}\right)\cdot\dot{q}_i\Rightarrow\\&\Rightarrow\vec{J}_{L_i}={}^0\vec{z}_{i-1}\times{}^0\vec{r}_{O_{i-1},O_n}\\&\vec{J}_{A_i}\cdot\dot{q}_i={}^0\vec{z}_{i-1}\cdot\dot{q}_i\quad\Rightarrow\vec{J}_{A_i}={}^0\vec{z}_{i-1}\end{aligned} \tag{27}$$

Con esto previamente definido únicamente faltaría definir los vectores ${}^{O}Z_{i-1}$ y ${}^{O}r_{i-1,on}$ los cuales serán definidos de la siguiente manera:

$$ {}^{i-1}\vec{z}_{i-1} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \Rightarrow {}^{0}\vec{z}_{i-1} = {}^{0}R_1(q_1)\cdot{}^{1}R_2(q_2)\cdot\ldots\cdot{}^{i-2}R_{i-1}(q_{i-1})\cdot{}^{i-1}\vec{z}_{i-1} $$

$$ {}^{0}\vec{r}_{O_{i-1},O_n} = {}^{0}\vec{r}_{O_0,O_n} - {}^{0}\vec{r}_{O_0,O_{i-1}} \Rightarrow \begin{cases} {}^{0}\vec{r}_{O_0,O_n} \Rightarrow \left[{}^{0}A_1(q_1)\cdot{}^{1}A_2(q_2)\cdot\ldots\cdot{}^{n-1}A_n(q_n)\right]_{2,4}^{1,4} \\ \vec{r}_{O_0,O_{i-1}} \Rightarrow \left[{}^{0}A_1(q_1)\cdot{}^{1}A_2(q_2)\cdot\ldots\cdot{}^{i-2}A_{i-1}(q_{i-1})\right]_{2,4}^{1,4} \end{cases}_{3,4} \quad (28) $$

Con todo esto se podrá definir la matriz Jacobiana para un robot de 6 GDL, como en el caso del PUMA, de la siguiente manera:

$$ J = \left[\underset{\vec{z}_0}{\vec{r}_0 \Lambda (\vec{r}_6 - \vec{r}_0)} \quad \underset{\vec{z}_1}{\vec{r}_1 \Lambda (\vec{r}_6 - \vec{r}_1)} \quad \underset{\vec{z}_2}{\vec{r}_2 \Lambda (\vec{r}_6 - \vec{r}_2)} \quad \underset{\vec{z}_3}{\vec{r}_2 \Lambda (\vec{r}_6 - \vec{r}_2)} \quad \underset{\vec{z}_4}{\vec{r}_4 \Lambda (\vec{r}_6 - \vec{r}_4)} \quad \underset{\vec{z}_5}{\vec{r}_5 \Lambda (\vec{r}_6 - \vec{r}_5)} \right] \quad (29) $$

Siendo su derivada la siguiente expresión:

$$ \dot{J} = \left[\underset{\vec{z}_0}{\frac{\partial}{\partial q_1}\cdot\vec{r}_6} \quad \underset{\vec{z}_1}{\frac{\partial}{\partial q_2}\cdot\vec{r}_6} \quad \underset{\vec{z}_2}{\frac{\partial}{\partial q_3}\cdot\vec{r}_6} \quad \underset{\vec{z}_3}{\frac{\partial}{\partial q_4}\cdot\vec{r}_6} \quad \underset{\vec{z}_4}{\frac{\partial}{\partial q_5}\cdot\vec{r}_6} \quad \underset{\vec{z}_5}{\frac{\partial}{\partial q_6}\cdot\vec{r}_6} \right] \quad (30) $$

Siendo $r_n$los vectores que corresponden al vector de traslación de la matriz homogénea, de dimensiones 1x3 cada uno.

Por otro lado, los vectores Z_{n-1} son correspondientes a las submatrices de rotación dentro de la matriz de transformación homogénea correspondiente siendo también de dimensiones 1x3.

En el caso de este proyecto, se ha hallado la primera fila de la jacobiana del robot PUMA 560 de manera diferencial.

4.2 Cinemática directa diferencial

Esta permite obtener la velocidad y aceleración lineal y angular del elemento terminal en función de las velocidades y aceleraciones de los nudos. Dicha es la que se aplicará posteriormente en el desarrollo del proyecto.

Una vez se ha calculado la matriz Jacobiana en función de la posición de los nudos, la resolución de esta es inmediata con la aplicación de las siguientes fórmulas para la velocidad y la aceleración:

$$\vec{\dot{p}} = J \cdot \vec{\dot{q}}$$
$$\vec{\ddot{p}} = \dot{J} \cdot \vec{\dot{q}} + J \cdot \vec{\ddot{q}} \qquad (31)$$

En el caso de la segunda fórmula la obtención de la derivada de la matriz Jacobina se realizará de la siguiente manera:

$$\dot{J} = \frac{\partial J}{\partial q_1} \cdot \dot{q}_1 + \frac{\partial J}{\partial q_2} \cdot \dot{q}_2 + \ldots + \frac{\partial J}{\partial q_n} \cdot \dot{q}_n \qquad (32)$$

4.3 Cinemática inversa diferencial

En este caso, el siguiente procedimiento permite calcular las velocidades y aceleraciones de los nudos a partir del movimiento del elemento terminal.

Este problema es mucho más completo y su resolución depende de la no singularidad de la inversa de la matriz Jacobiana, es decir cuando el determinante de dicha matriz no sea 0. En caso afirmativo, en consecuencia, dicha matriz no será invertible, lo cual sucederá cuando el robot alcanza una configuración singular o también llamada degenerada. Cerca de las configuraciones singulares, las velocidades en el espacio de los nudos, necesarias para mantener valores razonables de velocidad en el elemento terminal, pueden hacerse extremadamente grandes tendiendo a infinito. Estas configuraciones ocurren en la frontera del espacio de trabajo del robot y pueden ser tanto interiores como exteriores.

Las ecuaciones para la resolución de este problema son las siguientes:

$$\vec{\dot{q}} = J^{-1} \cdot \vec{\dot{p}}$$
$$\vec{\ddot{q}} = J^{-1} \cdot \left[\vec{\ddot{p}} - \dot{J} \cdot \vec{\dot{q}}\right] \quad \text{siendo } \dot{J} = \frac{\partial J}{\partial q_1} \cdot \dot{q}_1 + \frac{\partial J}{\partial q_2} \cdot \dot{q}_2 + \ldots + \frac{\partial J}{\partial q_n} \cdot \dot{q}_n \qquad (33)$$

En este proyecto no se resolverá dicho problema dada las altas dificultades y al inmenso número de posibilidades que conlleva.

5 Generación de trayectorias

La función de un generador de trayectorias es generar las entradas de referencia para el sistema de control de movimiento, el cual se encarga de garantizar que el robot ejecute el recorrido planificado en el tiempo planificado.

Las trayectorias pueden especificarse de dos tipos:

-En el espacio de los nudos: trata de pasar de la configuración de articulación $Q_i = (q_{i1}, q_{i2}, q_{i3}, q_{i4}, q_{i5}, q_{i6})$ a otra $Q_f = (q_{f1}, q_{f2}, q_{f3}, q_{f4}, q_{f5}, q_{f6})$.

-En el espacio cartesiano: pasa de la localización $L1 = (x1, y1, z1, \alpha 1, \beta 1, \gamma 1)$ a otra $L2 = (x1, y1, z1, \alpha 1, \beta 1, \gamma 1)$. Todas ellas hacen referencia a la matriz de transformación homogénea del efector final la cual contiene tanto su orientación como su posición.

5.1 Generación de trayectorias en el espacio de los nudos

La generación de trayectorias en el espacio de las articulaciones es más sencilla, y permite generar funciones lineales para cada uno de los ejes, sin embargo, genera un movimiento no lineal para el efector final.

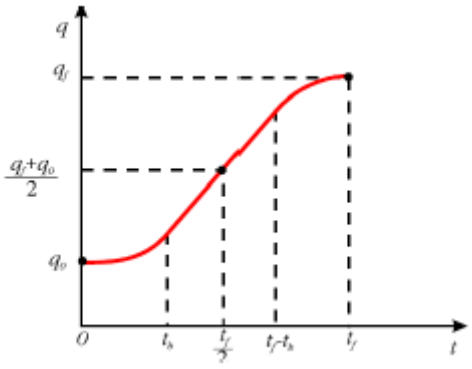


Figura 16: Trayectoria de segmentos lineales con tramos parabólicos

Uno de los métodos más comunes para generar trayectorias en el espacio de las articulaciones, (y el que se va a emplear en este trabajo), es el de segmentos lineales con tramos parabólicos. Este constituye un método muy simple, altamente utilizado y también genera velocidades constantes durante un tramo temporal.

Como se puede observar en la Figura 16, se tiene un tramo de velocidad constante entre los segmentos t_b y $t_f - t_b$. Dicho tramo se debe unir mediante segmentos parabólicos de una manera suave, mediante una continuidad en las derivadas.

Como se puede apreciar en la anterior Figura 16, la función presenta una simetría respecto del punto $\{\frac{tf}{2}, \frac{qf+q0}{2}\}$ además se suponen velocidades nulas en los instantes 0 y tf.

La pendiente del tramo lineal se corresponde a la velocidad constante V.

Para la obtención de los tramos parabólicos se emplea la siguiente ecuación:

$$q(t)=\alpha+\beta\cdot t+\gamma\cdot t^2 \quad (34)$$

-Para el primer tramo [0,tb] basándose en las condiciones predefinidas anteriormente en los extremos tanto de posición como de velocidad, llegamos a las siguientes ecuaciones:

$$\left.\begin{aligned} q(0)=q_0 \Rightarrow \alpha=q_0 \\ \dot{q}(0)=0 \Rightarrow \beta=0 \\ \dot{q}(t_b)=V \Rightarrow \gamma=\frac{V}{2\cdot t_b} \end{aligned}\right\} \quad q(t)=q_0+\frac{V}{2\cdot t_b}\cdot t_2;\quad 0\le t\le t_b \quad (35)$$

-Segundo tramo [tb,tf-tb], de donde saldrá la siguiente ecuación:

$$q(t)=\frac{q_f+q_0-V\cdot t_f}{2}+V\cdot t \quad (36)$$

-Por último, para el tercer tramo [tf-tb,tf], se obtendrá de un modo similar al tramo uno la siguiente ecuación:

$$q(t)=q_f-\frac{\frac{V}{t_b}\cdot t_f^2}{2}+\frac{V}{t_b}\cdot t_f\cdot t-\frac{\frac{V}{t_b}}{2}\cdot t^2 \quad (37)$$

Las dos expresiones anteriores han de ser coincidentes en tb, por lo cual se deducirá la siguiente expresión:

$$t_b=\frac{q_0-q_f+V\cdot t_f}{V} \quad (38)$$

De dicha ecuación, despejando la velocidad se tiene:

$$V=\frac{q_f-q_0}{t_f-t_0} \quad (39)$$

Sabiendo que el valor de tb no puede ser mayor que $\frac{tf}{2}$ de esta condición, se puede obtener el máximo valor que se le podría asignar a la velocidad:

$$V=\frac{2\cdot(q_f-q_0)}{t_f} \quad (40)$$

6 Dinámica de robots

El comportamiento dinámico de un sistema formado por n sólidos rígidos unidos entre sí por pares cinemáticos ideales y con n grados de libertad puede describirse mediante n ecuaciones diferenciales de segundo orden, altamente no lineales y acopladas.

Dichas ecuaciones diferenciales son conocidas como ecuaciones dinámicas del movimiento del sistema mecánico y conforman su modelo dinámico.

En el modelo dinámico de un robot existen dos aspectos principales a considerar: el movimiento y las acciones. Por un lado, el movimiento de un sistema consiste en una secuencia de posiciones, velocidades y aceleraciones de algún punto o puntos del sistema previamente definida, dicha secuencia se denomina trayectoria y ya ha sido explicada en apartados anteriores. Las acciones bien sean fuerzas o pares, se clasifican a su vez en internas (de restricción o reacción) y externas (aplicadas o de inercia). Por ello el modelo dinámico del robot debe describir la relación entre el movimiento y las acciones que lo han causado, de modo que dado uno de estos grupos de magnitudes se pueda calcular el otro.

Una vez establecido el problema dinámico, se pueden considerar fundamentalmente los siguientes problemas: el Problema Dinámico Inverso y el Problema Dinámico Directo.

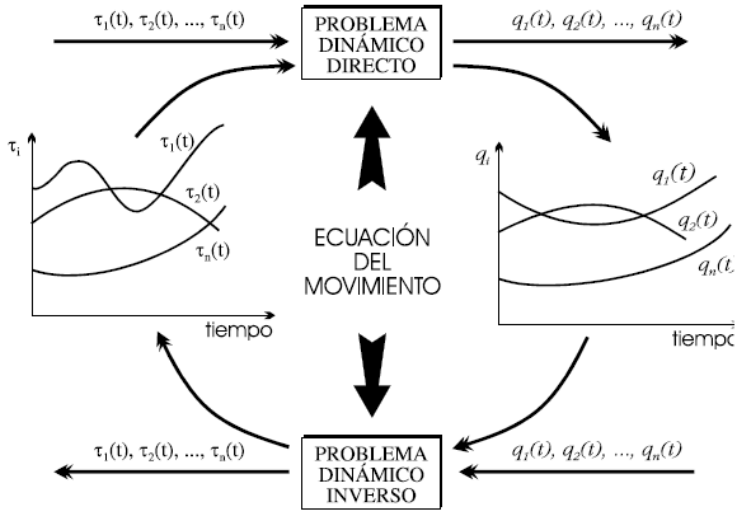


Figura 17: Dinámica de robots

El Problema Dinámico Inverso fue el primero en ser abordado. La computación eficiente de todos los pares o fuerzas es esencial cuando se trata el diseño e implementación de un sistema de control para un brazo robótico. Este pretende tener en consideración las características dinámicas del robot. Dicho problema dinámico será el desarrollado posteriormente en este proyecto.

Por otro lado, el Problema Dinámico Directo pretende la simulación del comportamiento dinámico del robot y tiene su aplicación tanto en el diseño mecánico del robot como en la simulación de nuevas estrategias de control.

En resumen, el objetivo que tiene la dinámica de robots consiste en obtener algoritmos dinámicos eficientes a nivel computacional. Es por ello por lo que se pueden considerar distintos principios de la dinámica a partir de los cuales obtener las ecuaciones del movimiento, distintos

tipos de formulación, distintos tipos de variables con las que formular el problema, distintas maneras de expresar las variables que intervienen en los algoritmos y distintos tipos de procesamientos de dichos algoritmos.

6.1 Formulación de Lagrange-Euler

Este es el tipo de formulación que se empleará en el proyecto. Es una formulación cerrada en la cual las ecuaciones del movimiento están expresadas en términos de la cinemática de nudos. La formulación de Lagrange adopta la siguiente configuración para las cadenas cinemáticas abiertas:

$$\vec{\tau}=D(\vec{q})\ddot{\vec{q}}+\vec{h}(\vec{q},\dot{\vec{q}})+\vec{n}(\vec{q}) \quad (41)$$

Siendo D, la matriz de inercia generalizada, h el vector que incluye los términos centrífugos y de Coriolis y n el vector que incluye los términos gravitatorios.

Las ecuaciones de Lagrange-Euler son las siguientes:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right)-\frac{\partial L}{\partial q_i}=\tau_i; \quad i=1,2,\ldots,n \quad (42)$$

Siendo la función de Lagrange:

$$L(q_i,\dot{q}_i)=K-P \quad (43)$$

Donde K representa la energía cinética y P la energía potencial del sistema mecánico.

$$K=\sum_{sistema}\left[\frac{1}{2}\cdot m_k\cdot(\vec{v}_{G_k})^2\right]+\sum_{sistema}\left[\frac{1}{2}\cdot I_{Gk}\cdot(\vec{\omega}_k)^2\right]$$
$$P=\sum_{sistema}-m_k\cdot(\vec{g})^T\cdot\vec{r}_{G_k} \quad (44)$$

En esta formulación no se consideran elementos elásticos.

Las ecuaciones del movimiento de Lagrange se deducirán a partir del tratamiento variacional de la ecuación del movimiento de Newton. Dado un vector r_p el cual define la posición de una partícula en el espacio, la ecuación de Newton con ello dará lugar a:

$$\vec{F}_P=m_P\cdot\frac{d^2[\vec{r}_P(q_1,q_2,\ldots,q_n)]}{dt^2} \quad (45)$$

Multiplicando la expresión anterior por el diferencial de dicho vector respecto a las variables de las articulaciones, y ya sumado para todas las partículas que componen el sistema mecánico, se tendrá que:

$$\sum_{s.m.}(\vec{F}_P)^T\cdot\frac{\partial\vec{r}_P}{\partial q_i}=\sum_{s.m.}m_P\cdot(\ddot{\vec{r}}_P)^T\cdot\frac{\partial\vec{r}_P}{\partial q_i} \quad (46)$$

Obteniendo así expresiones análogas para cada una de las coordenadas generalizadas del robot.

Si se deriva respecto al tiempo la expresión anterior se tendrá:

$$\frac{d(\vec{r}_P)}{dt} = \dot{\vec{r}}_P = \sum_{i=1}^{n} \frac{\partial \vec{r}_P}{\partial q_i} \cdot \dot{q}_i + \frac{\partial \vec{r}_P}{\partial t} \quad (47)$$

Si se deriva ahora respecto a las variables articulares se tendrá:

$$\frac{\partial \dot{\vec{r}}_P}{\partial \dot{q}_i} = \frac{\partial \vec{r}_P}{\partial q_i} \quad (48)$$

Por otro lado, se denota “K”, energía cinética como:

$$K = \sum_{s.m.} \frac{1}{2} \cdot m_P \cdot \left(\dot{\vec{r}}_P\right)^T \cdot \dot{\vec{r}}_P \quad (49)$$

La cual derivada respecto al tiempo y a las variables articulares quedará de la siguiente manera:

$$\frac{d\left(\frac{\partial K}{\partial \dot{q}_i}\right)}{dt} = \sum_{s.m.} m_P \cdot \left(\ddot{\vec{r}}_P\right)^T \cdot \frac{\partial \vec{r}_P}{\partial q_i} + \sum_{s.m.} m_P \cdot \left(\dot{\vec{r}}_P\right)^T \cdot \frac{\partial \dot{\vec{r}}_P}{\partial q_i} \quad (50)$$

De las expresiones anteriores y sabiendo que “Q” es:

$$Q_i = \sum_{s.m.} \left(\vec{F}_P \cdot \frac{\partial \vec{r}_P}{\partial q_i} \right) \quad (51)$$

Se puede sacar la expresión de la fuerza generalizada correspondiente a la coordenada generalizada i-ésima, esta última será el par o la fuerza que ejerce el actuador correspondiente a dicho nudo:

$$Q_i = \frac{d\left(\frac{\partial K}{\partial \dot{q}_i}\right)}{dt} - \frac{\partial K}{\partial q_i} \quad i = 1,2,\ldots,n \quad (52)$$

Clasificando las fuerzas que actúan sobre el sistema en gravitatorias o activas, se podría expresar la función de las fuerzas generalizadas del sistema de la siguiente manera:

$$Q_i^a = \frac{d\left(\frac{\partial K}{\partial \dot{q}_i}\right)}{dt} - \frac{\partial K}{\partial q_i} + \frac{\partial P}{\partial q_i} \quad i = 1,2,\ldots,n \quad (53)$$

Por último, si se introduce la función de Lagrange:

$$L(\vec{q}.\vec{q}) = K - P \quad (54)$$

Y se renombran por simplicidad las fuerzas generalizadas activas como ”Q”, se tendrá que:

$$\frac{d\left(\frac{\partial L}{\partial \dot{q}_i}\right)}{dt} - \frac{\partial L}{\partial q_i} = Q_i \quad i = 1,2,\ldots,n \quad (55)$$

Esta última ecuación representa la ecuación de movimiento de Lagrange.

6.2 Aplicación de las ecuaciones de Lagrange-Euler a un robot

Primeramente, se ha determinado la energía cinética correspondiente a la barra de un robot. Para ello primeramente se obtiene la velocidad de un punto cualquiera situado sobre dicha barra.

El vector que fija la posición de dicho punto sobre el sistema de referencia fijo se dotará por: ${}^{O}r_{OiP}$. Extendiendo este vector a cuarta dimensión (añadiéndole un 1), se tendrá lo siguiente:

$$\begin{bmatrix} {}^{0}\vec{r}_{O_0,P} \\ 1 \end{bmatrix} = {}^{0}A_i \cdot \begin{bmatrix} {}^{i}\vec{r}_{O_i,P} \\ 1 \end{bmatrix} \quad (56)$$

Que se denotará de la siguiente manera:

$${}^{0}\vec{r}_{O_0,P} = {}^{0}A_i(q_1,q_2,\ldots,q_n)\,{}^{i}\vec{r}_{O_i,P} \quad (57)$$

Gráficamente dicho vector sería el siguiente:

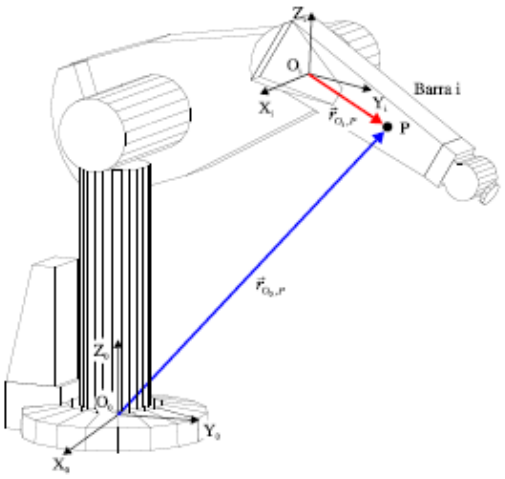


Figura 18: Vector ${}^{O}r_{OiP}$ representado en el PUMA 560

Por tanto, la velocidad absoluta del punto P, se podrá obtener de la siguiente manera:

$$ {}^0\dot{\vec{r}}_{O_0,P} = \frac{d\left({}^0\vec{r}_{O_0,P}\right)}{dt} = \left(\sum_{j=1}^{I} \frac{\partial {}^0A_I}{\partial q_j} \cdot \dot{q}_j\right) \cdot {}^I\vec{r}_{O_I,P} \tag{58} $$

Donde se ha tenido en cuenta que el vector de posición definido en el sistema de referencia local es constante.

Elevando al cuadrado la expresión anterior:

$$ \left({}^0\dot{\vec{r}}_{O_0,P}\right)^2 = \left({}^0\dot{\vec{r}}_{O_0,P}\right)^T \cdot \left({}^0\dot{\vec{r}}_{O_0,P}\right) \tag{59} $$

De forma matricial quedará:

$$ \left({}^0\dot{\vec{r}}_{O_0,P}\right)^2 = tr\left[{}^0\dot{\vec{r}}_{O_0,P} \cdot \left({}^0\dot{\vec{r}}_{O_0,P}\right)^T\right] \tag{60} $$

Sustituyendo en esta ecuación con la expresión de la velocidad absoluta en un punto P se podrá obtener:

$$ \begin{aligned} \left({}^0\dot{\vec{r}}_{O_0,P}\right)^2 &= tr\left[\left(\sum_{j=1}^{I} \frac{\partial {}^0A_I}{\partial q_j} \cdot \dot{q}_j\right) \cdot {}^I\vec{r}_{O_I,P} \cdot \sum_{k=1}^{I}\left(\frac{\partial {}^0A_I}{\partial q_k} \cdot \dot{q}_k \cdot {}^I\vec{r}_{O_I,P}\right)^T\right] = \\ &= tr\left[\sum_{j=1}^{I}\sum_{k=1}^{I} \frac{\partial {}^0A_I}{\partial q_j} \cdot {}^I\vec{r}_{O_I,P} \cdot \left({}^I\vec{r}_{O_I,P}\right)^T \cdot \left(\frac{\partial {}^0A_I}{\partial q_k}\right)^T \cdot \dot{q}_j \cdot \dot{q}_k\right] = \\ &= \sum_{j=1}^{I}\sum_{k=1}^{I} tr\left[\frac{\partial {}^0A_I}{\partial q_j} \cdot {}^I\vec{r}_{O_I,P} \cdot \left({}^I\vec{r}_{O_I,P}\right)^T \cdot \left(\frac{\partial {}^0A_I}{\partial q_k}\right)^T \cdot \dot{q}_j \cdot \dot{q}_k\right] \end{aligned} \tag{61} $$

Por lo tanto, la energía cinemática que está asociada a una partícula de masa “dm”, situada en el punto P de la barra i-ésima, será:

$$ \begin{aligned} dK_I &= \frac{1}{2} \cdot dm \cdot \left\{\sum_{j=1}^{I}\sum_{k=1}^{I} tr\left[\frac{\partial {}^0A_I}{\partial q_j} \cdot {}^I\vec{r}_{O_I,P} \cdot \left({}^I\vec{r}_{O_I,P}\right)^T \cdot \left(\frac{\partial {}^0A_I}{\partial q_k}\right)^T \cdot \dot{q}_j \cdot \dot{q}_k\right]\right\} = \\ &= \frac{1}{2} \cdot \left\{\sum_{j=1}^{I}\sum_{k=1}^{I} tr\left[\frac{\partial {}^0A_I}{\partial q_j} \cdot {}^I\vec{r}_{O_I,P} \cdot dm \cdot \left({}^I\vec{r}_{O_I,P}\right)^T \cdot \left(\frac{\partial {}^0A_I}{\partial q_k}\right)^T \cdot \dot{q}_j \cdot \dot{q}_k\right]\right\} \end{aligned} \tag{62} $$

La energía cinética de toda la barra vendrá dada por la expresión:

$$ K_I = \int dK_I = \frac{1}{2} \cdot \left\{\sum_{j=1}^{I}\sum_{k=1}^{I} tr\left[\frac{\partial {}^0A_I}{\partial q_j} \cdot \left(\int {}^I\vec{r}_{O_I,P} \cdot \left({}^I\vec{r}_{O_I,P}\right)^T \cdot dm\right) \cdot \left(\frac{\partial {}^0A_I}{\partial q_k}\right)^T \cdot \dot{q}_j \cdot \dot{q}_k\right]\right\} \tag{63} $$

El término integral de dentro de los paréntesis, se denomina pseudo matriz de inercia de la barra i respecto al sistema de referencia {Oi-Xi Yi Zi} y se denota por "H". Por lo tanto, la expresión anterior quedaría de la siguiente manera:

$$K_i = \frac{1}{2} \cdot \left\{ \sum_{j=1}^{i} \sum_{k=1}^{i} tr \left[\frac{\partial {}^0A_j}{\partial q_j} \cdot {}^iH_{O_i} \cdot \left(\frac{\partial {}^0A_k}{\partial q_k} \right)^T \cdot \dot{q}_j \cdot \dot{q}_k \right] \right\} \tag{64}$$

Se puede demostrar que los términos del tensor de pseudo inercia son los siguientes:

$${}^iH_{O_i} = \begin{bmatrix} \frac{-{}^iI_{O_{xx}} + {}^iI_{O_{yy}} + {}^iI_{O_{zz}}}{2} & {}^iI_{O_{xy}} & {}^iI_{O_{xz}} & {}^ix_{G_i} \cdot m_i \\ {}^iI_{O_{xy}} & \frac{{}^iI_{O_{xx}} - {}^iI_{O_{yy}} + {}^iI_{O_{zz}}}{2} & {}^iI_{O_{yz}} & {}^iy_{G_i} \cdot m_i \\ {}^iI_{O_{xz}} & {}^iI_{O_{yz}} & \frac{{}^iI_{O_{xx}} + {}^iI_{O_{yy}} - {}^iI_{O_{zz}}}{2} & {}^iz_{G_i} \cdot m_i \\ m_i \cdot {}^ix_{G_i} & m_i \cdot {}^iy_{G_i} & m_i \cdot {}^iz_{G_i} & m_i \end{bmatrix} \tag{65}$$

La relación de los momentos y productos de inercia respecto al sistema de referencia local a la barra i, {Oi-Xi Yi Zi}, con los correspondientes a un sistema de regencia paralelo al anterior, pero con origen en el centro de gravedad de la propia barra {Gi-Xi Yi Zi}, vendrá dado por expresiones del siguiente tipo:

$$\begin{aligned} {}^iI_{O_{XX}} &= {}^iI_{G_{XX}} + m_i \cdot \left[\left({}^iy_{G_i} \right)^2 + \left({}^iz_{G_i} \right)^2 \right] \\ {}^iI_{O_{XY}} &= {}^iI_{G_{XY}} + m_i \cdot \left[{}^ix_{G_i} + {}^iy_{G_i} \right] \end{aligned} \tag{66}$$

Por otra parte, la energía potencial del robot vendrá dada por la siguiente expresión:

$$P = -\sum_{i=1}^{n} m_i \cdot \bar{g}^T \cdot {}^0A_i \cdot {}^i\bar{r}_{O_i,G_i} \tag{67}$$

$$\bar{g} = \begin{bmatrix} 0 & 0 & -g & 0 \end{bmatrix}^T$$

Los últimos dos términos de la ecuación anterior hacen referencia al vector de posición desde el origen del sistema de referencia fijo al centro de gravedad de la barra i-ésima.

Aplicando ahora la ecuación de movimiento de Lagrange vista anteriormente, se tendrá la siguiente expresión para la ecuación del movimiento de un robot:

$$\sum_{j=1}^{n} D_{ij} \cdot \ddot{q}_j + \sum_{j=1}^{n} \sum_{k=1}^{n} h_{ijk} \cdot \dot{q}_j \dot{q}_k + n_i = \tau_i \quad i = 1,2,\ldots,n \tag{68}$$

Siendo:

$$D_{ij} = \sum_{p=\max(i,j)}^{n} tr\left[\frac{\partial {}^0A_p}{\partial q_k} \cdot {}^pH_{O_p} \cdot \left(\frac{\partial {}^0A_p}{\partial q_k}\right)^T\right]$$

$$h_{ijk} = \sum_{p=\max(i,j,k)}^{n} tr\left[\frac{\partial^2({}^0A_p)}{\partial q_k \partial q_m} {}^pH_{O_p} \cdot \left(\frac{\partial {}^0A_p}{\partial q_k}\right)^T\right]$$

$$P = -\sum_{i=1}^{n} m_i \cdot \vec{g}^T \cdot {}^0A_i \cdot {}^i\vec{r}_{O_i,G_i} \qquad (69)$$

Las expresiones anteriores son las que se utilizarán a la hora de hacer los cálculos de la dinámica inversa para el robot PUMA. Dicha expresión se puede expresar de forma reducida de la siguiente manera:

$$D(\vec{q}) \cdot \ddot{\vec{q}} + \vec{h}(\vec{q}, \dot{\vec{q}}) + \vec{n}(\vec{q}) = \vec{\tau} \qquad (70)$$

7 Desarrollo del proyecto

7.1 PUMA 560 principales dimensiones y sistemas de referencia

Como ya se ha definido en apartados anteriores, el PUMA 560 es un robot de seis grados de libertad, el cual fue diseñado para la investigación y la industria. Se trata de un robot de pequeñas dimensiones, versátil y muy útil cuando se trabajaba con componentes de tamaño y peso reducidos.

A continuación, se observa una Figura con las dimensiones y medidas principales de dicho robot, establecidas por criterios universales, así como el posicionamiento y orientación de los sistemas de referencia de cada una de sus barras y sus variables articulares (siguiendo el criterio de Denavit Hartenberg).

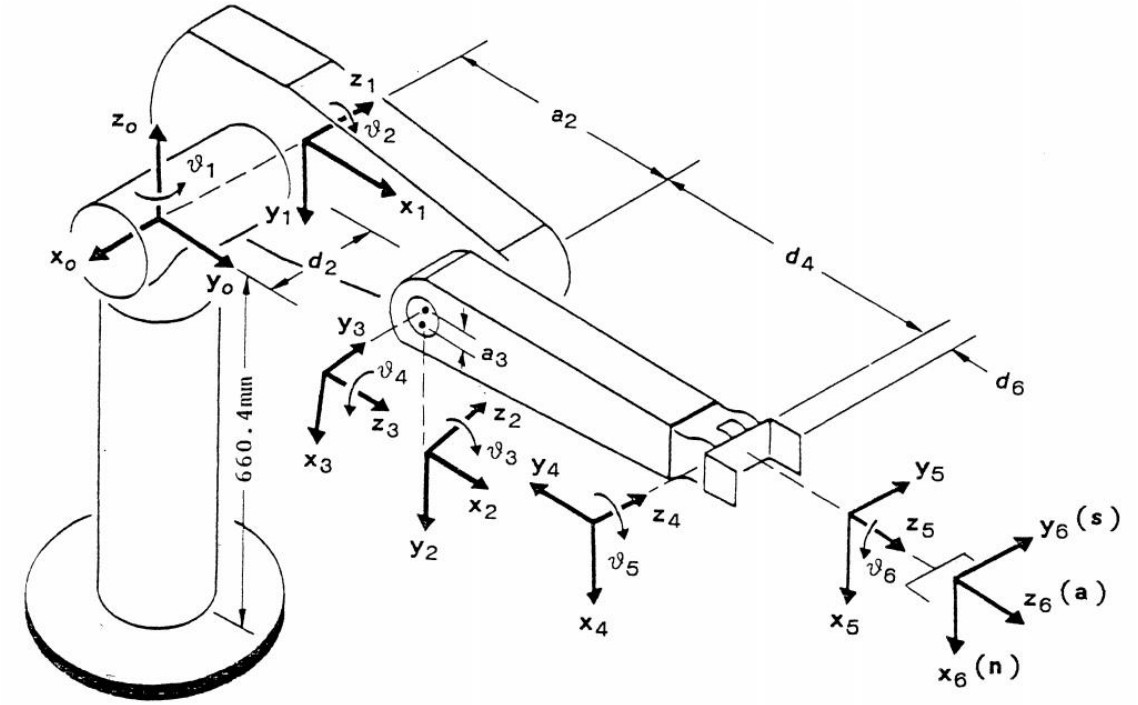
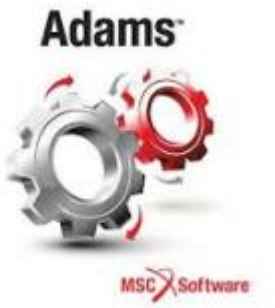


Figura 19: Dimensiones principales y sistemas de referencia del PUMA 560

7.2 Software utilizado

Con el fin de realizar el estudio de dicho robot PUMA 560 de seis grados de libertad, se tiene una inmensa cantidad de datos que tratar y al mismo tiempo simular y modelar para obtener más datos con el fin de hallar un modelo que resulte el más próximo a la realidad. Es por ello por lo que se ha utilizado un software específico de cada uno de los ámbitos de trabajo anteriormente citados.

- **MSC ADAMS View**



MSC ADAMS View es un software comercial capaz de realizar análisis sobre mecanismos. Está compuesto por varios módulos que permiten simular el funcionamiento de un mecanismo por medio de animaciones. También permite realizar el análisis de vibraciones, así como realizar el análisis de esfuerzos. En el caso de este proyecto, se usará tanto para la simulación cinemática como la dinámica del robot PUMA.

- **MATLAB**

MATLAB es un software matemático compuesto por un entorno de desarrollo integrado y un lenguaje de programación propio, este se llama lenguaje M. Este software permite una gran flexibilidad y facilidad a la hora de realizar cálculos con matrices. MATLAB también puede ser utilizado para representar datos y funciones, implementar algoritmos o crear interfaces de usuario.

MATLAB posee infinidad de usos, pero lo que resulta más interesante para este proyecto es el tratamiento de matrices u de datos y la capacidad que tiene de automatización.

- **Microsoft Excel**

Microsoft Excel es una aplicación distribuida por Microsoft Office para hojas de cálculo. Este programa no tendrá mucho peso en el desarrollo del trabajo únicamente será utilizado para agrupar y presentar datos y realizar algún cálculo.

7.3 Parametrización de Denavit-Hartenberg y ensamblado

Como se ha mencionado en apartados anteriores, la cinemática relaciona las coordenadas de las articulaciones con la posición y orientación del efector final del robot.

Para la correcta consecución del estudio de la cinemática directa de dicho robot, hay que basarse como también se ha explicado en apartados anteriores, en la notación de Denavit y Hartenberg. Dicha notación constituye un procedimiento totalmente sistemático, el cual se mostrará a continuación aplicado al caso del PUMA 560.

La notación de Denavit y Hartenberg aplicada al robot PUMA sigue el criterio de Paul, quedando los sistemas de referencia propios de cada barra de la manera que se muestra en las siguientes imágenes:

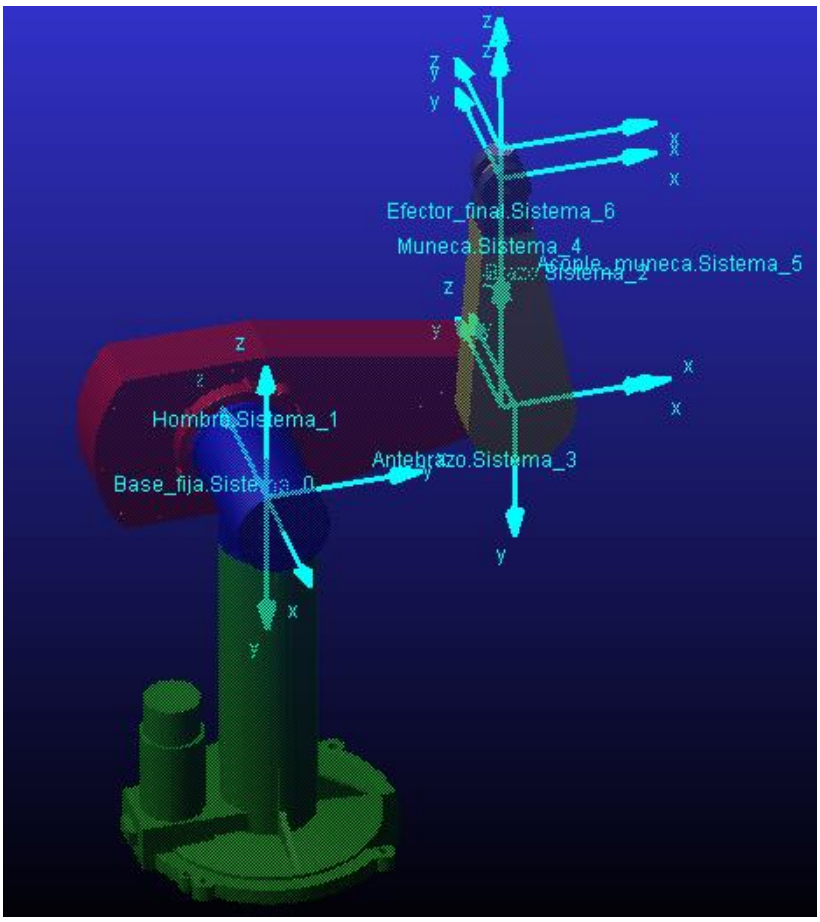


Figura 20: DH PUMA 560 ADAMS View vista 3D

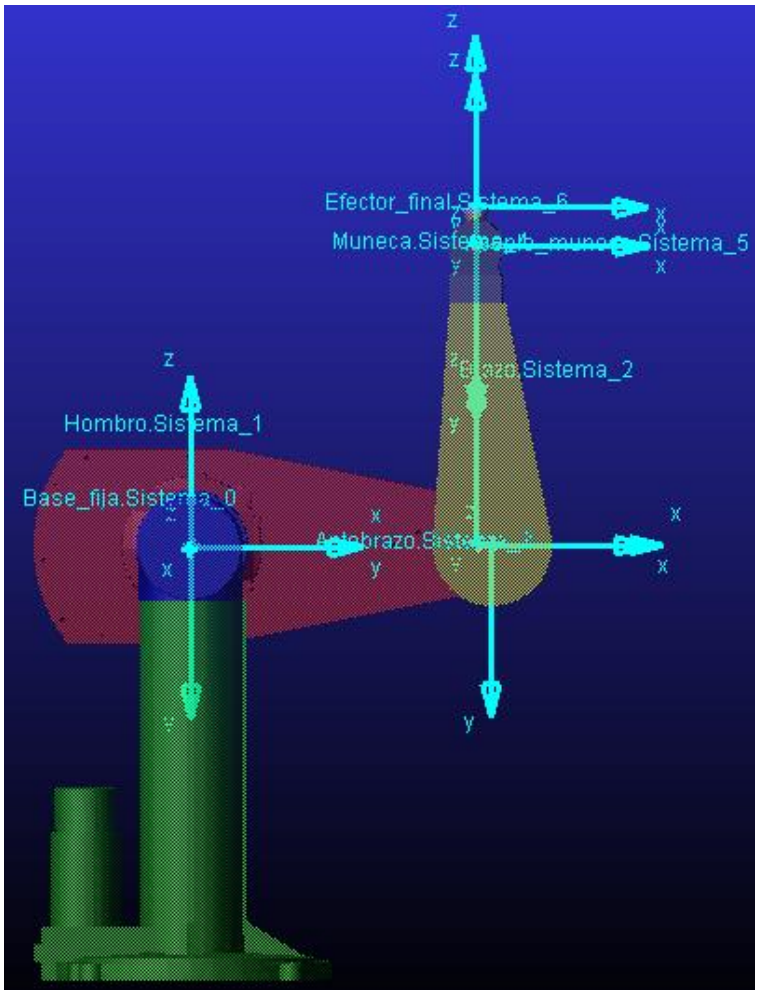


Figura 21: DH PUMA 560 ADAMS View vista alzado

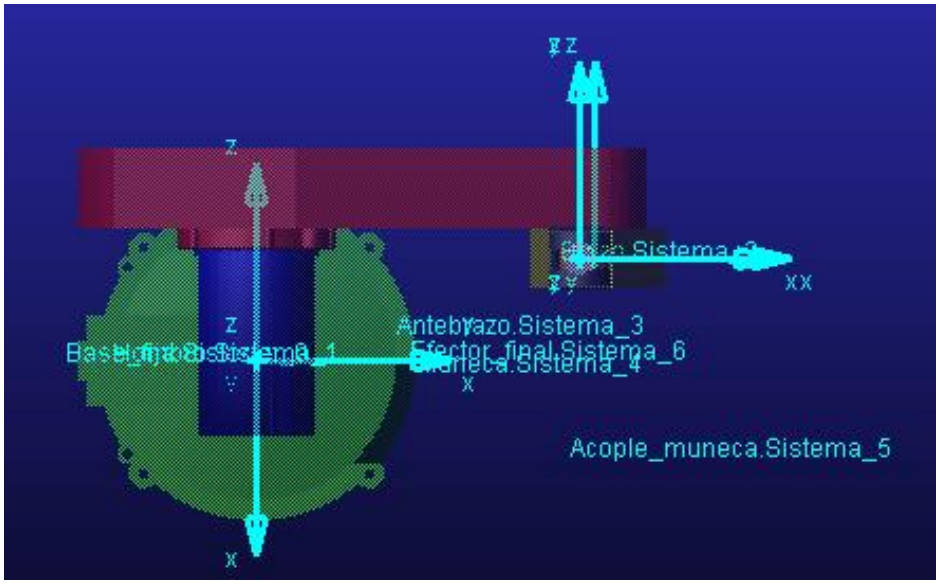


Figura 22: DH PUMA 560 ADAMS View vista planta

De la misma forma indicada por el algoritmo de Denavit y Hartenberg, se han obtenido los parámetros correspondientes que se muestran en la tabla 1:

Tabla 1: Parámetros DH

i	Alfa (rad)	a (mm)	d (mm)	Theta (rad)
1	-pi/2	0	0	q1
2	0	431.8	129.54	q2
3	Pi/2	-20.32	0	q3
4	-pi/2	0	432.09	q4
5	Pi/2	0	0	q5
6	0	0	56.26	q6

Todo este procedimiento se ha realizado previamente a mano, antes de realizar el modelo en ADAMS View, pero para mayor claridad se adjuntan directamente este tipo de tablas e imágenes en ADAMS View.

Una vez preparada la tabla de Denavit-Hartenberg, se procederá a la descarga del archivo CAD del brazo robótico, para su posterior uso en formato "Parasolid" compatible con ADAMS View.

Una vez ya se tengan los ficheros CAD descargados se procederá a abrir un fichero en ADAMS View y se irán importando cada una de las distintas partes del robot, a su vez se irán renombrando con un nombre característico desde el propio ADAMS View.

Una vez situada cada parte en su lugar correspondiente, se asignará a cada barra un *Marker*, donde se situará el sistema de referencia local de cada eslabón. Dichos *Marker*s quedarán como se muestra en las imágenes adjuntadas al principio de este apartado y tendrán las siguientes características, como se muestra en las figuras 20, 21 y 22:

Tabla 2: Localización y orientación de los *Markers* asignados a cada barra

Marker	Localización XYZ (m)	Orientación respecto base (grados)
Sistema 0	0,0,0.62357	0,0,0
Sistema 1	0,0,0.62357	90,0,-90
Sistema 2	0.4318,0.12954,0.62357	90,0,-90
Sistema 3	0.41148,0.12954,0.62357	0,0,90
Sistema 4	0.41148,0.12954,1.05566	90,0,-90
Sistema 5	0.41148,0.12954,1.05566	0,0,90
Sistema 6	0.41148,0.12954,1.11192	0,0,90

Una vez definidos los *Marker*s correspondientes a los sistemas de referencia de DH, se colocarán sobre ellos los pares de revolución correspondientes a cada una de las articulaciones. Estos pares se situarán en el mismo punto que los *Marker*s seleccionando primero el cuerpo conducido y luego el conductor.

Estas uniones, tendrán las siguientes características: (extraídas del modelo de ADAMS View):

```
Object Name      :  .PrototipoPUMA_bueno.Union_fija
Object Type      :  Fixed Joint
Parent Type      :  Model
Adams ID         :  1
Active           :  NO_OPINION
I Marker           : .PrototipoPUMA_bueno.ground.Par_fijo_suelo
J Marker           : .PrototipoPUMA_bueno.Base_fija.Par_fijo


Object Name      :  .PrototipoPUMA_bueno.Articulacion_1
Object Type      :  Revolute Joint
Parent Type      :  Model
Adams ID         :  2
Active           :  NO_OPINION
I Marker           : .PrototipoPUMA_bueno.Hombro.Eje1_hombro
J Marker           : .PrototipoPUMA_bueno.Base_fija.Eje1_base
Initial Conditions
  Angular Displacement : NOT SET
  Angular Velocity     : NOT SET


Object Name      :  .PrototipoPUMA_bueno.Articulacion_2
Object Type      :  Revolute Joint
Parent Type      :  Model
Adams ID         :  3
Active           :  NO_OPINION
I Marker           : .PrototipoPUMA_bueno.Brazo.Eje2_brazo
J Marker           : .PrototipoPUMA_bueno.Hombro.Eje2_hombro
Initial Conditions
  Angular Displacement : NOT SET
  Angular Velocity     : NOT SET


Object Name      :  .PrototipoPUMA_bueno.Articulacion_3
Object Type      :  Revolute Joint
Parent Type      :  Model
Adams ID         :  4
Active           :  NO_OPINION
I Marker           : .PrototipoPUMA_bueno.Antebrazo.Eje3_antebrazo
J Marker           : .PrototipoPUMA_bueno.Brazo.Eje3_brazo
Initial Conditions
  Angular Displacement : NOT SET
  Angular Velocity     : NOT SET
```

```
Object Name      :  .PrototipoPUMA_bueno.Articulacion_4
Object Type      :  Revolute Joint
Parent Type      :  Model
Adams ID         :  5
Active           :  NO_OPINION
I Marker           : .PrototipoPUMA_bueno.Muneca.Eje4_muneca
J Marker           : .PrototipoPUMA_bueno.Antebrazo.Eje4_antebrazo
Initial Conditions
  Angular Displacement : NOT SET
  Angular Velocity     : NOT SET


Object Name      :  .PrototipoPUMA_bueno.Articulacion_5
Object Type      :  Revolute Joint
Parent Type      :  Model
Adams ID         :  6
Active           :  NO_OPINION
I Marker           : .PrototipoPUMA_bueno.Acople_muneca.Eje5_acople
J Marker           : .PrototipoPUMA_bueno.Muneca.Eje5_muneca
Initial Conditions
  Angular Displacement : NOT SET
  Angular Velocity     : NOT SET


Object Name      :  .PrototipoPUMA_bueno.Articulacion_6
Object Type      :  Revolute Joint
Parent Type      :  Model
Adams ID         :  7
Active           :  NO_OPINION
I Marker           : .PrototipoPUMA_bueno.Efector_final.Eje6_efector
J Marker           : .PrototipoPUMA_bueno.Acople_muneca.Eje6_acople
Initial Conditions
  Angular Displacement : NOT SET
  Angular Velocity     : NOT SET
```

Figura 23: Características de los pares en ADAMS View

Son todos pares de revolución menos el primero de ellos que se corresponde a una unión fija entre el robot y el suelo.

Una vez se tengan dichas uniones en el modelo, se les asignará un movimiento, para ello habrá que aplicar a cada uno de ellos un "rotational joint motion". Se seleccionará dicha opción y se hará clic sobre los pares ya definidos, de este modo sobre dichos pares se crearán los movimientos, a falta de darles las características que se desean.

7.4 Resolución Cinemática Directa

Para el estudio de la cinemática directa del robot PUMA 560, se detallarán las variables articulares que se desean tanto al principio como al final de la trayectoria, así como el tiempo de ejecución del movimiento y el número de Steps del mismo.

7.4.1 Trayectorias, velocidades y aceleraciones, Cinemática Directa

Con el fin de calcular las ecuaciones que deben seguir dichos *motions*, se ha creado una tabla Excel, donde se han colocado los parámetros restrictivos que se han visto en el apartado de trayectorias, así como la posición inicial y final del robot, para posteriormente introducirlos en MATLAB.

Tabla 3, Tabla 4 y Tabla 5: Parámetros generación de trayectorias

	tf	tb	tf-tb	V	qf rad	q0 rad	qf	q0	q mov
Eje 1	3	1	2	0.218166156	0.43633231	0	25	0	25
Eje 2	3	1	2	0.130899694	0.26179939	0	15	0	15
Eje 3	3	1	2	0.392699082	0.78539816	0	45	0	45
Eje 4	3	1	2	0	0	0	0	0	0
Eje 5	3	1	2	0.261799388	0.52359878	0	30	0	30
Eje 6	3	1	2	0	0	0	0	0	0

	Tramo Parab 1		Tramo recto		Tramo parab 2			1 der Parab 1	
	alfa	gamma	a	V	alfa	beta	gamma	alfa	gamma
Eje 1	0	0.10908308	-0.10908308	0.218166156	-0.54541539	0.65449847	-0.10908308	0	0.21816616
Eje 2	0	0.06544985	-0.06544985	0.130899694	-0.32724923	0.39269908	-0.06544985	0	0.13089969
Eje 3	0	0.19634954	-0.19634954	0.392699082	-0.9817477	1.17809725	-0.19634954	0	0.39269908
Eje 4	0	0	0	0	0	0	0	0	0
Eje 5	0	0.13089969	-0.13089969	0.261799388	-0.65449847	0.78539816	-0.13089969	0	0.26179939
Eje 6	0	0	0	0	0	0	0	0	0

1 der Recto	1 der Parab 2			2 der 1parab	2 der recto	2 der parab
V	alfa	beta	gamma	gamma	-	gamma
0.21816616	0	0.65449847	-0.21816616	0.21816616	0	-0.21816616
0.13089969	0	0.39269908	-0.13089969	0.13089969	0	-0.13089969
0.39269908	0	1.17809725	-0.39269908	0.39269908	0	-0.39269908
0	0	0	0	0	0	0
0.26179939	0	0.78539816	-0.26179939	0.26179939	0	-0.26179939
0	0	0	0	0	0	0

Este Excel creado con las fórmulas explicadas en el apartado de teoría de trayectorias, proporciona los coeficientes de las ecuaciones de cada tramo de los movimientos de los nudos. tanto de los tramos rectos como de los tramos parabólicos para cada eje. El robot se ha modelado para que realice un movimiento que cambie sus variables articulares de q_0 a q_f (en

grados), como se puede apreciar en la tabla, y que realice dicho movimiento en un tiempo de 3 segundos con 80 steps.

A continuación, se usarán estas ecuaciones en ADAMS View para describir la función del movimiento que ha de producir cada motion. Para ello se pulsará sobre cada motion y se seleccionará "Modify" y en apartado de definición de la función se escribirán las funciones de los movimientos con sus correspondientes condiciones, es decir que se ejecute cada tramo dentro de su intervalo de tiempo. Como ejemplo se observa en la Figura 24 la función del movimiento del primer motion.

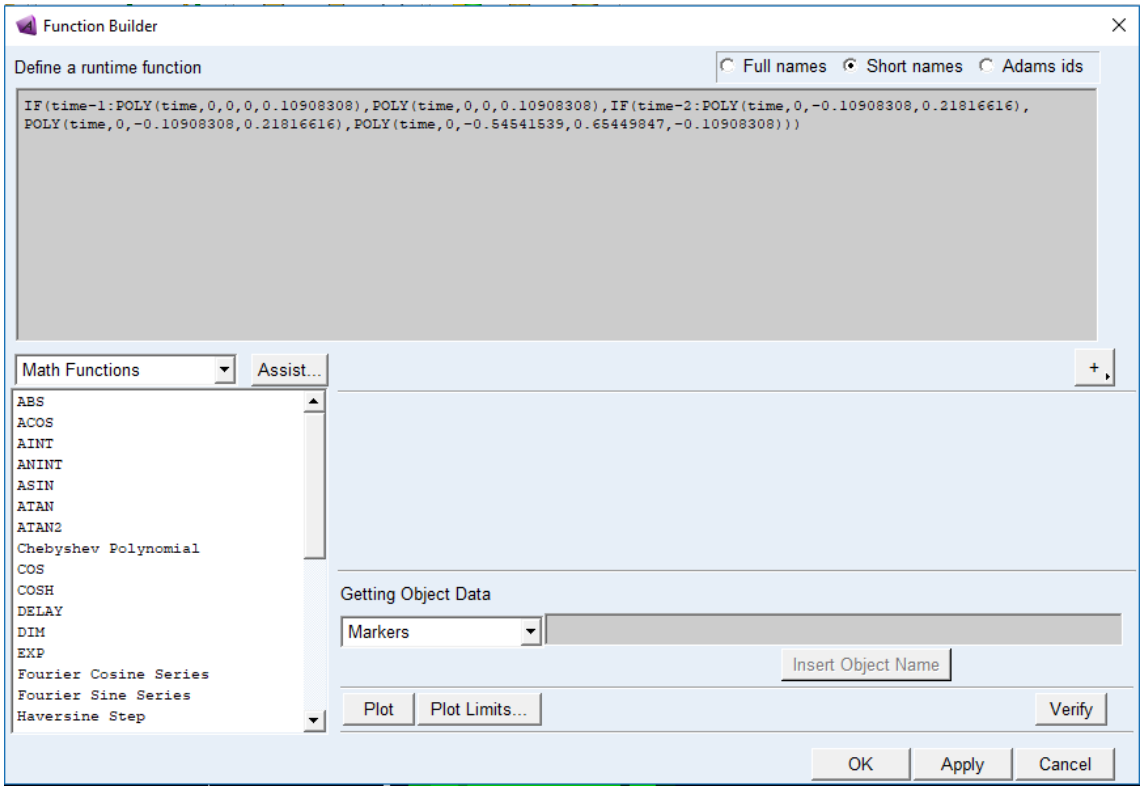


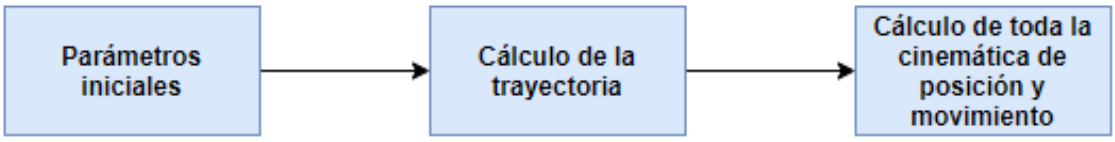
Figura 24: Función del primer motion en ADAMS View

Como se puede observar se trata de una función cuya variable es el tiempo, es decir evolucionará conforme dicha variable vaya cambiando. Se ha utilizado funciones "IF", para definir las diferentes partes de evolución del movimiento, las cuales tienen distintas funciones, tanto los tramos parabólicos como los rectos.

Cabe señalar también que, para mejorar la presentación de dicha función, se ha añadido la función "POLY", la cual sirve para representar de forma abreviada un polinomio, en este caso puede llegar a ser hasta de tercer orden. La sintaxis de dicha función permite poner en primer lugar el desfase, en segundo lugar, el término independiente, en tercer lugar, el coeficiente lineal y a partir de aquí permite poner los coeficientes cuadráticos y así sucesivamente, quedando las ecuaciones como se muestra en la Figura 24.

Los otros 6 movimientos se definirán de manera análoga a este, solamente variarán los coeficientes de las funciones y las condiciones de tiempo en que se ejecutan cada tramo de movimiento.

7.4.2 Resolución Cinemática Directa en MATLAB



- Parámetros iniciales: En el script "Cinematicadirectaentera.m", se han introducido previamente todos los parámetros correspondientes al método de Denavit y Hartenberg, así como todas las distancias de las barras. En este fichero también se han introducido las variables articulares Theta en la posición inicial y en la posición final que se desea que alcance el brazo robótico. Por último, también se ha introducido en este como dato importante el tiempo en el cual se desea que realice dicho movimiento y el incremento de tiempo, el cual es importante a la hora de calcular la trayectoria. Este fichero, también agrupa todas las funciones que se encargan de la resolución de la cinemática directa y que serán presentadas a continuación.

- Cálculo de la trayectoria: ''nuevagentray" es una función que recibe como input el tiempo total y el incremento de tiempo que se desea para el movimiento. Esta función tiene definidas las ecuaciones del movimiento de cada tramo según evoluciona el tiempo. "nuevagentray" devolverá una matriz de tantas filas como saltos temporales se realicen con el incremento de tiempo establecido hasta alcanzar el tiempo final, y con tantas columnas como variables articulares haya, en el caso del PUMA hay seis. Esta función será llamada dentro de "Cinematicadirectaentera.m".

- Cálculo de toda la cinemática de posición y movimiento: se trata de la función "jacobcalculo", la cual recibe las variables articulares en su posición inicial, las velocidades y aceleraciones extraídas de la función de generación de trayectorias y por último recibe también las variables articulares en su posición final. Dicha función es la que permite resolver al completo la cinemática directa, devolviendo como resultado las coordenadas de posición del efector final, sus ángulos de Euler y las velocidades y aceleraciones de las seis articulaciones. Dicha función también se encuentra incluida en "Cinematicadirectaentera.m".

A continuación, se explicará el funcionamiento del archivo "Cinematicadirectaentera.m", así como todas las funciones que lo integran y donde se resuelve el problema cinemático directo.

- "Cinematicadirectaentera.m":

```
%El siguiente programa resuelve la cinemática directa completa del robot
%PUMA 560 dado la matriz DH del robot y los parámetros Theta iniciales y
%finales, así como el tiempo del movimiento y el incremento del mismo.
%--------------------------------------------------------------------------

%Primer paso: añadir la matriz con
%los parámetros de DH y las longitudes de todas las barras del robot.

%  alpha    a     d    theta tipo  xi  yi  zi  xf  yf  zf    R   G   B
PUMA=[ 0       0    0     0     0    0   0   0   0   0 -623.57 1   0   1;
   -pi/2     0    0     0     0    0   0   0   0   0  129.54 0   1   1;
     0     431.8 129.54 0     0    0   0   0 -431.8 0   0    0   0  0.9;
   pi/2   -20.32  0     0     0    0   0   0  20.32 0   0    0   1   0;
  -pi/2      0   432.09 0     0    0   0   0   0 432.09 0   1    0   0;
   pi/2      0    0     0     0    0   0   0   0   0    0  0.5  0.5  0;
     0       0   56.26  0     0    0   0   0   0   0 -56.26 0.4 0.8 0.8];
 %--------------------------------------------------------------------------
 %Segundo paso: llamada de la funcion "nuevagentray" la cual me devolverá
 %el ángulo en radianes de cada uno de los motores del robot,
 %así como la velocidad y aceleración de cada uno de ellos durante el
 %intervalo de tiempo especificado y para las posiciones deseadas.
 %Esta parte me devolverá una matriz de 18 columnas y tantas filas como
 %saltos de incremento de tiempo hayan hasta que se cumpla el tiempo final
 %del movimiento.

thetasinigrad=[0 0 0 0 0 0];
thetasfingrad=[25 15 45 0 30 0];
thetaini=gradrad(thetasinigrad);
thetafin=gradrad(thetasfingrad);
tfin=3;
inct=0.0375;
[trayvelacel]=nuevagentray(tfin,inct);
 %--------------------------------------------------------------------------
%A continuación se iniciará un bucle for que se repita tanto como tantas
%filas tenga "trayvelacel" y se aplicará la función "jacobcalculo" para
%resolver la cinemática directa.
tamano=size(trayvelacel);
Cinematicaresultado=[];
for i=1:tamano(1)
    Cinematicaresultado=[Cinematicaresultado;jacobcalculo(thetaini,...
     trayvelacel(i,7:12),trayvelacel(i,13:18),...
     trayvelacel(i,1:6))];
end
Cinematicaresultado;
```

- "Nuevagentray.m"

```
function [Output]=nuevagentray(tiempo,inc)
contador=1;
%Valores de todos los coeficientes de la trayectoria en una misma matriz

%             tr par 1     /          tr rec           /
%            alf   gamma          a              V           alf
coeficientes=[0 0.109083078 -0.109083078    0.218166156 -0.545415391
              0 0.065449847 -0.065449847    0.130899694 -0.327249235
              0 0.196349541 -0.196349541    0.392699082 -0.981747704
              0      0             0             0             0
              0 0.130899694 -0.130899694    0.261799388 -0.654498469
              0      0             0             0             0

  tr par 2           /    der tr par 1 /  der tr rec  /      der tr par 2
 beta        gamma       alfa    gamma    a        V     alfa     beta        gamma
0.654498469   -0.109083078    0   0.218166156 0   0.218166156 0   0.654498469 -0.218166156
0.392699082   -0.065449847    0   0.130899694 0   0.130899694 0   0.392699082 -0.130899694
1.178097245   -0.196349541    0   0.392699082 0   0.392699082 0   1.178097245 -0.392699082
  0           0           0       0       0       0       0       0           0
0.785398163   -0.130899694    0   0.261799388 0   0.261799388 0   0.785398163 -0.261799388
  0           0           0       0       0       0       0       0           0


 /   der 2trpar/ der2 tr rec/ der 2 tr par 2
gamma       -       gamma
   0.218166156 0    -0.218166156;
   0.130899694 0    -0.130899694;
   0.392699082 0    -0.392699082;
   0        0        0;
   0.261799388 0    -0.261799388;
   0        0        0];

%-------------------------------------------------------------------------
%a continuacion se ejecutará un bucle for con todas las ecuaciones del
%movimiento calculadas para los diferentes tramos, segun se encuentre en
%uno o en otro el resultado será diferente y con ello se generará la
%trayectoria
for i=inc:inc:tiempo
    for j=1:6 %los ejes
        if i<1
            Output(contador,j)=coeficientes(j,2)*i*i;
            Output(contador,j+6)=2*coeficientes(j,2)*i;
            Output(contador,j+12)=2*coeficientes(j,2);
        elseif i==1
            Output(contador,j)=coeficientes(j,2)*i;
            Output(contador,j+6)=coeficientes(j,2);
            Output(contador,j+12)=0;
        elseif i<2
            Output(contador,j)=coeficientes(j,3)+coeficientes(j,4)*i;
            Output(contador,j+6)=coeficientes(j,4);
            Output(contador,j+12)=0;
        elseif i==2
            Output(contador,j)= coeficientes(j,3)+coeficientes(j,4)*i;
            Output(contador,j+6)=coeficientes(j,4);
            Output(contador,j+12)=0;
```

```
        else
            Output(contador,j)=coeficientes(j,5)+coeficientes(j,6)*i+coeficientes(j,7)*i*i;
            Output(contador,j+6)=coeficientes(j,6)+2*coeficientes(j,7)*i;
            Output(contador,j+12)=2*coeficientes(j,7);
        end
    end
    contador=contador+1;
end
end
 %Esta función generará una matriz de seis columnas y tf/inc filas
```

- "Jacobcalculo.m"

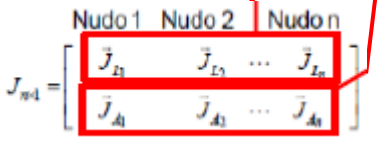
```
%Matriz jacobiana mediante metodo de Whitney
function output=jacobcalculo(thetin,velocidades,aceleraciones,thetasfin)

%trabajo en radianes y milimetros
%parametros DH del robot
q=sym('q',[1 6]);
qdh=thetin;
d=[0 129.54 0 432.09 0 56.26];
a=[0 431.8 -20.32 0 0 0];
alfa=[-pi/2 0 pi/2 -pi/2 pi/2 0];
%saco las matrices de transformción homogenea entre sistemas
%de coordenadas consecutivos
A01 = denavit(q(1), d(1), a(1), alfa(1));
A12 = denavit(q(2), d(2), a(2), alfa(2));
A23 = denavit(q(3), d(3), a(3), alfa(3));
A34 = denavit(q(4), d(4), a(4), alfa(4));
A45 = denavit(q(5), d(5), a(5), alfa(5));
A56 = denavit(q(6), d(6), a(6), alfa(6));
%matriz de transformación del primer al ultimo sistema:
A06= A01 * A12 * A23 * A34 * A45 * A56;
%saco de las matrices homogeneas las de rotación
R01 = [A01(1,1) A01(1,2) A01(1,3);A01(2,1) A01(2,2) A01(2,3);A01(3,1) A01(3,2) A01(3,3)];
R12 = [A12(1,1) A12(1,2) A12(1,3);A12(2,1) A12(2,2) A12(2,3);A12(3,1) A12(3,2) A12(3,3)];
R23 = [A23(1,1) A23(1,2) A23(1,3);A23(2,1) A23(2,2) A23(2,3);A23(3,1) A23(3,2) A23(3,3)];
R34 = [A34(1,1) A34(1,2) A34(1,3);A34(2,1) A34(2,2) A34(2,3);A34(3,1) A34(3,2) A34(3,3)];
R45 = [A45(1,1) A45(1,2) A45(1,3);A45(2,1) A45(2,2) A45(2,3);A45(3,1) A45(3,2) A45(3,3)];
R56 = [A56(1,1) A56(1,2) A56(1,3);A56(2,1) A56(2,2) A56(2,3);A56(3,1) A56(3,2) A56(3,3)];
%la parte de debajo de la Jacobiana sera:
z=[0;0;1];
%aplico el metodo de whitney:
JR1=z;
JR2=R01*z;
JR3=R01*R12*z;
JR4=R01*R12*R23*z;
JR5=R01*R12*R23*R34*z;
JR6=R01*R12*R23*R34*R45*z;

%a continuación saco por diferenciación la parte de arriba de
%la jacobiana y monto la matriz
Posicion=[A06(1,4);A06(2,4);A06(3,4)];
Jacobiano=[jacobian(Posicion,q);JR1 JR2 JR3 JR4 JR5 JR6];
```

$$T = {}^{0}A_1(q_1)\,{}^{1}A_2(q_2)\cdots{}^{n-1}A_n(q_n)$$

$${}^{i-1}\vec{z}_{i-1} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \Rightarrow {}^{0}\vec{z}_{i-1} = {}^{0}R_1(q_1)\cdot{}^{1}R_2(q_2)\cdot\ldots\cdot{}^{i-2}R_{i-1}(q_{i-1})\cdot{}^{i-1}\vec{z}_{i-1}$$



```
%qdh, es el parametro theta, qa es el desplazamiento
%sustituyo y obtengo numericamente la matriz jacobiana:
qa=thetasfin;
qs=(qdh+qa);

PosDenavit=double(subs(A06,q,qs));%mat homogenea de transf
JacobianoDouble=double(subs(Jacobiano,q,qs));

%Meto la velocidad en radianes luego
%multiplico la Jacobiana por las velocidades angulares.
va=velocidades;
VAR= transpose(va);

Vel=JacobianoDouble*VAR;
aa =[aceleraciones];
%a continuacion saco los angulos de euler segun las formulas
xteta=sqrt(A06(3,1)^2+A06(3,2)^2);
yteta=A06(3,3);
tetapos =double(subs(atan2(xteta,yteta),q,qs));
tetaneg =double(subs(atan2(-xteta,yteta),q,qs));
psipos=double(subs(atan2(A06(1,3)/sin(tetapos),-A06(2,3)/sin(tetapos)),q,qs));
psineg=double(subs(atan2(A06(1,3)/sin(tetaneg),-A06(2,3)/sin(tetaneg)),q,qs));
phipos=double(subs(atan2(A06(3,1)/sin(tetapos),A06(3,2)/sin(tetapos)),q,qs));
phineg=double(subs(atan2(A06(3,1)/sin(tetaneg),A06(3,2)/sin(tetaneg)),q,qs));

% Primero saco la derivada de la matriz jacobiana(jacopunto)
jacopunto= diff(Jacobiano,q(1),1)*va(1)+ diff(Jacobiano,q(2),1)*va(2)...
    +diff(Jacobiano,q(3),1)*va(3)+diff(Jacobiano,q(4),1)*va(4)+diff(Jacobiano,...
    q(5),1)*va(5)+diff(Jacobiano,q(6),1)*va(6);
%a continuacion saco las aceleraciones:
Acel = double(subs(jacopunto,q,qs))*transpose(va)+double(subs(Jacobiano,q,qs))...
    *transpose(aa);

output=[PosDenavit(1,4),PosDenavit(2,4),PosDenavit(3,4),tetapos,psipos,...
    phipos,tetaneg,psineg,phineg,Vel(1),Vel(2),Vel(3),Vel(4),Vel(5),Vel(6),...
    Acel(1),Acel(2),Acel(3),Acel(4),Acel(5),Acel(6)];
end
```

$$\vec{p} = J \cdot \vec{q}$$

$$\theta = \operatorname{atan2}\left(\sqrt{r_{31}^2 + r_{32}^2}, r_{33}\right)$$

$$\phi = \operatorname{atan2}\left(\frac{r_{13}}{\operatorname{sen}(\theta)}, -\frac{r_{23}}{\operatorname{sen}(\theta)}\right)$$

$$\psi = \operatorname{atan2}\left(\frac{r_{31}}{\operatorname{sen}(\theta)}, \frac{r_{32}}{\operatorname{sen}(\theta)}\right)$$

$$\dot{J} = \frac{\partial J}{\partial q_1} \cdot \dot{q}_1 + \frac{\partial J}{\partial q_2} \cdot \dot{q}_2 + \ldots + \frac{\partial J}{\partial q_n} \cdot \dot{q}_n$$

$$\ddot{\vec{p}} = \dot{J} \cdot \dot{\vec{q}} + J \cdot \ddot{\vec{q}}$$

- "gradrad.m" (usada en "Cinematicadirectaentera.m")

```
%recibe el ángulo en grados y lo transforma en radianes
function [resul]=gradrad(ang)

    i=1;
    for cont=1:length(ang)

        resul(i)=ang(i)*pi/180;
        i=i+1;
    end
```

7.4.3 Resultados de la Cinemática Directa

Una vez se tiene el código en MATLAB y los resultados de la simulación realizada con ADAMS View y MATLAB para un caso en concreto, se va a proceder a comparar los resultados obtenidos por ambos métodos para ver que coincidan y asegurar que la modelización cinemática es correcta.

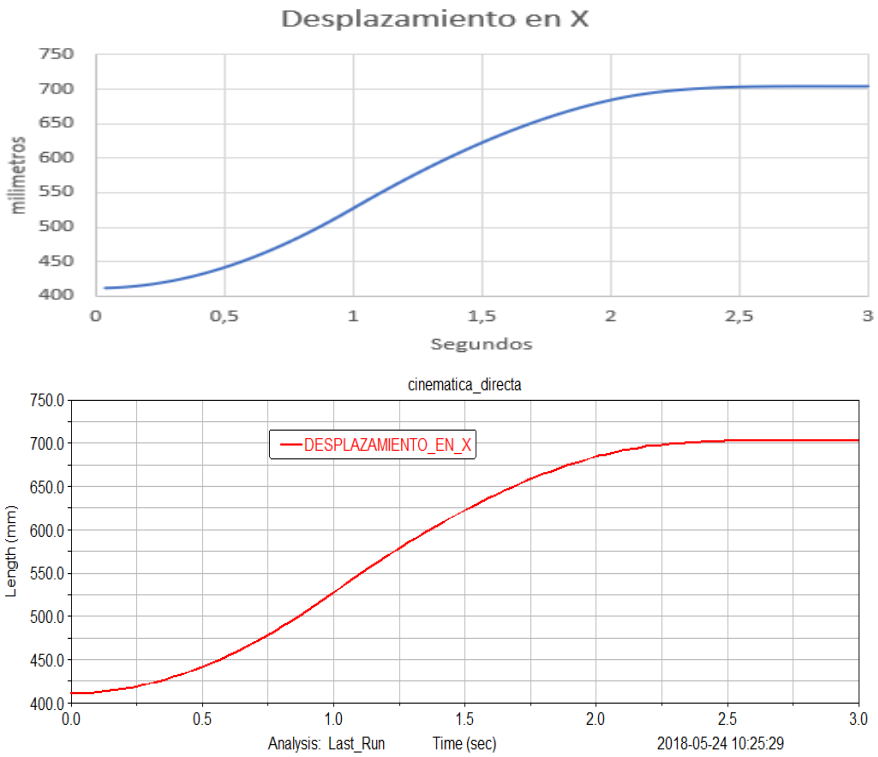


Figura 25: Desplazamiento en X MATLAB y ADAMS View

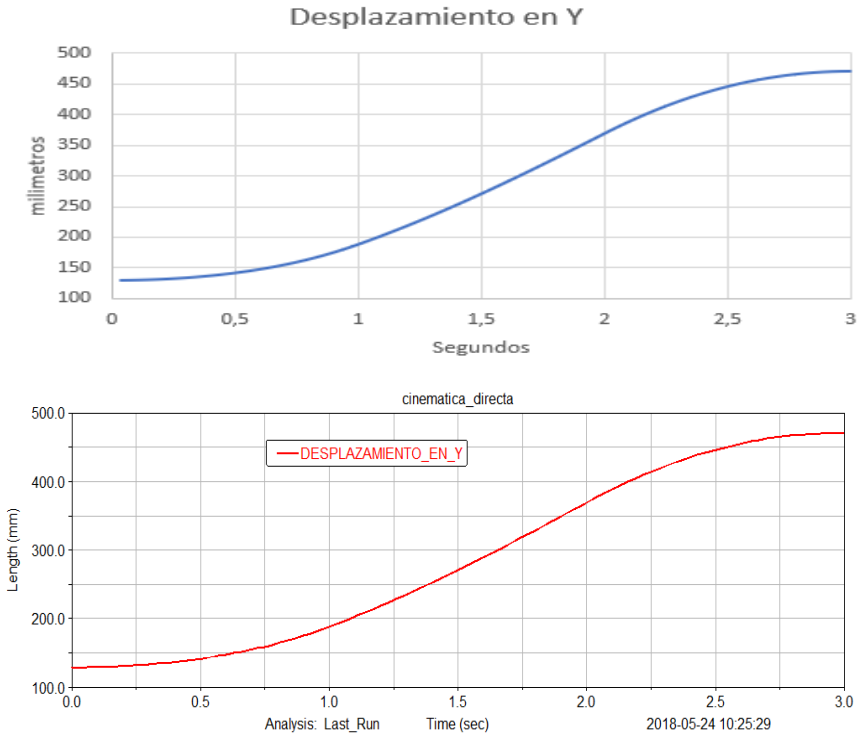


Figura 26: Desplazamiento en Y MATLAB y ADAMS View

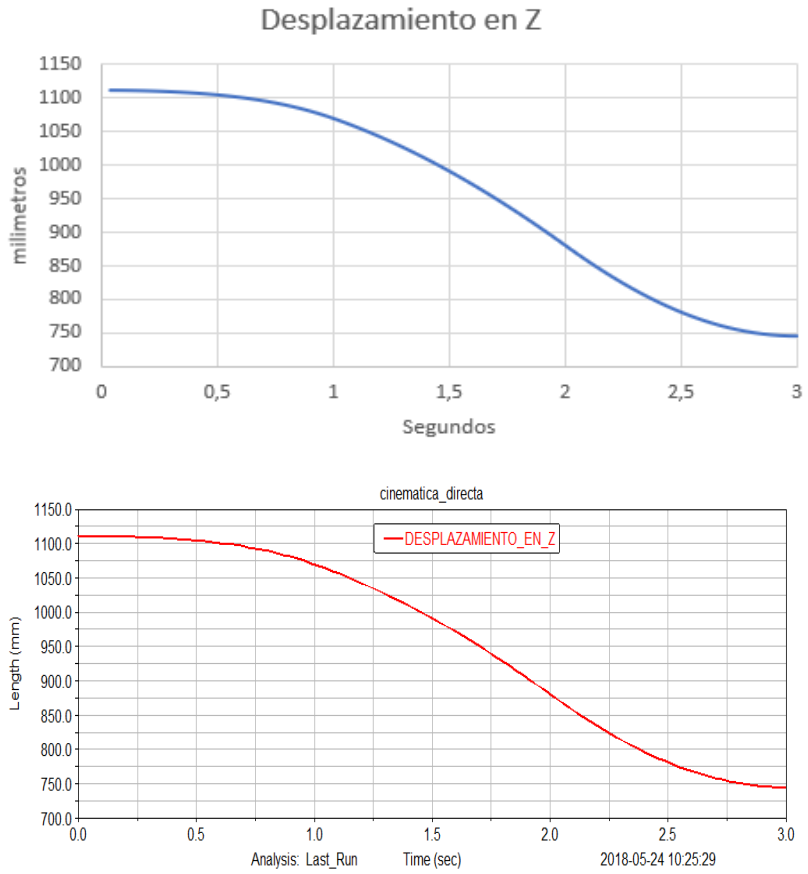


Figura 27: Desplazamiento en Z MATLAB y ADAMS View

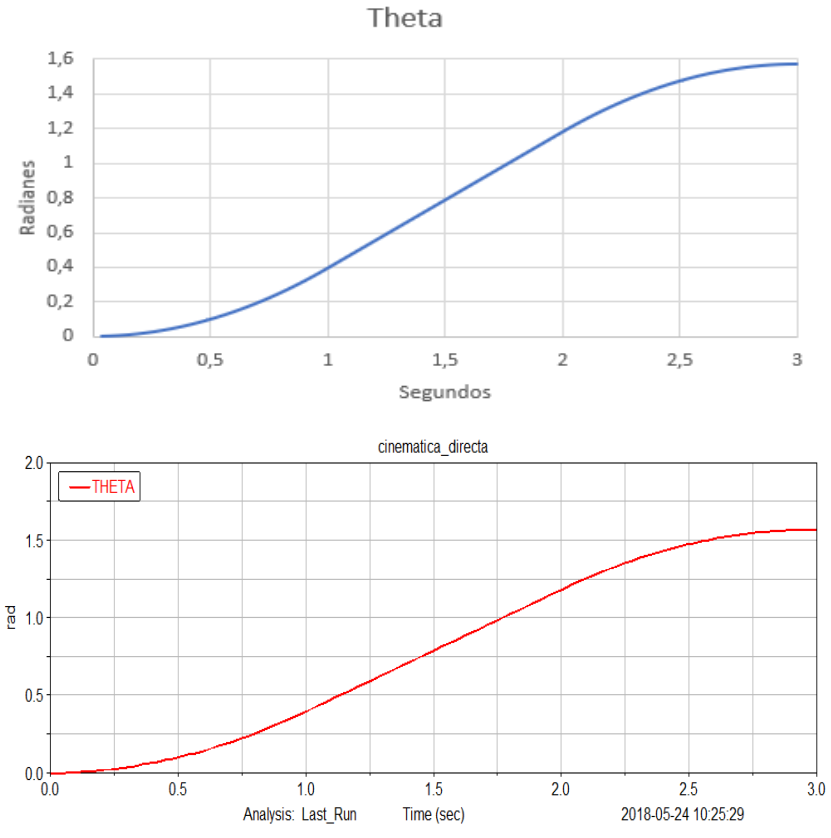


Figura 28: Ángulo Theta MATLAB y ADAMS View

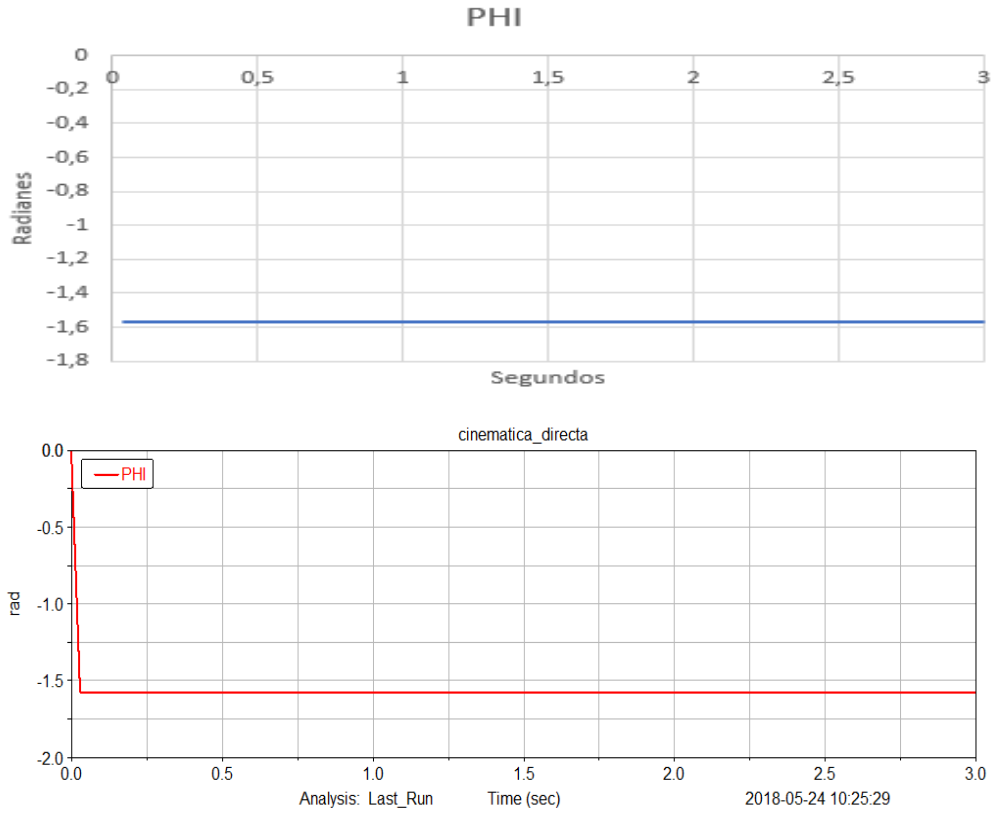


Figura 29: Ángulo PHI MATLAB y ADAMS View

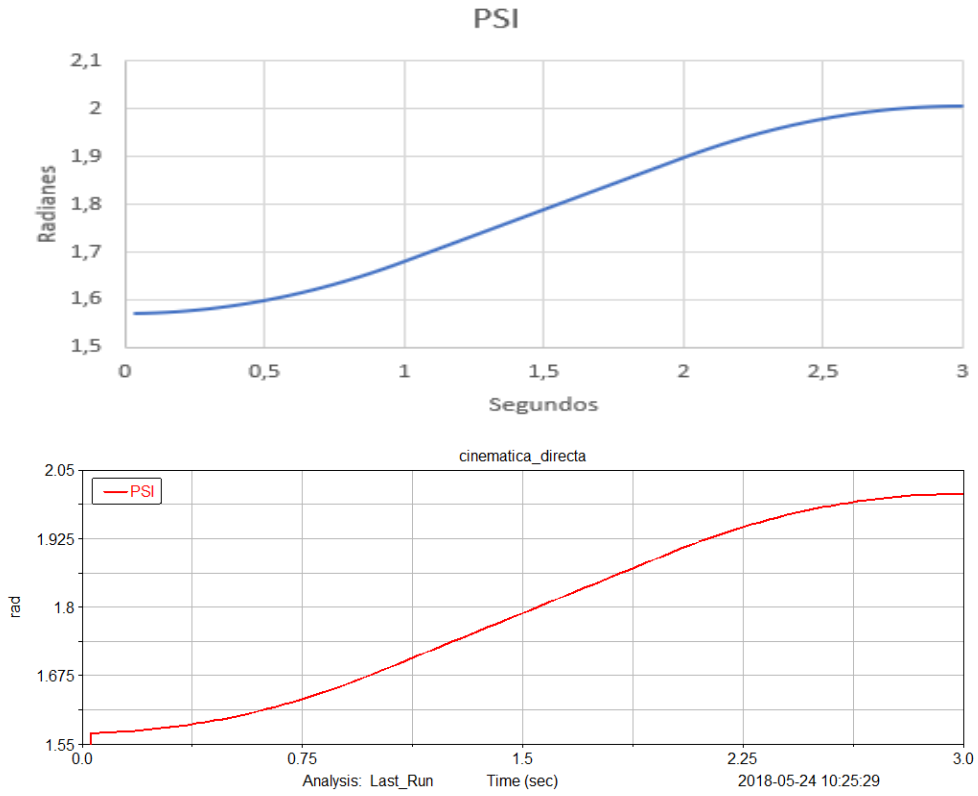


Figura 30: Ángulo PSI MATLAB y ADAMS View

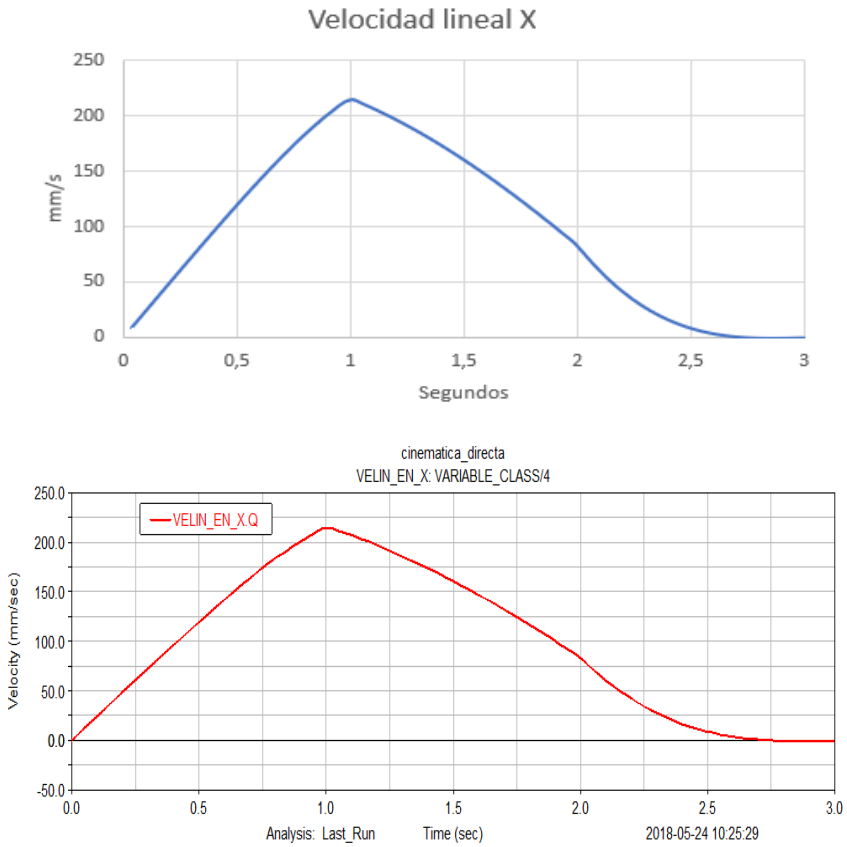


Figura 31: Velocidad lineal en X MATLAB y ADAMS View

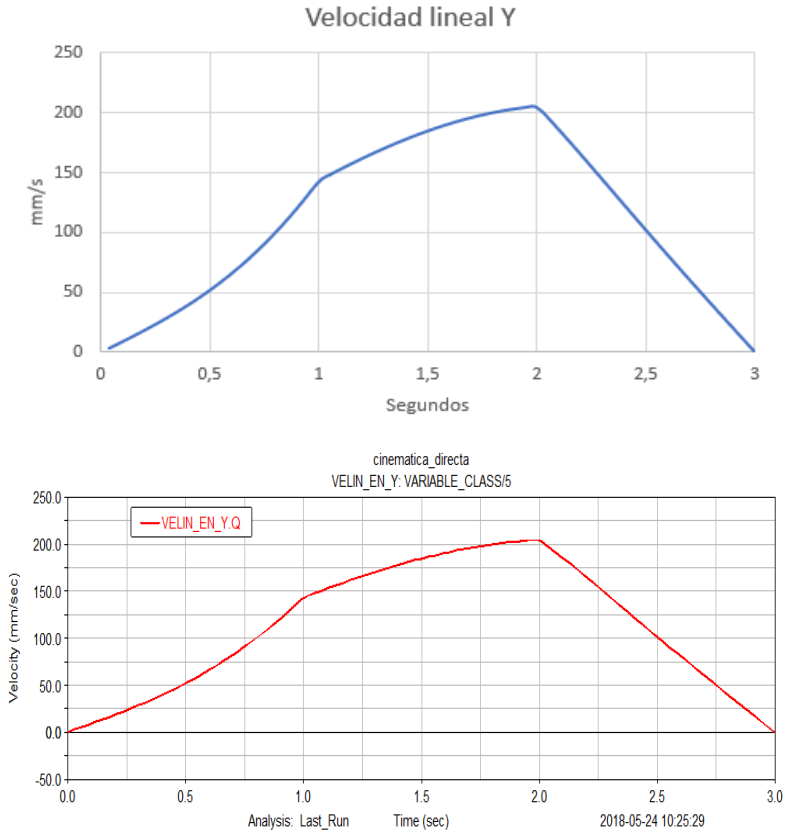


Figura 32: Velocidad lineal en Y MATLAB y ADAMS View

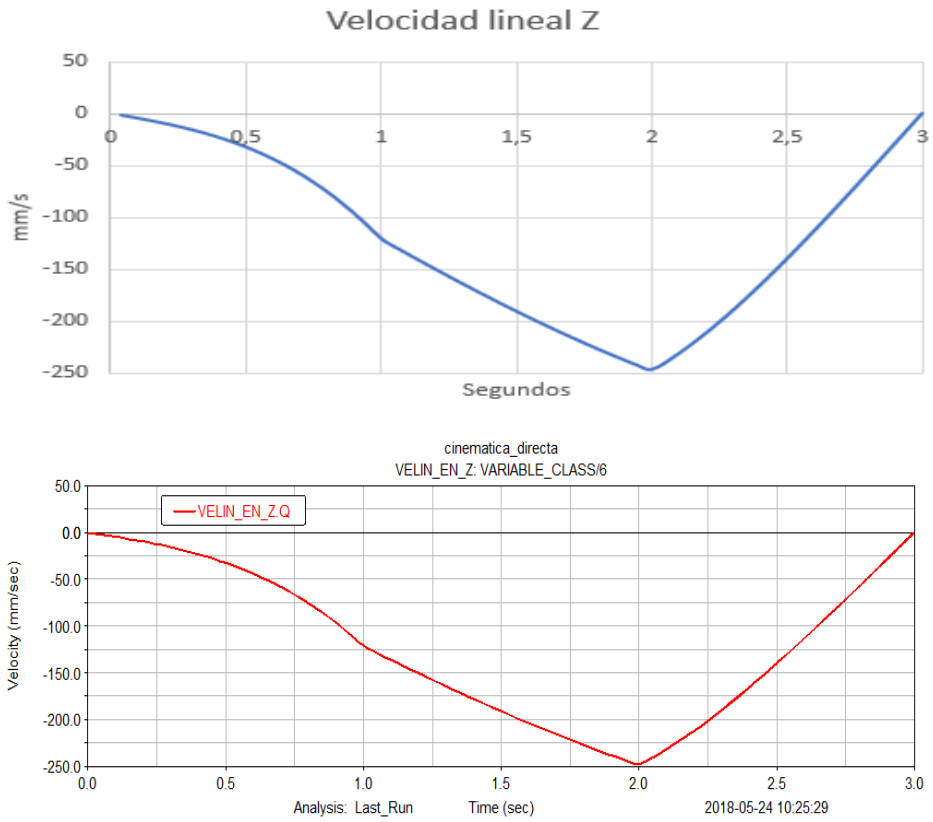


Figura 33: Velocidad lineal en Z MATLAB y ADAMS View

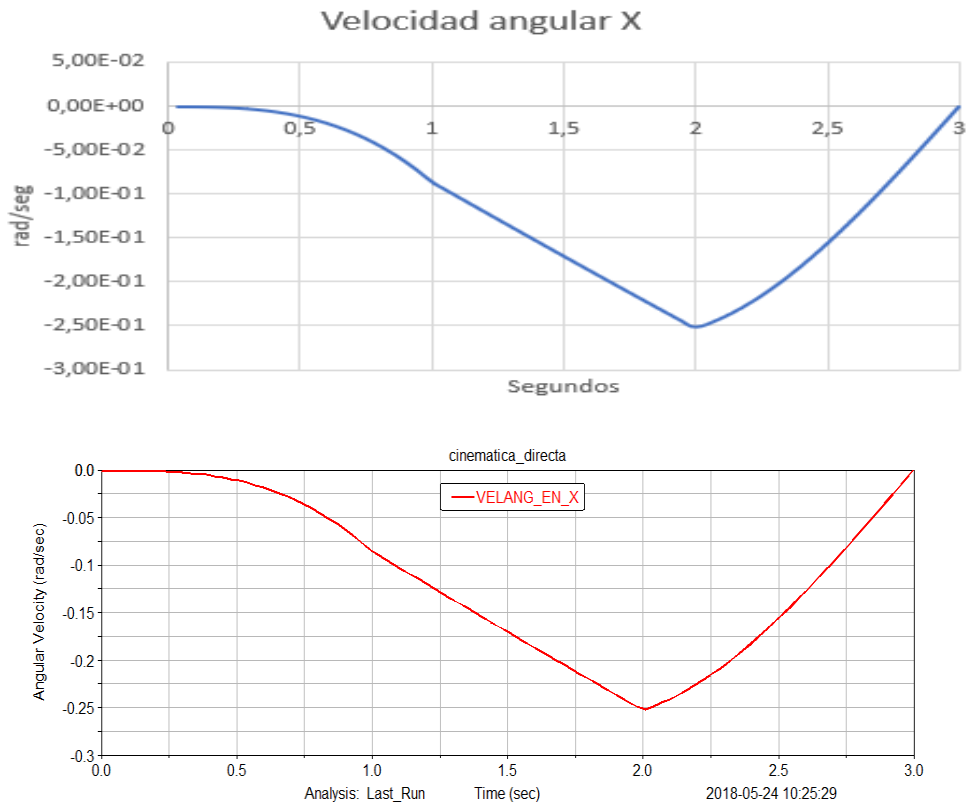


Figura 34: Velocidad angular en X MATLAB y ADAMS View

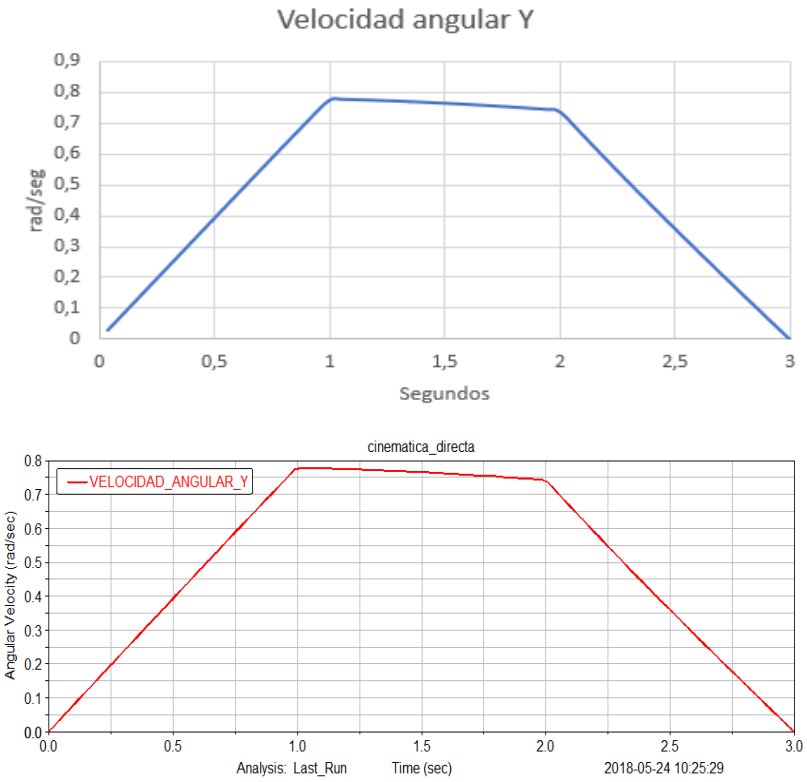


Figura 35: Velocidad angular en Y MATLAB y ADAMS View

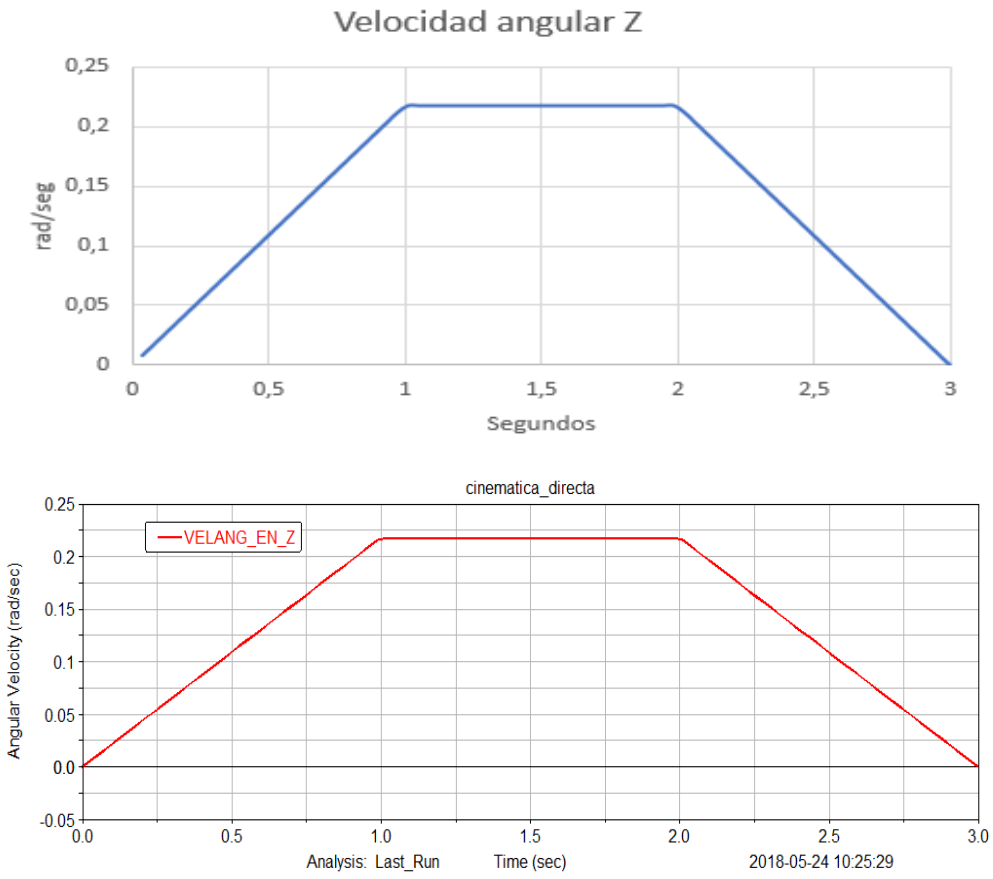


Figura 36: Velocidad angular en Z MATLAB y ADAMS View

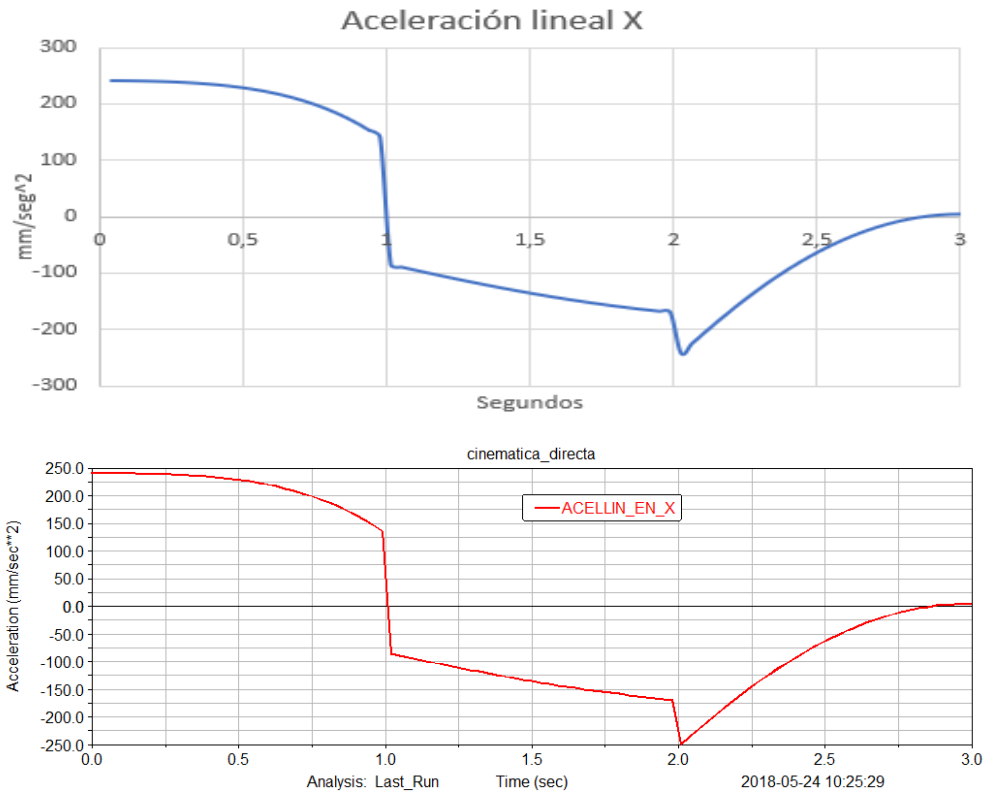


Figura 37: Aceleración linear en X MATLAB y ADAMS View

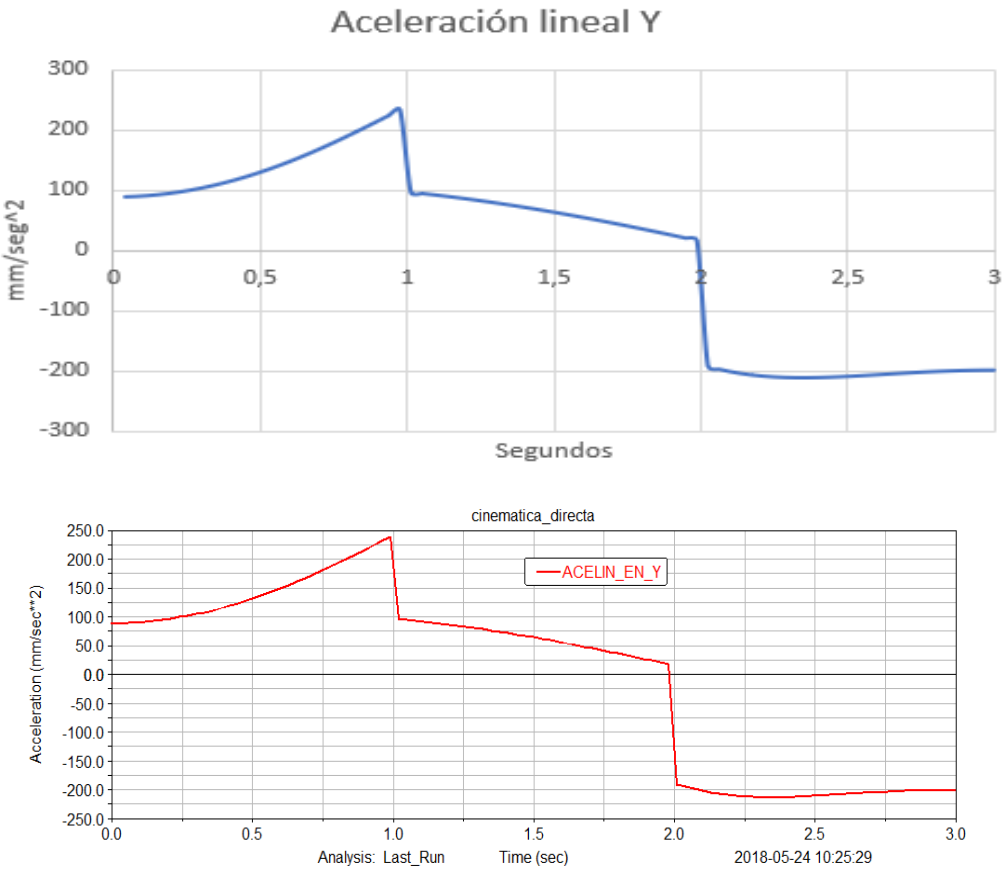


Figura 38: Aceleración lineal en Y MATLAB y ADAMS View

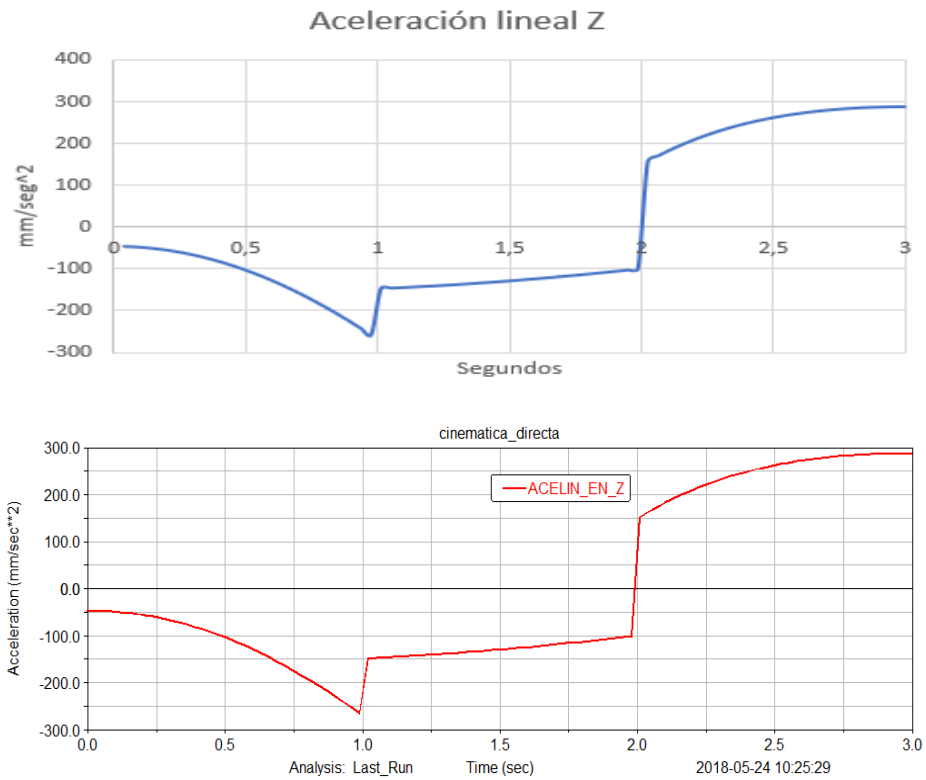


Figura 39: Aceleración lineal en Z MATLAB y ADAMS View

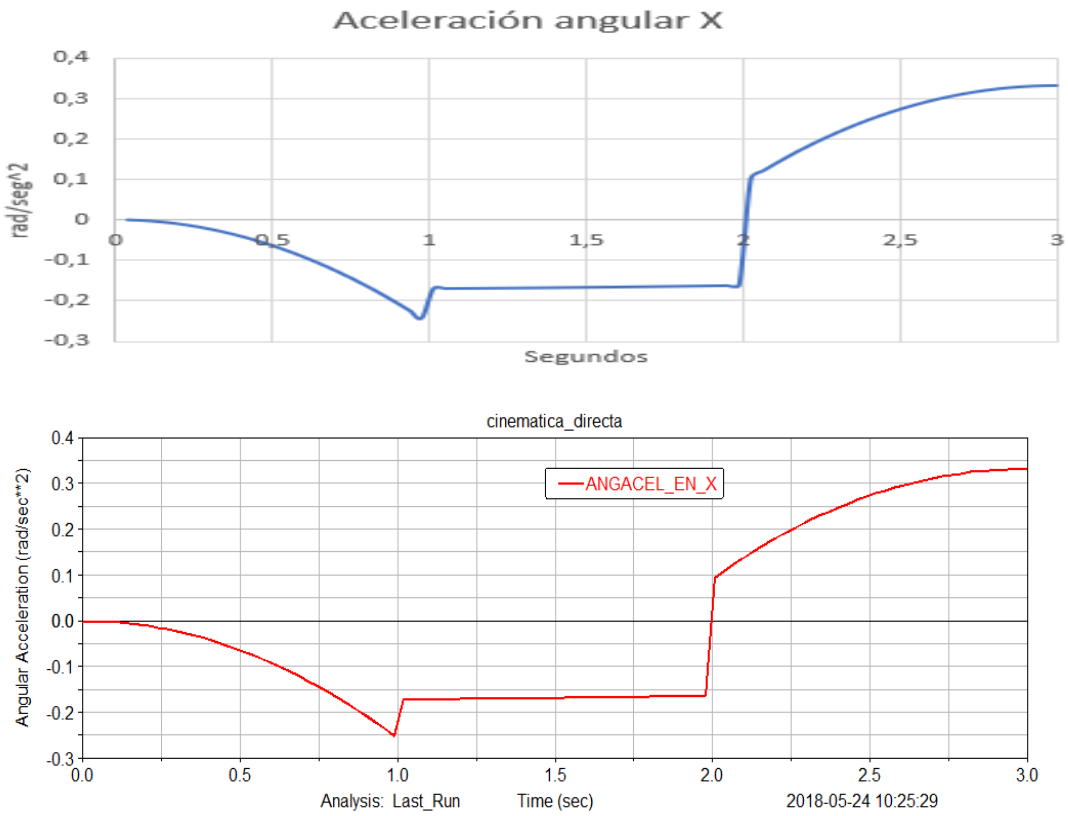


Figura 40: Aceleración angular en X MATLAB y ADAMS View

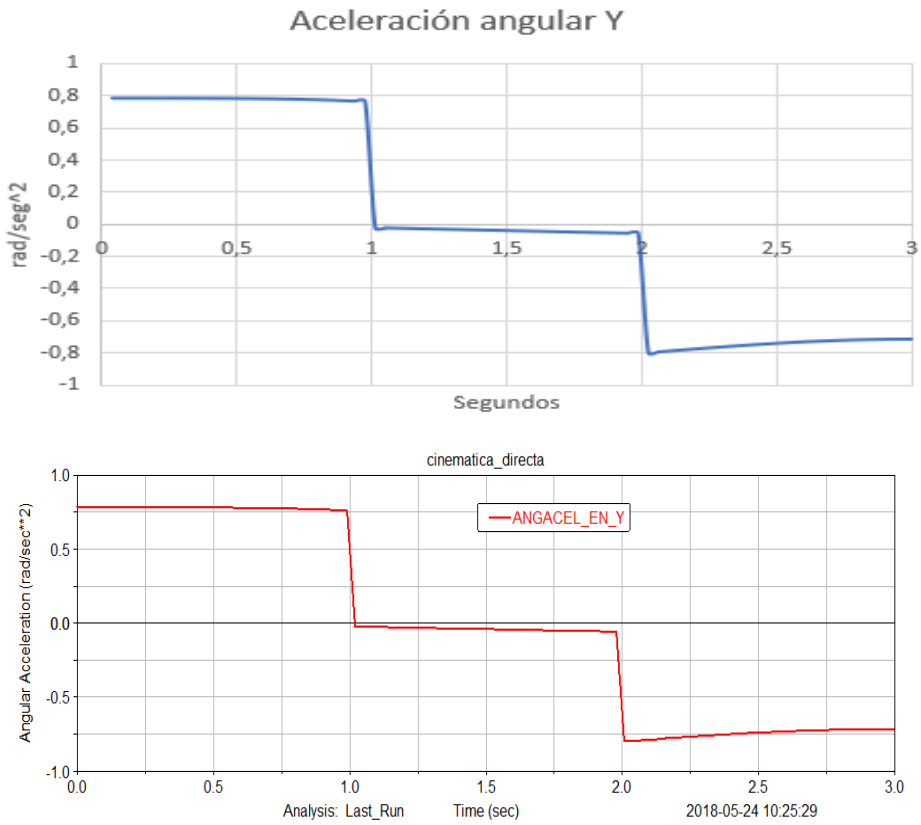


Figura 41: Aceleración angular en Y MATLAB y ADAMS View

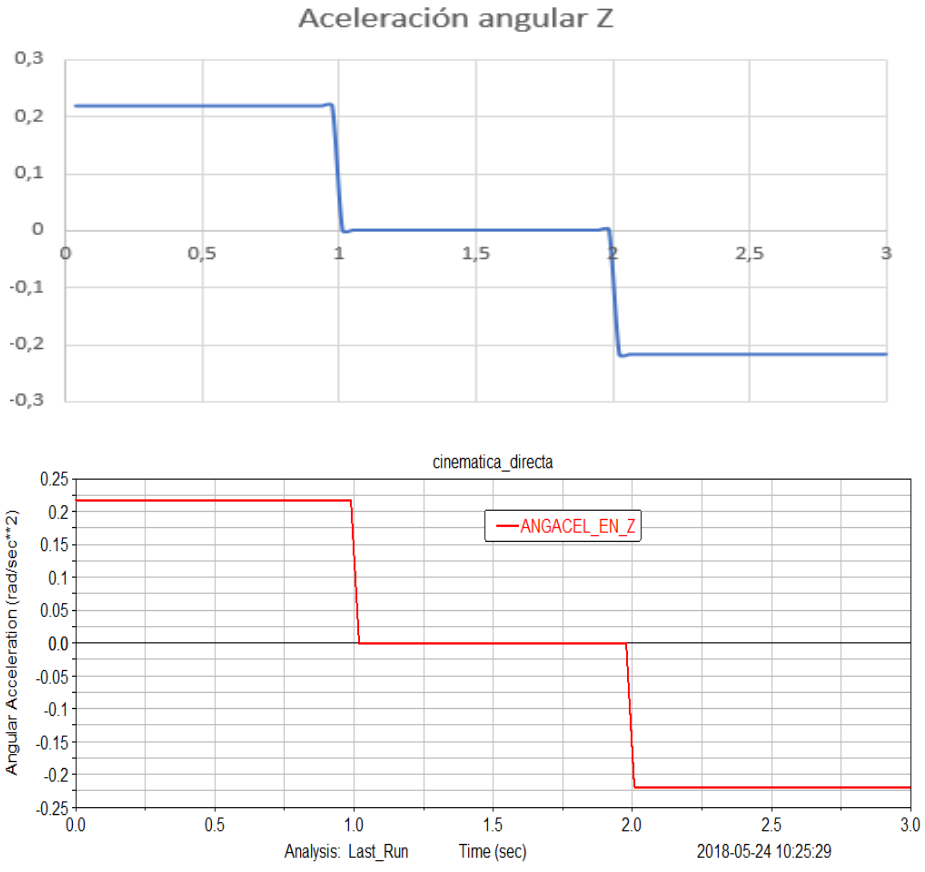


Figura 42: Aceleración angular en Z MATLAB y ADAMS View

Como se puede observar, los resultados tanto de la simulación con ADAMS View como los del modelado matemático con MATLAB del robot PUMA 560, son coincidentes.

Se ha realizado exitosamente la simulación y el modelado de las coordenadas X, Y, Z del efector final, los ángulos de Euler, las velocidades y aceleraciones tanto traslacionales como angulares.

Además, el código de MATLAB puede obtener las posiciones de los nudos y la segunda terna de ángulos de Euler para cada instante y para cualquier combinación de variables articulares deseada.

En cuanto a las unidades de medida de ADAMS View se han usado las que se muestran en la siguiente Figura 43:

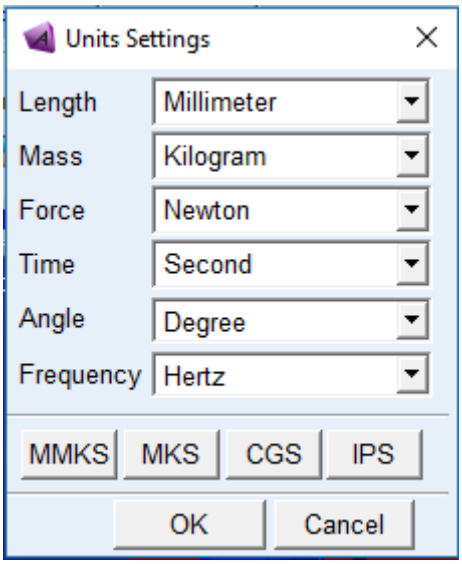
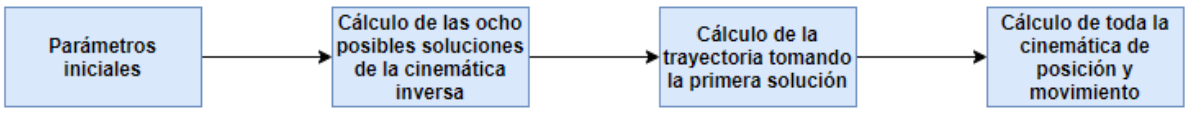


Figura 43: Unidades de medida utilizadas

7.5 Resolución Cinemática Inversa

Para el caso de estudio de la cinemática inversa del robot puma 560, se detallará una posición final deseada para el efector final, así como su orientación. Dicha información se especificará en forma de matriz homogénea. A partir de dicha matriz homogénea, se obtendrá mediante un programa de MATLAB las ocho posibles configuraciones de variables articulares para alcanzar dicha posición. A partir de este momento, la trayectoria y simulación tanto de MATLAB como de ADAMS View se realizarán de manera análoga a la cinemática directa utilizando la primera solución de las ocho posibles.

7.5.1 Resolución Cinemática Inversa en MATLAB



- Parámetros iniciales: En el script "Cinematicainversa.m", se han introducido previamente todos los parámetros correspondientes al método de Denavit y Hartenberg, así como todas las distancias de las barras, de manera idéntica al script de la cinemática directa. En este fichero también se han introducido las variables articulares Theta deseadas en la posición inicial y la matriz homogénea con la posición y orientación final deseada para el efector final. Por último, también se ha introducido en este como dato importante el tiempo en el cual se desea que realice dicho movimiento y el incremento de tiempo, el cual es importante a la hora de calcular la trayectoria. Este fichero, también agrupa todas las funciones que se encargan de la resolución de la cinemática inversa y que serán presentadas a continuación.

- Generación de las ocho posibles soluciones de la cinemática inversa: "invcinpuma.m" es una función la cual recibe la matriz con todos los datos del PUMA 560, así como las variables articulares deseadas en la posición inicial y la matriz homogénea con la posición y orientación deseada para el efector final. Dicha función entrega como resultado una matriz 6x8, con las ocho posibles configuraciones articulares del robot para lograr que el efector final esté en dicha posición deseada.

- Cálculo de la trayectoria: ''nuevagentrayinv.m" es una función que recibe como input el tiempo final y el incremento de tiempo que se desea para el movimiento. Esta función tiene definidas las ecuaciones del movimiento de cada tramo según evoluciona el tiempo. "nuevagentrayinv.m" devolverá una matriz de tantas filas como saltos temporales se realicen con el incremento de tiempo establecido hasta alcanzar el tiempo final, y con tantas columnas como variables articulares haya. Cabe destacar que los coeficientes de la generación de trayectorias se han calculado de la misma manera que en la cinemática directa, pero aplicando ahora el primer resultado de "invpuma.m".

- Cálculo de toda la cinemática de posición y movimiento: se trata de la función "jacobcalculo.m", la cual recibe las variables articulares en su posición inicial, las velocidades y aceleraciones extraídas de la función de generación de trayectorias y por último recibe también las variables articulares en su posición final. Dicha función es la que permite resolver al completo la cinemática directa, devolviendo como resultado las coordenadas de posición del efector final, sus ángulos de Euler y las velocidades y aceleraciones de las seis articulaciones. Se trata de la misma función usada en la cinemática directa.

A continuación, se explicará el funcionamiento del archivo "Cinematicainversa.m", así como todas las funciones que lo integran y donde se resuelve el problema cinemático inverso.

- "cinematicainversa.m":

```
%El siguiente programa resuelve la cinemática inversa completa del robot
%PUMA 560 dado la matriz DH del robot y la matriz homogenea del efector
%final en su posicion final, así como el tiempo del movimiento y
%el incremento del mismo.
%-------------------------------------------------------------------------
%Primer paso: añadir la matriz con
%los parámetros de DH y las longitudes de todas las barras del robot.

%  alpha    a     d    theta tipo  xi  yi  zi  xf  yf  zf    R   G   B
PUMA=[ 0       0    0     0     0    0   0   0   0   0 -623.57 1   0   1;
   -pi/2     0    0     0     0    0   0   0   0   0  129.54 0   1   1;
     0     431.8 129.54 0     0    0   0   0 -431.8 0   0    0   0  0.9;
   pi/2   -20.32  0     0     0    0   0   0  20.32 0   0    0   1   0;
  -pi/2      0   432.09 0     0    0   0   0   0 432.09 0   1    0   0;
   pi/2      0    0     0     0    0   0   0   0   0    0  0.5  0.5  0;
     0       0   56.26  0     0    0   0   0   0   0 -56.26 0.4 0.8 0.8];
 %-------------------------------------------------------------------------
 %Segundo paso: llamada de la funcion "nuevagentrayinv" la cual me devolverá
 %el ángulo en radianes de cada uno de los motores del robot,
 %así como la velocidad y aceleración de cada uno de ellos durante el
 %intervalo de tiempo especificado y para las posiciones deseadas.
 %Esta parte me devolverá una matriz de 18 columnas y tantas filas como
 %saltos de incremento de tiempo hayan hasta que se cumpla el tiempo final
 %del movimiento.
 thetasinigrad=[0 0 0 0 0 0];
 mathomfin=[0 1 0 200;1 0 0 250;0 0 1 0;0 0 0 1];
 thetaini=gradrad(thetasinigrad); %robot en posición de reposo
 tfin=3;
```

```
 inct=0.0375;
 mat=invcinpuma(PUMA,mathomfin,thetaini);%usaremos la primera solución pero
 %podemos usar cualquiera de ellas que el resultado es el mismo
 [trayvelacelinv]=nuevagentrayinv(tfin,inct);
 %-------------------------------------------------------------------------
 %A continuación se iniciará un bucle for que se repita tanto como tantas
 %filas tenga "trayvelacel" y se aplicará la función "jacobcalculo" para
 %resolver la cinemática directa.
 tamano=size(trayvelacelinv);
 Cinematicaresultado=[];
 for i=1:tamano(1)
     Cinematicaresultado=[Cinematicaresultado;jacobcalculo(thetaini,...
     trayvelacelinv(i,7:12),trayvelacelinv(i,13:18),...
     trayvelacelinv(i,1:6))];
 end
 Cinematicaresultado;
```

- "invcinpuma.m" (se aplicarán en esta las fórmulas vistas en el apartado de teoría de cinemática inversa. Devolverá una matriz de ocho columnas y seis filas).

```
%Le llega la matriz de DH del robot, la matriz homogenea de posición y orientación del efector final,
%y las variables theta de las articulaciones en la posición en la que se encuentra

function matincin=invcinpuma(robot,matriz,thetas)

px=matriz(1,4);
py=matriz(2,4);
pz=matriz(3,4);
nx=matriz(1,1);
ny=matriz(2,1);
nz=matriz(3,1);
sx=matriz(1,2);
sy=matriz(2,2);
sz=matriz(3,2);
ax=matriz(1,3);
ay=matriz(2,3);
az=matriz(3,3);

d2=robot(3,3);
d4=robot(5,3);
d6=robot(7,3);
a2=robot(3,2);
a3=robot(4,2);

 pwx=px-d6*ax;
 pwy=py-d6*ay;
 pwz=pz-d6*az;

 lamda=((((pwx*cos(thetas(1)))+(pwy*sin(thetas(1))))^2)+(pwz^2)-(a3^2)-(a2^2)-(d4^2))/(2*a2);

 %Posibles variables thetal:
 thetalpos=atan2(-pwx,pwy)+atan2(+sqrt(pwy^2+pwx^2-d2^2),d2);
 thetalneg=atan2(-pwx,pwy)+atan2(-sqrt(pwy^2+pwx^2-d2^2),d2);

 %Posibles variables theta3:
 theta3pos=atan2(d4,a3)+atan2(sqrt(a3^2+d4^2-lamda^2),lamda);
 theta3neg=atan2(d4,a3)+atan2(-sqrt(a3^2+d4^2-lamda^2),lamda);

%Posibles variables theta2:
theta21=atan2((cos(thetalpos)*pwx+sin(thetalpos)*pwy)*(-a3*sin(theta3pos)+d4*cos(theta3pos))...
    -pwz*(a2+a3*cos(theta3pos)+d4*sin(theta3pos)),(cos(thetalpos)*pwx+sin(thetalpos)*pwy)...
    *(a2+a3*cos(theta3pos)+d4*sin(theta3pos))+pwz*(-a3*sin(theta3pos)+d4*cos(theta3pos)));
theta22=atan2((cos(thetalpos)*pwx+sin(thetalpos)*pwy)*(-a3*sin(theta3neg)+d4*cos(theta3neg))...
    -pwz*(a2+a3*cos(theta3neg)+d4*sin(theta3neg)),(cos(thetalpos)*pwx+sin(thetalpos)*pwy)...
    *(a2+a3*cos(theta3neg)+d4*sin(theta3neg))+pwz*(-a3*sin(theta3neg)+d4*cos(theta3neg)));
theta23=atan2((cos(thetalneg)*pwx+sin(thetalneg)*pwy)*(-a3*sin(theta3pos)+d4*cos(theta3pos))...
    -pwz*(a2+a3*cos(theta3pos)+d4*sin(theta3pos)),(cos(thetalneg)*pwx+sin(thetalneg)*pwy)...
    *(a2+a3*cos(theta3pos)+d4*sin(theta3pos))+pwz*(-a3*sin(theta3pos)+d4*cos(theta3pos)));
theta24=atan2((cos(thetalneg)*pwx+sin(thetalneg)*pwy)*(-a3*sin(theta3neg)+d4*cos(theta3neg))...
    -pwz*(a2+a3*cos(theta3neg)+d4*sin(theta3neg)),(cos(thetalneg)*pwx+sin(thetalneg)*pwy)...
    *(a2+a3*cos(theta3neg)+d4*sin(theta3neg))+pwz*(-a3*sin(theta3neg)+d4*cos(theta3neg)));
```

```
%Posibles variables theta4:
theta411=atan2(-sin(thetalpos)*ax+cos(thetalpos)*ay,cos(thetalpos)*cos(theta21)*cos(theta3pos)...
    *ax+sin(thetalpos)*cos(theta21)*cos(theta3pos)*ay-sin(theta21)*sin(theta3pos)*az);
theta421=atan2(sin(thetalpos)*ax-cos(thetalpos)*ay,sin(theta21)*sin(theta3pos)*az-cos(thetalpos)...
    *cos(theta21)*cos(theta3pos)*ax-sin(thetalpos)*cos(theta21)*cos(theta3pos)*ay);
theta412=atan2(-sin(thetalpos)*ax+cos(thetalpos)*ay,cos(thetalpos)*cos(theta22)*cos(theta3neg)...
    *ax+sin(thetalpos)*cos(theta22)*cos(theta3neg)*ay-sin(theta22)*sin(theta3neg)*az);
theta422=atan2(sin(thetalpos)*ax-cos(thetalpos)*ay,sin(theta22)*sin(theta3neg)*az-cos(thetalpos)...
    *cos(theta22)*cos(theta3neg)*ax-sin(thetalpos)*cos(theta22)*cos(theta3neg)*ay);
theta413=atan2(-sin(thetalneg)*ax+cos(thetalneg)*ay,cos(thetalneg)*cos(theta23)*cos(theta3pos)...
    *ax+sin(thetalneg)*cos(theta23)*cos(theta3pos)*ay-sin(theta23)*sin(theta3pos)*az);
theta423=atan2(sin(thetalneg)*ax-cos(thetalneg)*ay,sin(theta23)*sin(theta3pos)*az-cos(thetalneg)...
    *cos(theta23)*cos(theta3pos)*ax-sin(thetalneg)*cos(theta23)*cos(theta3pos)*ay);
theta414=atan2(-sin(thetalneg)*ax+cos(thetalneg)*ay,cos(thetalneg)*cos(theta24)*cos(theta3neg)...
    *ax+sin(thetalneg)*cos(theta24)*cos(theta3neg)*ay-sin(theta24)*sin(theta3neg)*az);
theta424=atan2(sin(thetalneg)*ax-cos(thetalneg)*ay,sin(theta24)*sin(theta3neg)*az-cos(thetalneg)...
    *cos(theta24)*cos(theta3neg)*ax-sin(thetalneg)*cos(theta24)*cos(theta3neg)*ay);
```

```
%Posibles variables theta5:
theta51=atan2(sqrt(1-((cos(theta21)*cos(theta3pos)-sin(theta21)*sin(theta3pos))*az+(sin(thetalpos)...
    *cos(theta21)*sin(theta3pos)+sin(thetalpos)*sin(theta21)*sin(theta3pos))*ay+(cos(thetalpos)...
    *cos(theta21)*sin(theta3pos)+cos(thetalpos)*sin(theta21)*cos(theta3pos))*ax)^2),(cos(theta21)*...
    cos(theta3pos)-sin(theta21)*sin(theta3pos))*az+(sin(thetalpos)*cos(theta21)*sin(theta3pos)...
    +sin(thetalpos)*sin(theta21)*sin(theta3pos))*ay+(cos(thetalpos)*cos(theta21)*sin(theta3pos)...
    +cos(thetalpos)*sin(theta21)*cos(theta3pos)*ax));
theta52=atan2(sqrt(1-((cos(theta22)*cos(theta3neg)-sin(theta22)*sin(theta3neg))*az+(sin(thetalpos)...
    *cos(theta22)*sin(theta3neg)+sin(thetalpos)*sin(theta22)*sin(theta3neg))*ay+(cos(thetalpos)...
    *cos(theta22)*sin(theta3neg)+cos(thetalpos)*sin(theta22)*cos(theta3neg))*ax)^2),(cos(theta22)...
    *cos(theta3neg)-sin(theta22)*sin(theta3neg))*az+(sin(thetalpos)*cos(theta22)*sin(theta3neg)...
    +sin(thetalpos)*sin(theta22)*sin(theta3neg))*ay+(cos(thetalpos)*cos(theta22)*sin(theta3neg)...
    +cos(thetalpos)*sin(theta22)*cos(theta3neg)*ax));
theta53=atan2(sqrt(((cos(theta23)*cos(theta3pos)-sin(theta23)*sin(theta3pos))*sz+(sin(thetalneg)...
    *cos(theta23)*sin(theta3pos)+sin(thetalneg)*sin(theta23)*cos(theta3pos))*sy+(cos(thetalneg)...
    *cos(theta23)*sin(theta3pos)+cos(thetalneg)*sin(theta23)*cos(theta3pos))*sx)^2+((sin(theta23)...
    *sin(theta3pos)-cos(theta23)*cos(theta3pos))*nz+(-sin(thetalneg)*cos(theta23)*sin(theta3pos)...
    -sin(thetalneg)*sin(theta23)*cos(theta3pos))*ny+(-cos(thetalneg)*cos(theta23)*sin(theta3pos)...
    -cos(thetalneg)*sin(theta23)*cos(theta3pos))*nx)^2),(cos(theta23)*cos(theta3pos)-sin(theta23)...
    *sin(theta3pos))*az+(sin(thetalneg)*cos(theta23)*sin(theta3pos)+sin(thetalneg)*sin(theta23)...
    *cos(theta3pos))*ay+(cos(thetalneg)*cos(theta23)*sin(theta3pos)+cos(thetalneg)*sin(theta23)*cos(theta3pos))*ax);
```

```
theta54=atan2(sqrt(((cos(theta24)*cos(theta3neg)-sin(theta24)*sin(theta3neg))*sz+(sin(thetalneg)...
    *cos(theta24)*sin(theta3neg)+sin(thetalneg)*sin(theta24)*cos(theta3neg))*sy+(cos(thetalneg)...
    *cos(theta24)*sin(theta3neg)+cos(thetalneg)*sin(theta24)*cos(theta3neg))*sx)^2+((sin(theta24)...
    *sin(theta3neg)-cos(theta24)*cos(theta3neg))*nz+(-sin(thetalneg)*cos(theta24)*sin(theta3neg)...
    -sin(thetalneg)*sin(theta24)*cos(theta3neg))*ny+(-cos(thetalneg)*cos(theta24)*sin(theta3neg)...
    -cos(thetalneg)*sin(theta24)*cos(theta3neg))*nx)^2),(cos(theta24)*cos(theta3neg)-sin(theta24)...
    *sin(theta3neg))*az+(sin(thetalneg)*cos(theta24)*sin(theta3neg)+sin(thetalneg)*sin(theta24)...
    *cos(theta3neg))*ay+(cos(thetalneg)*cos(theta24)*sin(theta3neg)+cos(thetalneg)*sin(theta24)*cos(theta3neg))*ax);


%Posibles variables theta6:
theta611=atan2((cos(theta21)*cos(theta3pos)-sin(theta21)*sin(theta3pos))*sz+(sin(thetalpos)*cos(theta21)...
    *sin(theta3pos)+sin(thetalpos)*sin(theta21)*cos(theta3pos))*sy+(cos(thetalpos)*cos(theta21)*sin(theta3pos)...
    +cos(thetalpos)*sin(theta21)*cos(theta3pos))*sx,(sin(theta21)*sin(theta3pos)-cos(theta21)*cos(theta3pos))...
    *nz+(-sin(thetalpos)*cos(theta21)*sin(theta3pos)-sin(thetalpos)*sin(theta21)*cos(theta3pos))*ny+(-cos(thetalpos)...
    *cos(theta21)*sin(theta3pos)-cos(thetalpos)*sin(theta21)*cos(theta3pos))*nx);
theta621=atan2(-((cos(theta21)*cos(theta3pos)-sin(theta21)*sin(theta3pos))*sz+(sin(thetalpos)*cos(theta21)...
    *sin(theta3pos)+sin(thetalpos)*sin(theta21)*cos(theta3pos))*sy+(cos(thetalpos)*cos(theta21)*sin(theta3pos)...
    +cos(thetalpos)*sin(theta21)*cos(theta3pos))*sx),-((sin(theta21)*sin(theta3pos)-cos(theta21)*cos(theta3pos))...
    *nz+(-sin(thetalpos)*cos(theta21)*sin(theta3pos)-sin(thetalpos)*sin(theta21)*cos(theta3pos))*ny+(-cos(thetalpos)...
    *cos(theta21)*sin(theta3pos)-cos(thetalpos)*sin(theta21)*cos(theta3pos))*nx));
theta612=atan2((cos(theta22)*cos(theta3neg)-sin(theta22)*sin(theta3neg))*sz+(sin(thetalpos)*cos(theta22)...
    *sin(theta3neg)+sin(thetalpos)*sin(theta22)*cos(theta3neg))*sy+(cos(thetalpos)*cos(theta22)*sin(theta3neg)...
    +cos(thetalpos)*sin(theta22)*cos(theta3neg))*sx,(sin(theta22)*sin(theta3neg)-cos(theta22)*cos(theta3neg))...
    *nz+(-sin(thetalpos)*cos(theta22)*sin(theta3neg)-sin(thetalpos)*sin(theta22)*cos(theta3neg))*ny+(-cos(thetalpos)...
    *cos(theta22)*sin(theta3neg)-cos(thetalpos)*sin(theta22)*cos(theta3neg))*nx);
theta622=atan2(-((cos(theta22)*cos(theta3neg)-sin(theta22)*sin(theta3neg))*sz+(sin(thetalpos)*cos(theta22)...
    *sin(theta3neg)+sin(thetalpos)*sin(theta22)*cos(theta3neg))*sy+(cos(thetalpos)*cos(theta22)*sin(theta3neg)...
    +cos(thetalpos)*sin(theta22)*cos(theta3neg))*sx),-((sin(theta22)*sin(theta3neg)-cos(theta22)*cos(theta3neg))...
    *nz+(-sin(thetalpos)*cos(theta22)*sin(theta3neg)-sin(thetalpos)*sin(theta22)*cos(theta3neg))*ny+(-cos(thetalpos)...
    *cos(theta22)*sin(theta3neg)-cos(thetalpos)*sin(theta22)*cos(theta3neg))*nx));
theta613=atan2((cos(theta23)*cos(theta3pos)-sin(theta23)*sin(theta3pos))*sz+(sin(thetalneg)*cos(theta23)...
    *sin(theta3pos)+sin(thetalneg)*sin(theta23)*cos(theta3pos))*sy+(cos(thetalneg)*cos(theta23)*sin(theta3pos)...
    +cos(thetalneg)*sin(theta23)*cos(theta3pos))*sx,(sin(theta23)*sin(theta3pos)-cos(theta23)*cos(theta3pos))...
    *nz+(-sin(thetalneg)*cos(theta23)*sin(theta3pos)-sin(thetalneg)*sin(theta23)*cos(theta3pos))*ny+(-cos(thetalneg)...
    *cos(theta23)*sin(theta3pos)-cos(thetalneg)*sin(theta23)*cos(theta3pos))*nx);
theta623=atan2(-((cos(theta23)*cos(theta3pos)-sin(theta23)*sin(theta3pos))*sz+(sin(thetalneg)*cos(theta23)...
    *sin(theta3pos)+sin(thetalneg)*sin(theta23)*cos(theta3pos))*sy+(cos(thetalneg)*cos(theta23)*sin(theta3pos)...
    +cos(thetalneg)*sin(theta23)*cos(theta3pos))*sx),-((sin(theta23)*sin(theta3pos)-cos(theta23)*cos(theta3pos))...
    *nz+(-sin(thetalneg)*cos(theta23)*sin(theta3pos)-sin(thetalneg)*sin(theta23)*cos(theta3pos))*ny+(-cos(thetalneg)...
    *cos(theta23)*sin(theta3pos)-cos(thetalneg)*sin(theta23)*cos(theta3pos))*nx));
theta614=atan2((cos(theta24)*cos(theta3neg)-sin(theta24)*sin(theta3neg))*sz+(sin(thetalneg)*cos(theta24)...
    *sin(theta3neg)+sin(thetalneg)*sin(theta24)*cos(theta3neg))*sy+(cos(thetalneg)*cos(theta24)*sin(theta3neg)...
    +cos(thetalneg)*sin(theta24)*cos(theta3neg))*sx,(sin(theta24)*sin(theta3neg)-cos(theta24)*cos(theta3neg))...
    *nz+(-sin(thetalneg)*cos(theta24)*sin(theta3neg)-sin(thetalneg)*sin(theta24)*cos(theta3neg))*ny+(-cos(thetalneg)...
    *cos(theta24)*sin(theta3neg)-cos(thetalneg)*sin(theta24)*cos(theta3neg))*nx);
theta624=atan2(-((cos(theta24)*cos(theta3neg)-sin(theta24)*sin(theta3neg))*sz+(sin(thetalneg)*cos(theta24)...
    *sin(theta3neg)+sin(thetalneg)*sin(theta24)*cos(theta3neg))*sy+(cos(thetalneg)*cos(theta24)*sin(theta3neg)...
    +cos(thetalneg)*sin(theta24)*cos(theta3neg))*sx),-((sin(theta24)*sin(theta3neg)-cos(theta24)*cos(theta3neg))...
    *nz+(-sin(thetalneg)*cos(theta24)*sin(theta3neg)-sin(thetalneg)*sin(theta24)*cos(theta3neg))*ny+(-cos(thetalneg)...
    *cos(theta24)*sin(theta3neg)-cos(thetalneg)*sin(theta24)*cos(theta3neg))*nx));
```

```
columna1=[thetalpos;theta21;theta3pos;theta411;theta51;-theta611];
columna2=[thetalpos;theta21;theta3pos;theta421;-theta51;-theta621];
columna3=[thetalpos;theta22;theta3neg;theta422;theta52;-theta612];
columna4=[thetalpos;theta22;theta3neg;theta412;-theta52;-theta622];
columna5=[thetalneg;theta24;theta3neg;theta413;theta54;-theta613];
columna6=[thetalneg;theta24;theta3neg;theta423;-theta54;-theta623];
columna7=[thetalneg;theta23;theta3pos;theta424;theta53;-theta614];
columna8=[thetalneg;theta23;theta3pos;theta414;-theta53;-theta624];

matincin=[columna1,columna2,columna3,columna4,columna5,columna6,columna7,columna8];

end
```

- "nuevagentrayinv.m" (coeficientes calculados de manera análoga al apartado de cinemática directa usando la primera solución de la función "invcinpuma.m", resolución de las ocho posibles configuraciones articulares).

```
function [Output]=nuevagentrayinv(tiempo,inc)
cont=1;
%Valores de todos los coeficientes de la trayectoria en una misma matriz

%             tr par 1     /          tr rec           /                    tr par 2      /
%            alf   gamma         a              V           alf         beta         gamma
        coef=[0 0.119874449 -0.119874449 0.239748898 -0.599372245 0.719246694 -0.119874449
            0 -0.259589365 0.259589365 -0.519178729 1.297946823 -1.557536188 0.259589365
            0 1.017312715 -1.017312715 2.034625429 -5.086563574 6.103876288 -1.017312715
            0 0.785398163 -0.785398163 1.570796327 -3.926990817 4.71238898 -0.785398163
            0 0.765050242 -0.765050242 1.530100485 -3.825251212 4.590301454 -0.765050242
            0 -0.512573531 0.512573531 -1.025147061 2.562867654 -3.075441184 0.512573531

   der tr par 1 /  der tr rec  /      der tr par 2           /   der 2trpar/ der2 tr rec/ der 2 tr par 2
alfa    gamma  a        V     alfa     beta        gamma        gamma       -      gamma
 0  0.239748898 0 0.239748898 0 0.719246694 -0.239748898 0.239748898 0 -0.239748898;
  0 -0.519178729 0 -0.519178729 0 -1.557536188 0.519178729 -0.519178729 0 0.519178729;
  0 2.034625429 0 2.034625429 0 6.103876288 -2.034625429 2.034625429 0 -2.034625429;
  0 1.570796327 0 1.570796327   0 4.71238898 -1.570796327 1.570796327 0 -1.570796327;
  0 1.530100485 0 1.530100485 0 4.590301454 -1.530100485 1.530100485 0 -1.530100485;
  0 -1.025147061 0 -1.025147061 0 -3.075441184 1.025147061 -1.025147061 0 1.025147061];

        for i=inc:inc:tiempo
            for j=1:6 %los ejes
                if i<1
                    Output(cont,j)=coef(j,2)*i*i;
                    Output(cont,j+6)=2*coef(j,2)*i;
                    Output(cont,j+12)=2*coef(j,2);
                elseif i==1
                    Output(cont,j)=coef(j,2)*i;
                    Output(cont,j+6)=coef(j,2);
                    Output(cont,j+12)=0;
                elseif i<2
                    Output(cont,j)=coef(j,3)+coef(j,4)*i;
                    Output(cont,j+6)=coef(j,4);
                    Output(cont,j+12)=0;
                elseif i==2
                    Output(cont,j)= coef(j,3)+coef(j,4)*i;
                    Output(cont,j+6)=coef(j,4);
                    Output(cont,j+12)=0;
                else
                    Output(cont,j)=coef(j,5)+coef(j,6)*i+coef(j,7)*i*i;
                    Output(cont,j+6)=coef(j,6)+2*coef(j,7)*i;
                    Output(cont,j+12)=2*coef(j,7);
                end
            end
            cont=cont+1;
        end
        end
```

7.5.3 Resultados de la Cinemática Inversa

Una vez se tiene el código en MATLAB y los resultados de la simulación realizada con ADAMS View y MATLAB, se va a proceder a comparar los resultados obtenidos por ambos métodos para ver que coincidan y asegurar que la modelización de la cinemática inversa es correcta.

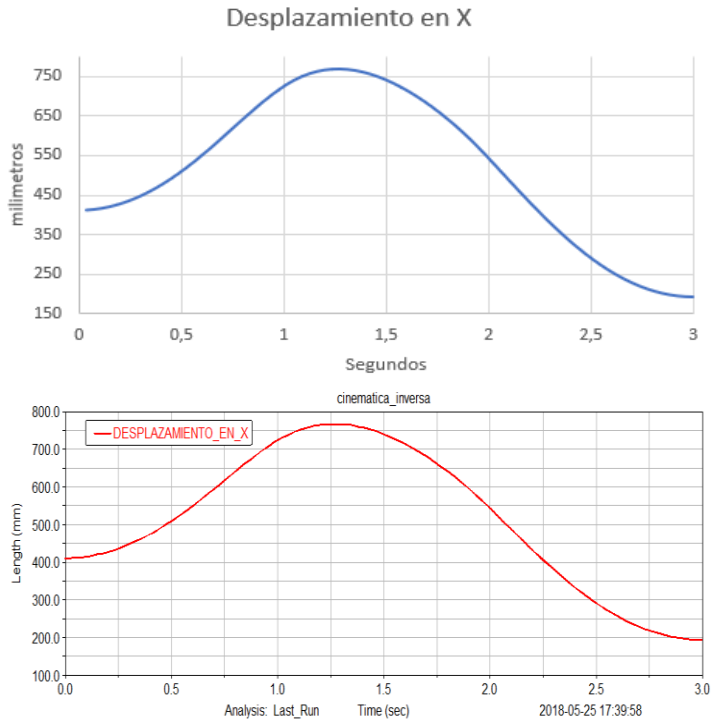


Figura 44: Desplazamiento en X MATLAB y ADAMS View

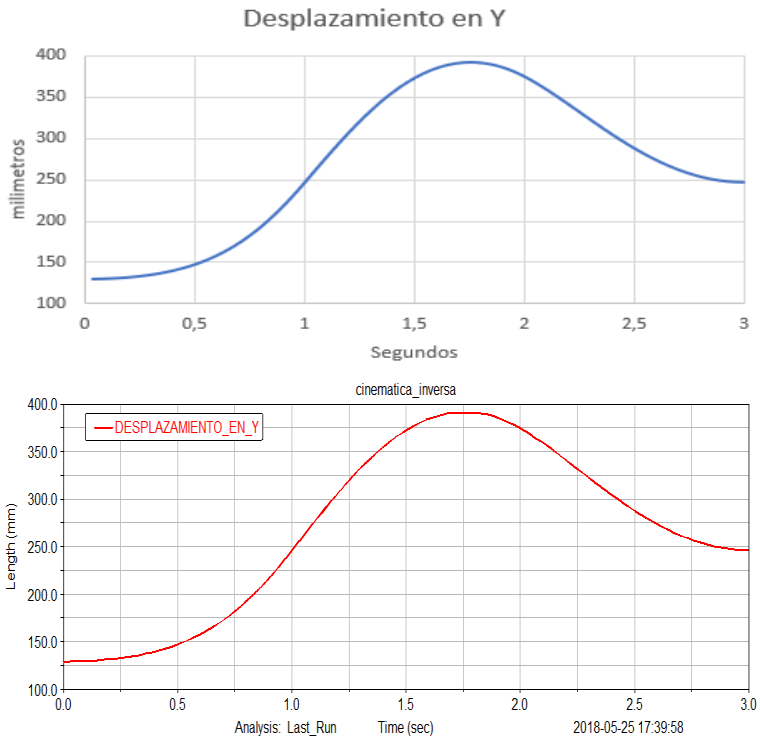


Figura 45: Desplazamiento en Y MATLAB y ADAMS View

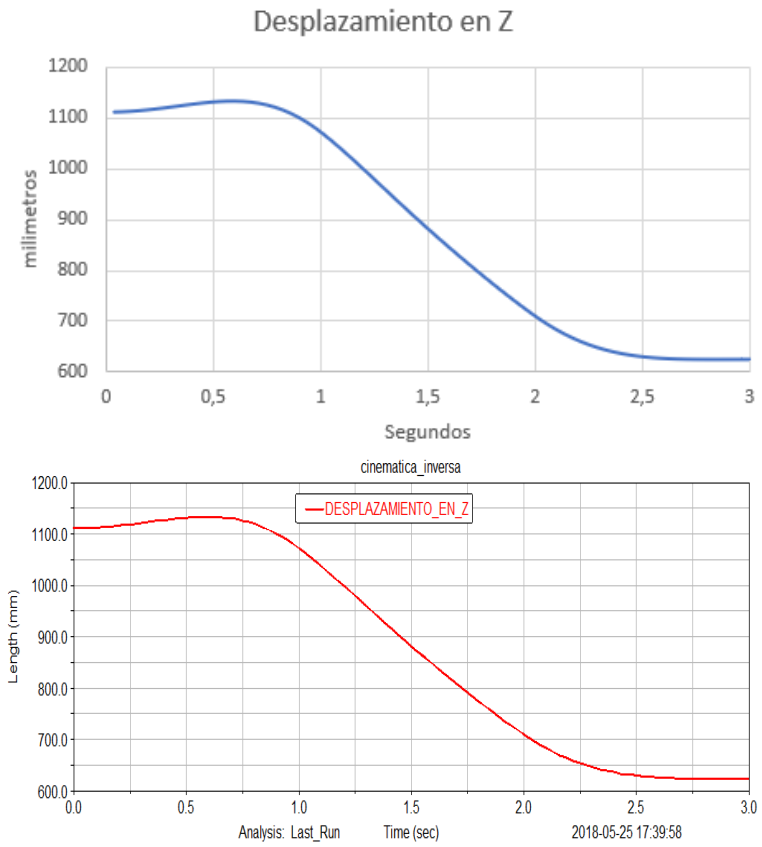


Figura 46: Desplazamiento en Z MATLAB y ADAMS View

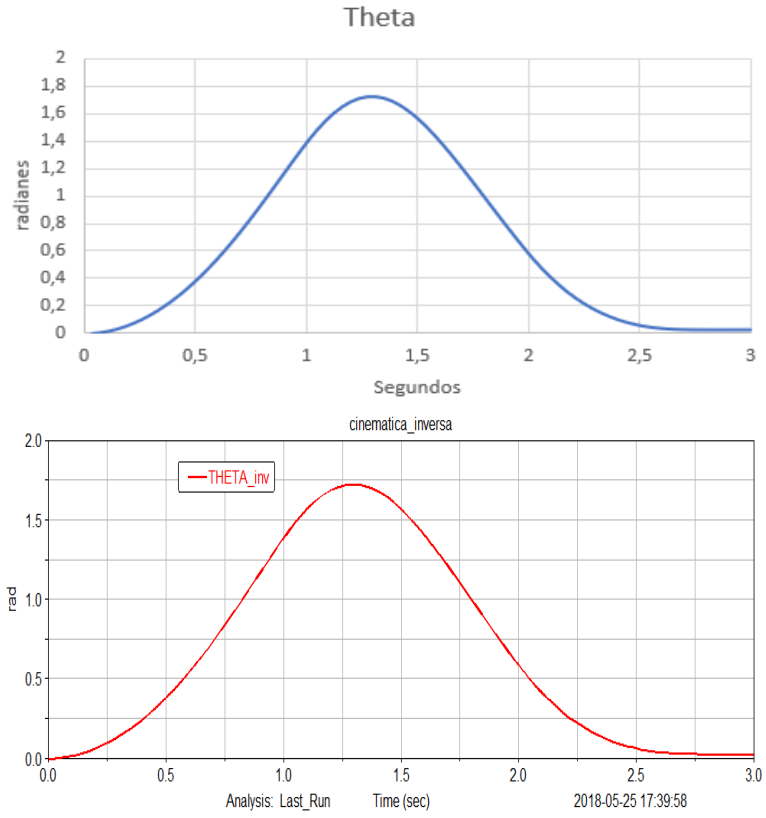


Figura 47: Ángulo Theta MATLAB y ADAMS View

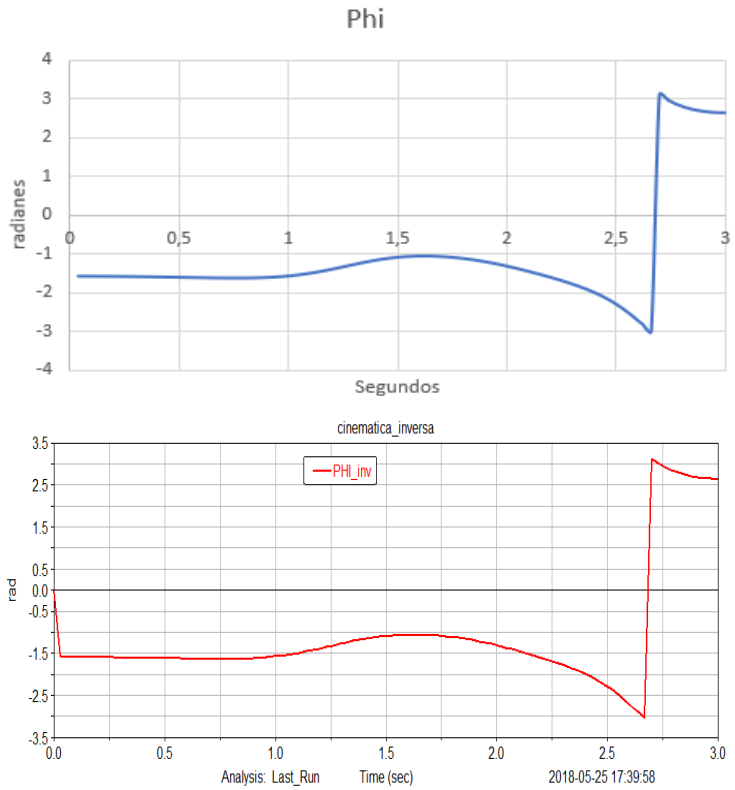


Figura 48: Ángulo PHI MATLAB y ADAMS View

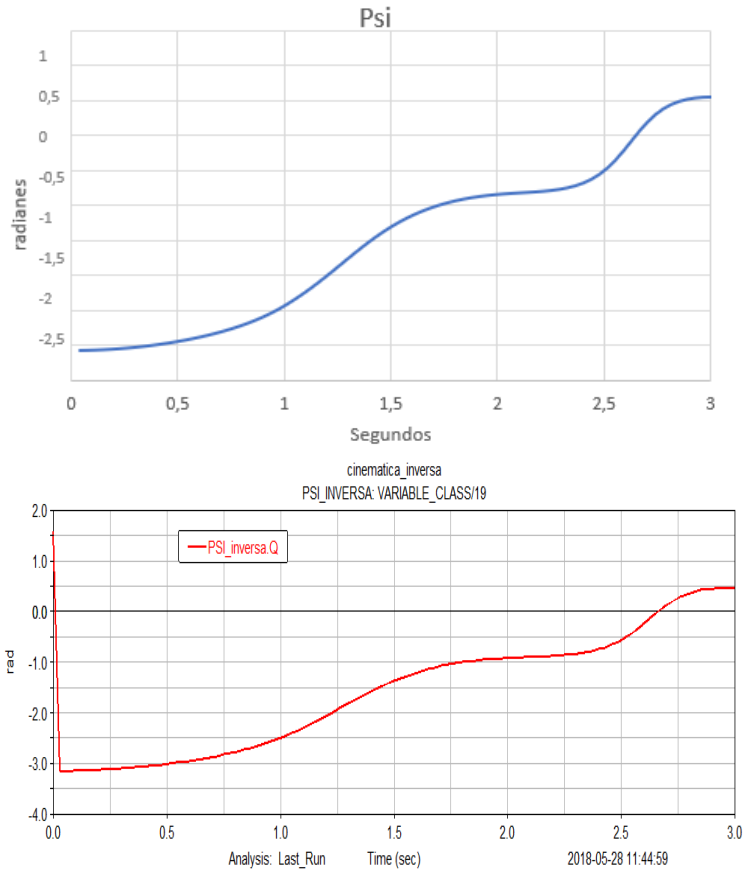


Figura 49: Ángulo PSI MATLAB y ADAMS View

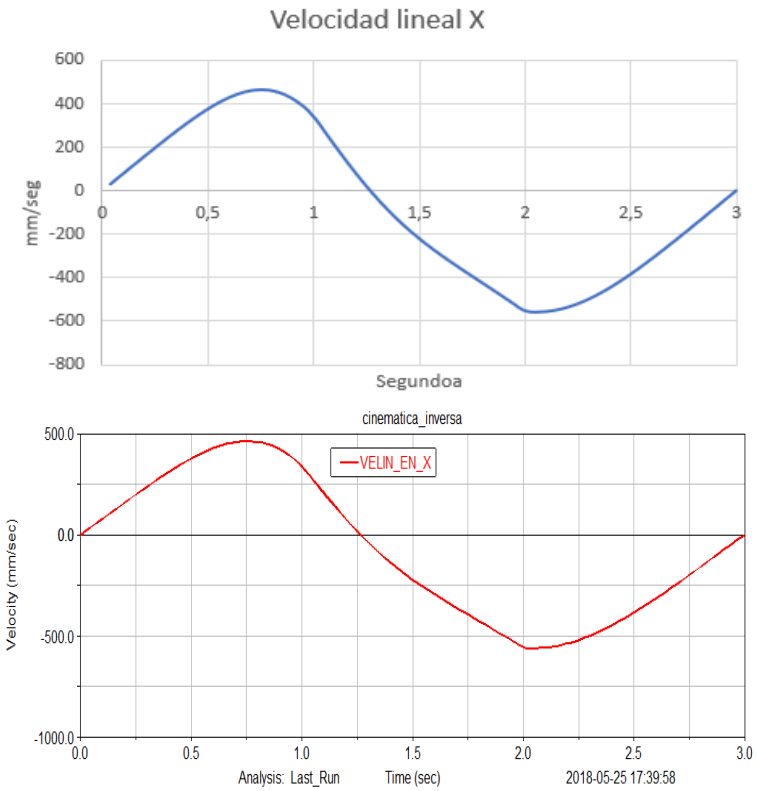


Figura 50: Velocidad lineal en X MATLAB y ADAMS View

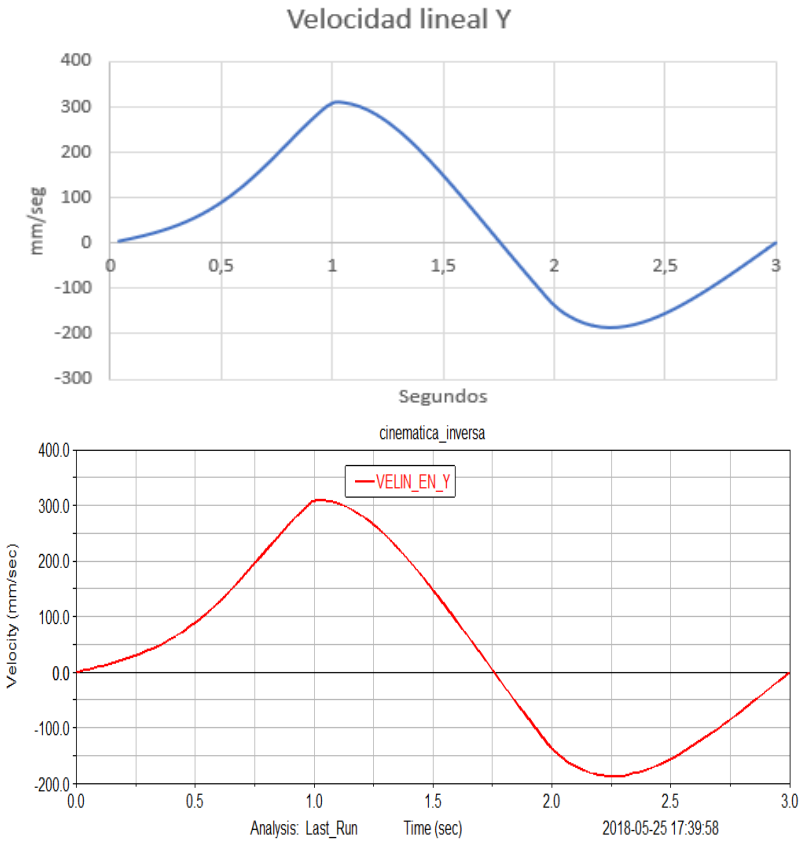


Figura 51: Velocidad lineal en Y MATLAB y ADAMS View

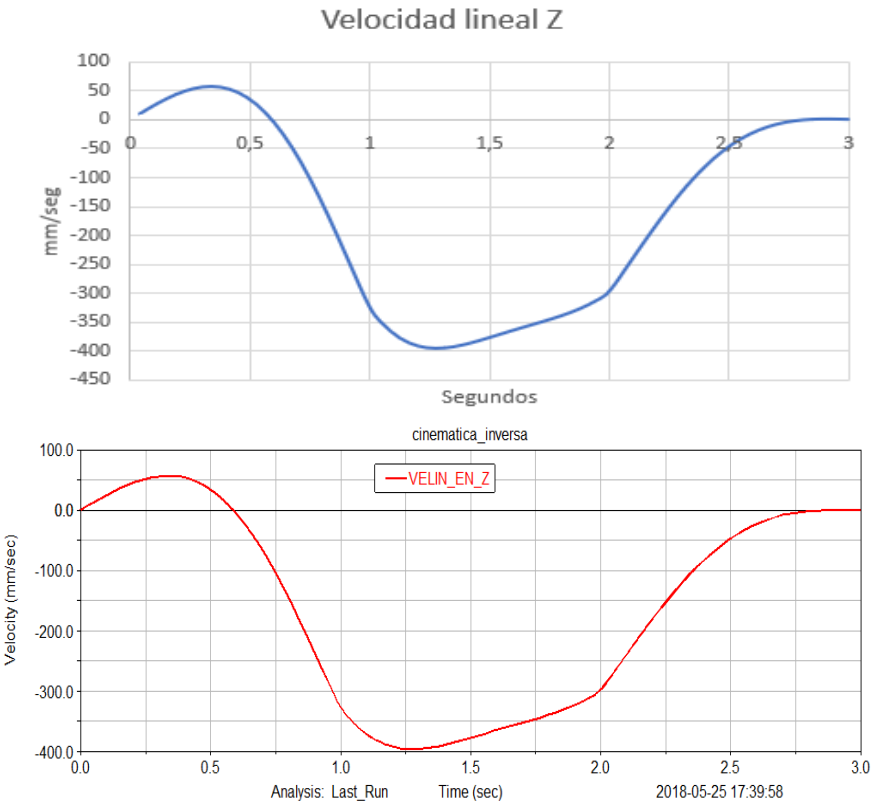


Figura 52: Velocidad lineal en Z MATLAB y ADAMS View

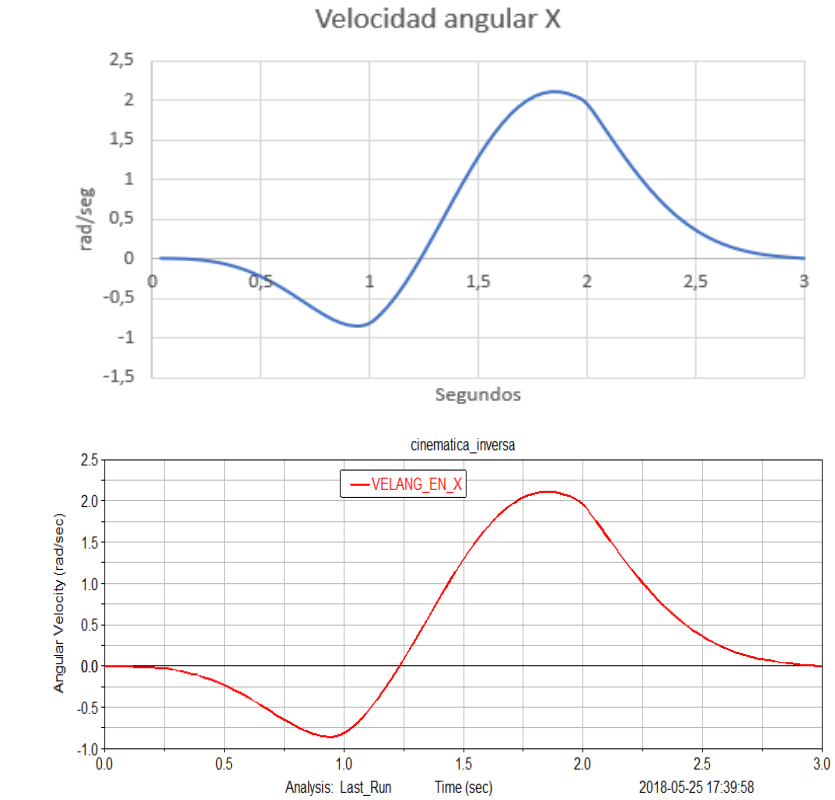


Figura 53: Velocidad angular en X MATLAB y ADAMS View

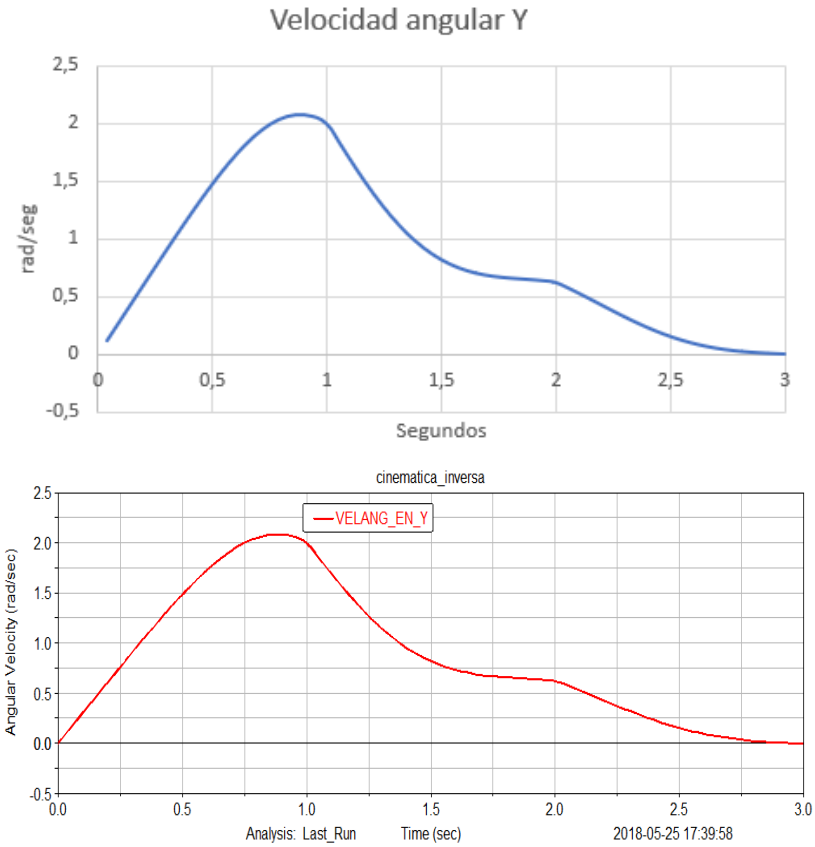


Figura 54: Velocidad angular en Y MATLAB y ADAMS View

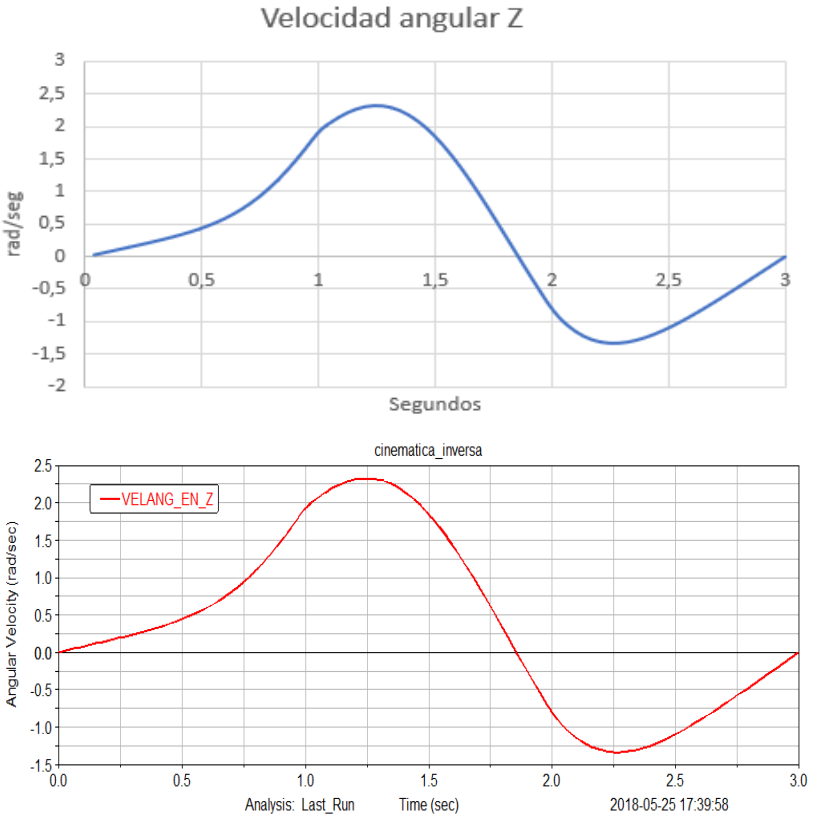


Figura 55: Velocidad angular en Z MATLAB y ADAMS View

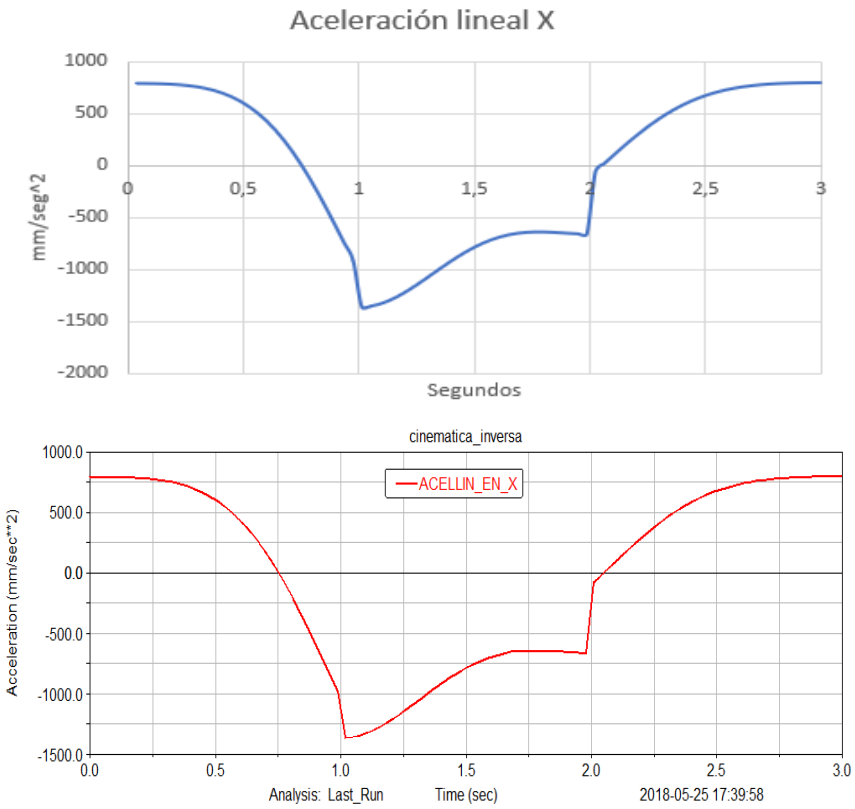


Figura 56: Aceleración lineal en X MATLAB y ADAMS View

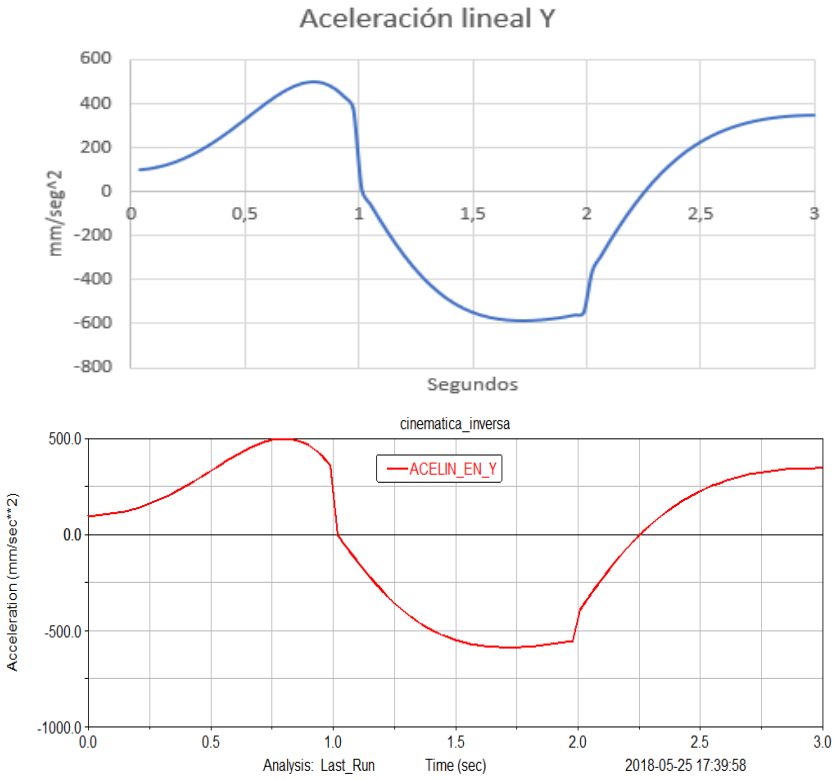


Figura 57: Aceleración lineal en Y MATLAB y ADAMS View

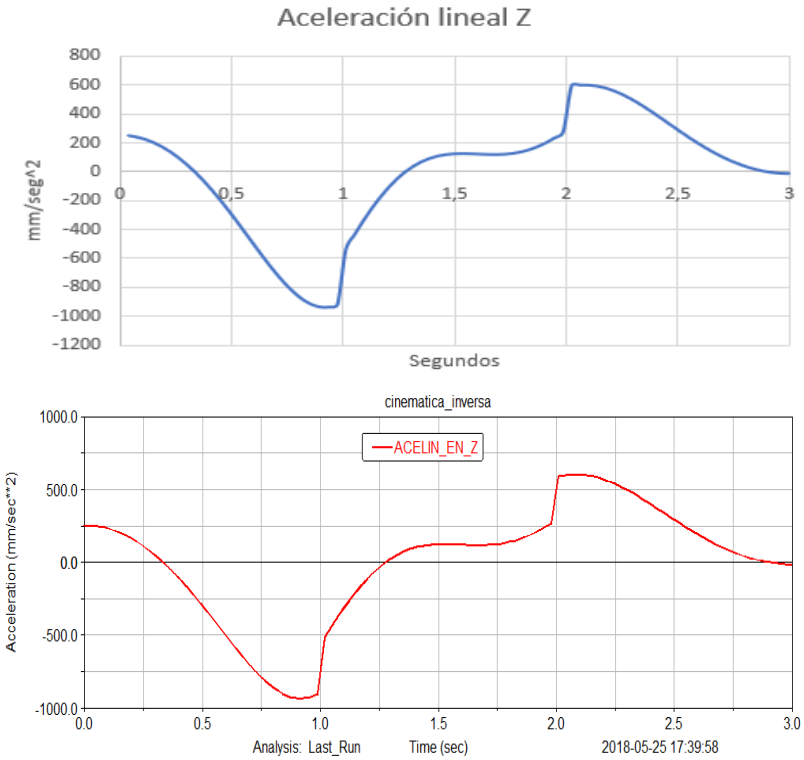


Figura 58: Aceleración lineal en Z MATLAB y ADAMS View

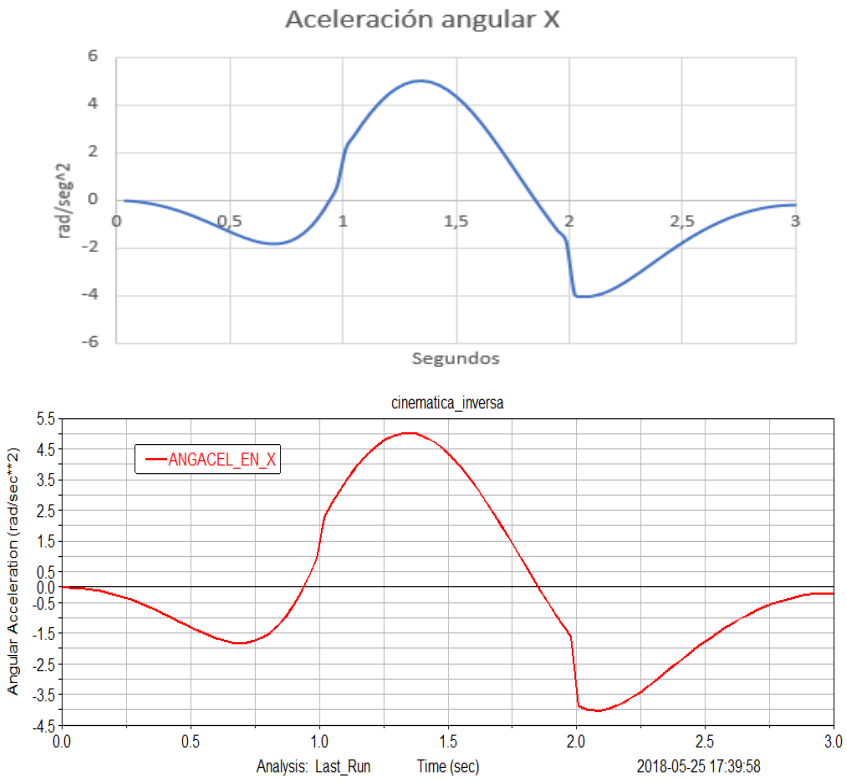


Figura 59: Aceleración angular en X MATLAB y ADAMS View

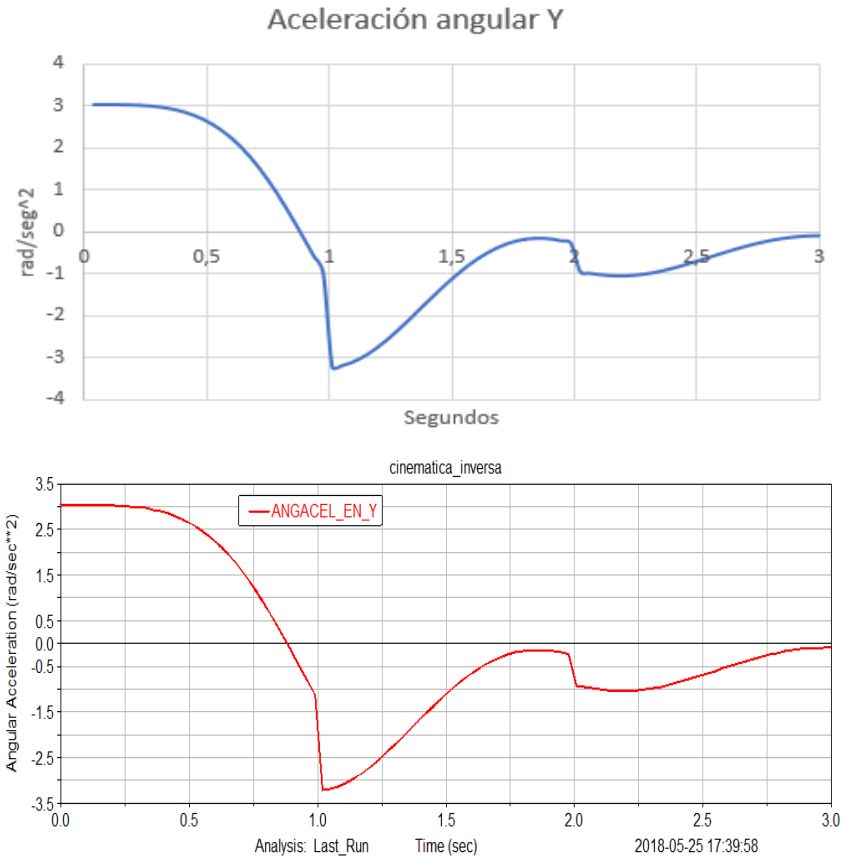


Figura 60: Aceleración angular en Y MATLAB y ADAMS View

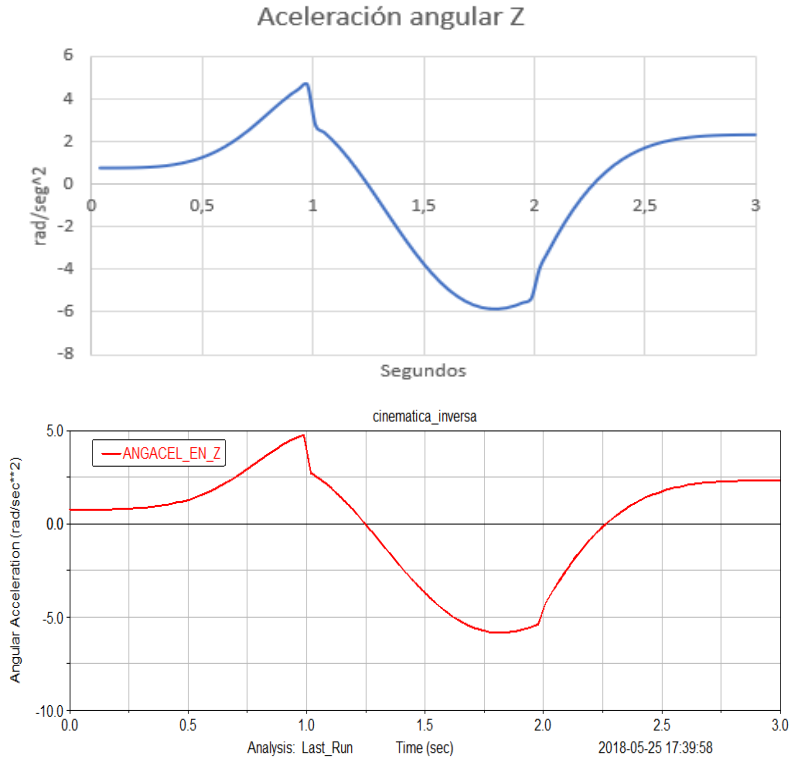


Figura 61: Aceleración angular en Z MATLAB y ADAMS View

Como se puede observar, los resultados tanto de la simulación con ADAMS View como los del modelado matemático desarrollado con MATLAB del robot PUMA 560, son coincidentes.

Se ha realizado exitosamente la simulación y el modelado de las coordenadas X, Y, Z del efector final, los ángulos de Euler, las velocidades y aceleraciones tanto traslacionales como angulares para este caso.

Además, el código de MATLAB puede obtener las posiciones de los nudos y la segunda terna de ángulos de Euler para cada instante y para cualquier combinación de variables articulares deseada. Dicho procedimiento también se podría haber aplicado de manera satisfactoria para cualquiera de las ocho soluciones de la cinemática inversa.

En cuanto a las unidades de medida de ADAMS View se han usado las que se muestran en la siguiente Figura 62:

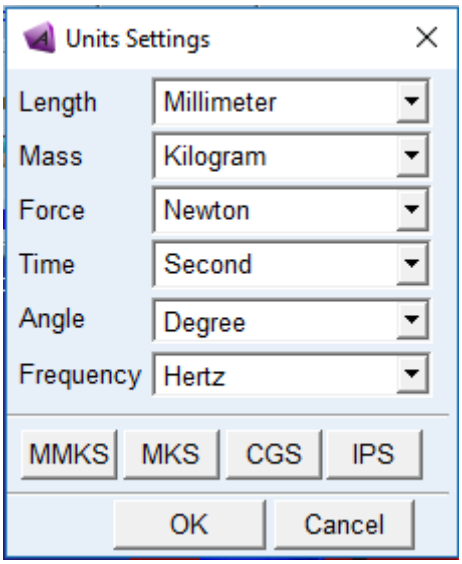


Figura 62: Unidades de medida utilizadas en la cinemática inversa

7.6 Resolución de la Dinámica inversa

Para resolver la dinámica del robot PUMA 560, primeramente, se han de establecer los parámetros másicos e inerciales del modelo. En el caso de este proyecto se van a utilizar los parámetros relativos a PAUL81:

Param	*Armstrong*	*Paul81*	*Tarn*
m_1	-	4.43	13.00
m_2	17.40	10.20	22.40
m_3	4.80	4.80	5.00
m_4	0.82	1.18	1.20
m_5	0.35	0.32	0.62
m_6	0.09	0.13	0.16

Figura 63: Masas de las barras en kg

Param	*Armstrong*	*Paul81*	*Tarn*
s_{x1}	-	0	0
s_{y1}	-	80	4
s_{z1}	-	0	-309
s_{x2}	68	216	103
s_{y2}	6	0	5
s_{z2}	-16	-26	-40
s_{x3}	0	0	20
s_{y3}	-70	-216	-4
s_{z3}	14	0	14
s_{x4}	0	0	0
s_{y4}	0	0	-3
s_{z4}	-19	-20	-86
s_{x5}	0	0	0
s_{y5}	0	0	-1
s_{z5}	0	0	-10
s_{x6}	0	0	0
s_{y6}	0	0	0
s_{z6}	32	10	3

Figura 64: Localización relativa de los Centros de Masas respecto a los S.D.R. de cada barra (mm)

Param	*Armstrong*	*Tarn*	*Paul81*
I_{xx_1}	-	1.100	0.195
I_{yy_1}	-	1.110	0.026
I_{zz_1}	0.350	0.177	0.195
I_{xx_2}	0.130	0.403	0.588
I_{yy_2}	0.524	0.969	1.886
I_{zz_2}	0.539	0.965	1.470
I_{xx_3}	66.0 e-3	74.8 e-3	324.0 e-3
I_{yy_3}	12.5 e-3	7.3 e-3	17.0 e-3
I_{zz_3}	86.0 e-3	75.6 e-3	324.0 e-3
I_{xx_4}	1.80 e-3	5.32 e-3	3.83 e-3
I_{yy_4}	1.80 e-3	5.20 e-3	3.83 e-3
I_{zz_4}	1.30 e-3	3.37 e-3	2.50 e-3
I_{xx_5}	300 e-6	487 e-6	216 e-6
I_{yy_5}	300 e-6	482 e-6	216 e-6
I_{zz_5}	400 e-6	572 e-6	348 e-6
I_{xx_6}	150 e-6	123 e-6	437 e-6
I_{yy_6}	150 e-6	123 e-6	437 e-6
I_{zz_6}	40 e-6	58 e-6	13 e-6

Figura 65: Momentos de inercia respecto a los Cetros de Masas de cada barra (Kg.m2)

7.6.1 Parámetros másicos e inerciales en ADAMS View

Teniendo en cuenta los datos presentados en el apartado anterior, se montará el modelo dinámico en ADAMS View. Primeramente, se establecerán los pesos y momentos de inercia de cada barra de la siguiente manera:

Se clicará con el botón derecho en cada barra y se seleccionará la opción "Modify". Una vez allí seleccionaremos la opción "User Imput" dentro de la categoría "Mass Properties" donde se introducirá la masa en kilos de cada cuerpo, así como los momentos de inercia de cada eje en Kg.m2. En el apartado "Center of Mass *Marker*" se detallará el *Marker* que representará el centro de masas de cada barra, quedando como se muestra en la Figura 66:

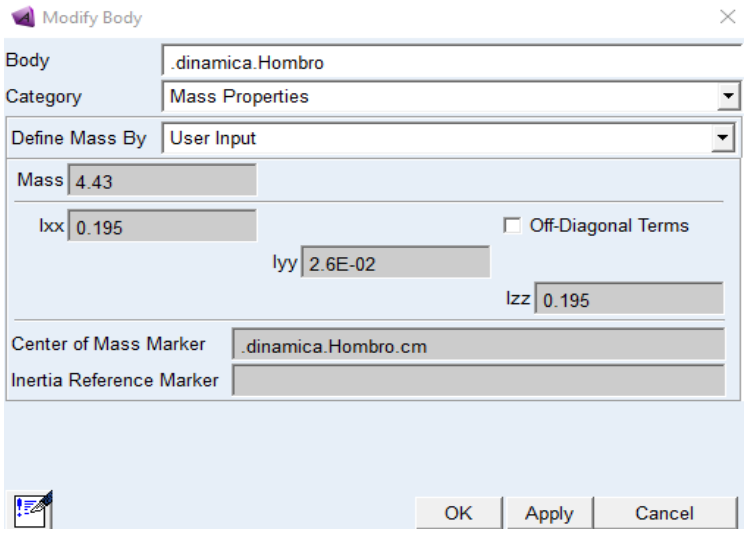


Figura 66: Parámetros másicos e inerciales en ADAMS View

Una vez definidos estos parámetros, es necesario poner la posición y orientación correcta de los *Marker*s correspondientes a los centros de masas de cada barra. Para ello iremos al árbol general y de cada cuerpo modificaremos cada uno de los *Marker*s de los centros de masas poniendo la distancia relativa a cada uno de sus respectivos sistemas de referencia quedado de la siguiente manera:

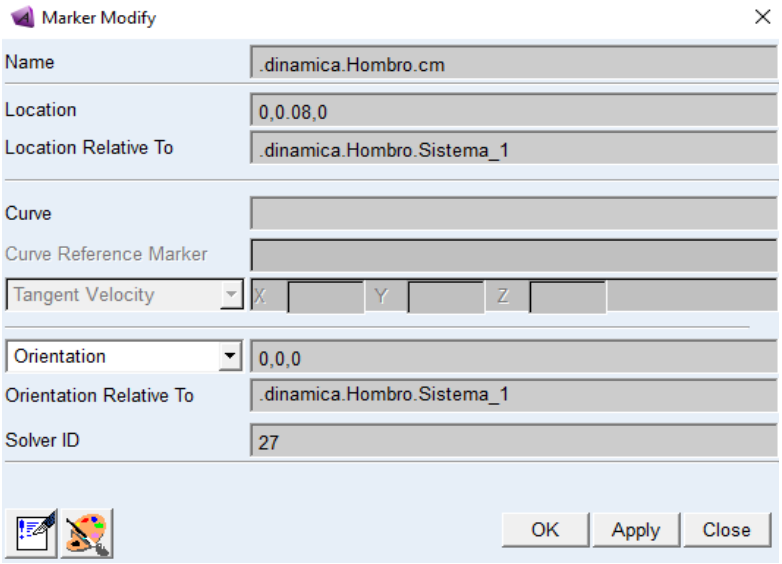


Figura 67: Localización y orientación C.d.M en ADAMS View

Una vez establecidos dichos parámetros el modelo de simulación en ADAMS View ya esta completo para ser simulado y medir cada uno de los torques que ejerce los motores sobre las articulaciones del robot.

Las unidades que se emplearán en este apartado son las que se indican en la Figura 68:

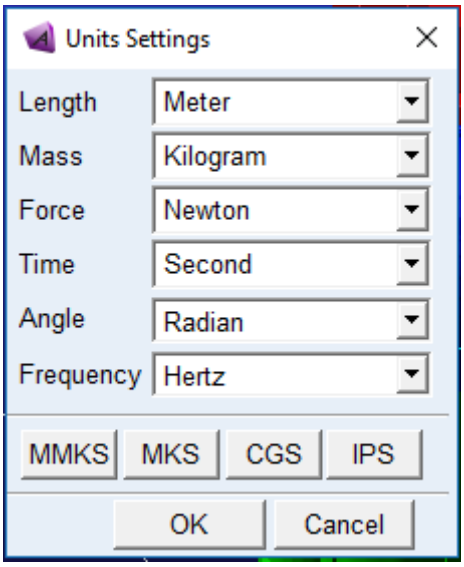
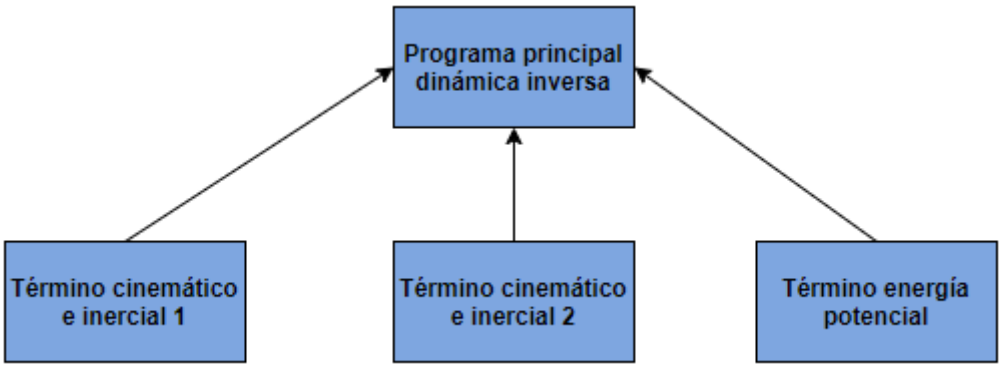


Figura 68: Unidades de medida para la simulación dinámica

7.6.2 Resolución Dinámica inversa en MATLAB



- El primer término de la ecuación que resolverá la dinámica inversa se calcula con la ejecución del script "montaD.m", el cual recibe los grados de libertad, las variables de nudo y las aceleraciones angulares. Con todo esto y junto a los datos inerciales incluidos en el mismo script, se resuelve el primer término de la ecuación de la dinámica para el caso del PUMA 560, devolviendo una matriz de seis filas y una columna.

- El segundo término de la ecuación de la dinámica se calcula con la ejecución del script "montahijk.m" el cual recibe las variables articulares, los parámetros inerciales y las velocidades angulares. Y devuelve una matriz de una columna y seis filas.

- El término de la energía potencial se calcula con la ejecución del script "montan.m" el cual recibe las variables articulares y los parámetros inerciales y gravitatorios. Este también devuelve una matriz de una columna y seis filas.

- La matriz H formada por operaciones con los productos de inercia, las masas y centros de masas de cada barra se calculará mediante el scipt "montaH.m" incluido en las funciones anteriores.

- Todas estas funciones se agrupan en el script "dinamicainversa.m" en el cual mediante un bucle calculan los torques de las diferentes articulaciones mientras dura el movimiento del robot y devuelve en una matriz de una columna y seis filas los respectivos torques de cada motor. Dicho script agrupa en él todas las funciones necesarias para la resolución de la dinámica inversa.

A continuación, se detallará el código de todas las funciones utilizadas para la resolución de la dinámica inversa del robot PUMA 560 para un caso en concreto.

- "motaD.m":

```
function [Resultadol]=montaD(gradlib,thetas,acel)
qh=thetas;
q=sym('q',[1 6]);
d=[0 0.12954 0 0.43209 0 0.05626];
a=[0 0.4318 -0.02032 0 0 0];
alfa=[-pi/2 0 pi/2 -pi/2 pi/2 0];

%saco las matrices de transformción homogenea entre sistemas
%de coordenadas consecutivos en funcion de q
A1 = denavit(q(1), d(1), a(1), alfa(1));
A2 = denavit(q(2), d(2), a(2), alfa(2));
A3 = denavit(q(3), d(3), a(3), alfa(3));
A4 = denavit(q(4), d(4), a(4), alfa(4));
A5 = denavit(q(5), d(5), a(5), alfa(5));
A6 = denavit(q(6), d(6), a(6), alfa(6));

%m,Kgm^2
%       sx       sy     sz       Ix        Iy       Iz        m
Dat=[   0     0.08     0    0.223352    0.026     0.223352  4.43;
      0.216 0    -0.026   0.5948952   2.3687864 1.9458912 10.2;
        0 -0.216     0    0.5479488   0.017     0.5479488 4.8;
        0    0    -0.02   0.004302    0.004302  0.0025    1.18;
        0    0       0    0.000216    0.000216  0.000348  0.32;
        0    0    0.01    0.00045     0.00045   0.000013  0.13];
```

$$ {}^{l}I_{O_{xx}} = {}^{l}I_{G_{xx}} + m_l \cdot \left[\left({}^{l}y_{G_l} \right)^2 + \left({}^{l}z_{G_l} \right)^2 \right] $$

-

```
   matD=[];

for i=1:gradlib
for j=1:gradlib

    if(j>=i)
      p=j;
       if (p==1)
        T=A1;
        elseif (p==2)
        T=A1*A2;
        elseif (p==3)
        T=A1*A2*A3;
        elseif (p==4)
        T=A1*A2*A3*A4;
        elseif (p==5)
        T=A1*A2*A3*A4*A5;
        else
        T=A1*A2*A3*A4*A5*A6;
        end

       matD(i,j)=double(subs(trace((diff(T,q(j),1))*montaH(Dat(p,:))*(diff(T,q(i),1))'),q,qh));

    else
       p=i;
        if (p==1)
        T=A1;
        elseif (p==2)
        T=A1*A2;
        elseif (p==3)
        T=A1*A2*A3;
        elseif (p==4)
        T=A1*A2*A3*A4;
        elseif (p==5)
        T=A1*A2*A3*A4*A5;
        else
        T=A1*A2*A3*A4*A5*A6;
        end

       matD(i,j)=double(subs(trace((diff(T,q(j),1))*montaH(Dat(p,:))*(diff(T,q(i),1))'),q,qh));

    end
end
end
Resultadol=matD*acel';
end
```

$$ \sum_{j=1}^{n} D_{ij} \cdot \ddot{q}_j $$

$$ D_{ij} = \sum_{p=\max(i,j)}^{n} tr\left[\frac{\partial {}^{0}A_p}{\partial q_k} \cdot {}^{p}H_{O_p} \cdot \left(\frac{\partial {}^{0}A_p}{\partial q_k} \right)^{T} \right] $$

- "Montahijk.m":

```
function [Resultado2]=montahijk(gradlib,thetas,vel)
qd=vel;
qh=thetas;
q=sym('q',[1 6]);
d=[0 0.12954 0 0.43209 0 0.05626];
a=[0 0.4318 -0.02032 0 0 0];
alfa=[-pi/2 0 pi/2 -pi/2 pi/2 0];

%saco las matrices de transformción homogenea entre sistemas
%de coordenadas consecutivos en funcion de q
A1 = denavit(q(1), d(1), a(1), alfa(1));
A2 = denavit(q(2), d(2), a(2), alfa(2));
A3 = denavit(q(3), d(3), a(3), alfa(3));
A4 = denavit(q(4), d(4), a(4), alfa(4));
A5 = denavit(q(5), d(5), a(5), alfa(5));
A6 = denavit(q(6), d(6), a(6), alfa(6));

%m,Kgm^2
%      sx       sy     sz       Ix         Iy       Iz         m
Dat=[  0     0.08     0    0.223352     0.026     0.223352   4.43;
      0.216  0    -0.026   0.5948952    2.3687864 1.9458912  10.2;
        0  -0.216    0     0.5479488    0.017     0.5479488  4.8;
        0    0    -0.02    0.004302     0.004302  0.0025     1.18;
        0    0       0     0.000216     0.000216  0.000348   0.32;
        0    0     0.01    0.00045      0.00045   0.000013   0.13];
```

```
 math=zeros(6,1);

 for i=1:gradlib
     for j=1:gradlib
        for k=1:gradlib

  if(i>=j)&&(i>=k)
      p=i;
      if (p==1)
      T=A1;
      elseif (p==2)
      T=A1*A2;
      elseif (p==3)
      T=A1*A2*A3;
      elseif (p==4)
      T=A1*A2*A3*A4;
      elseif (p==5)
      T=A1*A2*A3*A4*A5;
      else
      T=A1*A2*A3*A4*A5*A6;
      end

    math(i)=math(i)+(double(subs(trace(diff((diff(T,q(j),1)),...
        q(k),1)*montaH(Dat(p,:))*(diff(T,q(i),1))'),q,qh))*qd(j)*qd(k));
```

$$\sum_{j=1}^{n}\sum_{k=1}^{n} h_{jk}\,\dot{q}_j\dot{q}_k$$

$$h_{jk}=\sum_{p=\max(i,j,k)}^{n} tr\left[\frac{\partial^2({}^0A_p)}{\partial q_k \partial q_m}\,{}^pH_{o_p}\left(\frac{\partial {}^0A_p}{\partial q_k}\right)^T\right]$$

```
elseif (j>i)&&(j>=k)
    p=j;
     if (p==1)
     T=A1;
     elseif (p==2)
     T=A1*A2;
     elseif (p==3)
     T=A1*A2*A3;
     elseif (p==4)
     T=A1*A2*A3*A4;
     elseif (p==5)
     T=A1*A2*A3*A4*A5;
     else
     T=A1*A2*A3*A4*A5*A6;
     end

    math(i)=math(i)+(double(subs(trace(diff((diff(T,q(j),1)),...
        q(k),1)*montaH(Dat(p,:))*(diff(T,q(i),1))'),q,qh))*qd(j)*qd(k));

        else
         p=k;
         if (p==1)
         T=A1;
         elseif (p==2)
         T=A1*A2;
         elseif (p==3)
         T=A1*A2*A3;
         elseif (p==4)
         T=A1*A2*A3*A4;
         elseif (p==5)
         T=A1*A2*A3*A4*A5;
         else
         T=A1*A2*A3*A4*A5*A6;
         end

        math(i)=math(i)+((subs(trace(diff((diff(T,q(j),1)),...
            q(k),1)*montaH(Dat(p,:))*(diff(T,q(i),1))'),q,qh))*qd(j)*qd(k));
     end
           end
        end
    end
    Resultado2=math;
end
```

- ”montan.m”:

```
function [Resultado3]=montan(gradlib,thetas)
qh=thetas;
g=[0 0 -9.80665 0];
q=sym('q',[1 6]);
d=[0 0.12954 0 0.43209 0 0.05626];
a=[0 0.4318 -0.02032 0 0 0];
alfa=[-pi/2 0 pi/2 -pi/2 pi/2 0];

%saco las matrices de transformción homogenea entre sistemas
%de coordenadas consecutivos en funcion de q
A1 = denavit(q(1), d(1), a(1), alfa(1));
A2 = denavit(q(2), d(2), a(2), alfa(2));
A3 = denavit(q(3), d(3), a(3), alfa(3));
A4 = denavit(q(4), d(4), a(4), alfa(4));
A5 = denavit(q(5), d(5), a(5), alfa(5));
A6 = denavit(q(6), d(6), a(6), alfa(6));

%m,Kgm^2
%       sx       sy     sz       Ix         Iy       Iz         m
Dat=[  0     0.08     0    0.223352     0.026     0.223352  4.43;
      0.216 0    -0.026   0.5948952   2.3687864 1.9458912 10.2;
        0 -0.216     0    0.5479488   0.017     0.5479488 4.8;
        0    0    -0.02    0.004302    0.004302  0.0025    1.18;
        0    0        0    0.000216    0.000216  0.000348  0.32;
        0    0     0.01    0.00045     0.00045   0.000013  0.13];
    contador=1;
    matn=zeros(6,1);
  for i=1:gradlib
      for j=contador:gradlib
          if (j>i)
      p=j;
      if (p==1)
       T=A1;
       elseif (p==2)
       T=A1*A2;
       elseif (p==3)
       T=A1*A2*A3;
       elseif (p==4)
       T=A1*A2*A3*A4;
       elseif (p==5)
       T=A1*A2*A3*A4*A5;
       else
       T=A1*A2*A3*A4*A5*A6;
       end
    matn(i)=matn(i)+double(subs(-Dat(p,7)*g*diff(T,q(i),1)*([Dat(p,1:3),1]'),q,qh));
```

$$\cdot n_l =$$

$$P = -\sum_{l=1}^{n} m_l \cdot \bar{g}^T \cdot {}^0A_l \cdot {}^l\bar{r}_{O_l G_l}$$

```
    elseif (i>j)
 p=i;
if (p==1)
 T=A1;
 elseif (p==2)
 T=A1*A2;
 elseif (p==3)
 T=A1*A2*A3;
 elseif (p==4)
 T=A1*A2*A3*A4;
 elseif (p==5)
 T=A1*A2*A3*A4*A5;
 else
 T=A1*A2*A3*A4*A5*A6;
 end
matn(i)=matn(i)+double(subs(-Dat(p,7)*g*diff(T,q(i),1)*([Dat(p,1:3),1]'),q,qh));
```

```
           else
            p=i;
       if (p==1)
        T=A1;
        elseif (p==2)
        T=A1*A2;
        elseif (p==3)
        T=A1*A2*A3;
        elseif (p==4)
        T=A1*A2*A3*A4;
        elseif (p==5)
        T=A1*A2*A3*A4*A5;
        else
        T=A1*A2*A3*A4*A5*A6;
        end
       matn(i)=matn(i)+double(subs(-Dat(p,7)*g*diff(T,q(i),1)*([Dat(p,1:3),1]'),q,qh));
           end
       end
       contador=contador+1;

   end
 Resultado3=matn;

end
```

- ” montaH.m”:

```
function [matrizH]=montaH(datos)

matrizH(1,1)=((-datos(4)+datos(5)+datos(6))/2);
matrizH(1,2)=0;
matrizH(1,3)=0;
matrizH(1,4)=datos(1)*datos(7);
matrizH(2,1)=0;
matrizH(2,2)=((datos(4)-datos(5)+datos(6))/2);
matrizH(2,3)=0;
matrizH(2,4)=datos(2)*datos(7);
matrizH(3,1)=0;
matrizH(3,2)=0;
matrizH(3,3)=((datos(4)+datos(5)-datos(6))/2);
matrizH(3,4)=datos(3)*datos(7);
matrizH(4,1)=datos(7)*datos(1);
matrizH(4,2)=datos(7)*datos(2);
matrizH(4,3)=datos(7)*datos(3);
matrizH(4,4)=datos(7);

end
```

$$ {}^{1}H_{G_1} = \begin{bmatrix} \frac{-{}^{1}I_{O_{xx}} + {}^{1}I_{O_{yy}} + {}^{1}I_{O_{zz}}}{2} & {}^{1}I_{O_{xy}} & {}^{1}I_{O_{xz}} & {}^{1}x_{G_1} \cdot m_1 \\ {}^{1}I_{O_{xy}} & \frac{{}^{1}I_{O_{xx}} - {}^{1}I_{O_{yy}} + {}^{1}I_{O_{zz}}}{2} & {}^{1}I_{O_{yz}} & {}^{1}y_{G_1} \cdot m_1 \\ {}^{1}I_{O_{xz}} & {}^{1}I_{O_{yz}} & \frac{{}^{1}I_{O_{xx}} + {}^{1}I_{O_{yy}} - {}^{1}I_{O_{zz}}}{2} & {}^{1}z_{G_1} \cdot m_1 \\ m_1 \cdot {}^{1}x_{G_1} & m_1 \cdot {}^{1}y_{G_1} & m_1 \cdot {}^{1}z_{G_1} & m_1 \end{bmatrix} $$

- "dinamicainversa.m":

```
function [Resultado]=dinamicainversa()

[trayectoria]=nuevagentray(3,0.3);
gradlib=6;
tamano=size(trayectoria);
tam=tamano(1);
torque=[];

for i=1:tam
torque(i,1:6)=(montaD(gradlib,trayectoria(i,1:6),trayectoria(i,13:18))...
+montahijk(gradlib,trayectoria(i,1:6),trayectoria(i,7:12))+montan(gradlib,trayectoria(i,1:6)))';
end
Resultado=torque;

end
```

$$\sum_{j=1}^{n} D_{ij} \cdot \ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} h_{ijk} \cdot \dot{q}_j \dot{q}_k + n_i = \tau_i \quad i = 1,2,\ldots,n$$

7.6.3 Resultados Dinámica Inversa

Una vez se tiene el código en MATLAB y los resultados de la simulación realizada con ADAMS View y MATLAB para un caso en concreto, se va a proceder a comparar los resultados obtenidos por ambos métodos para ver que coincidan y asegurar que la modelización dinámica es correcta.

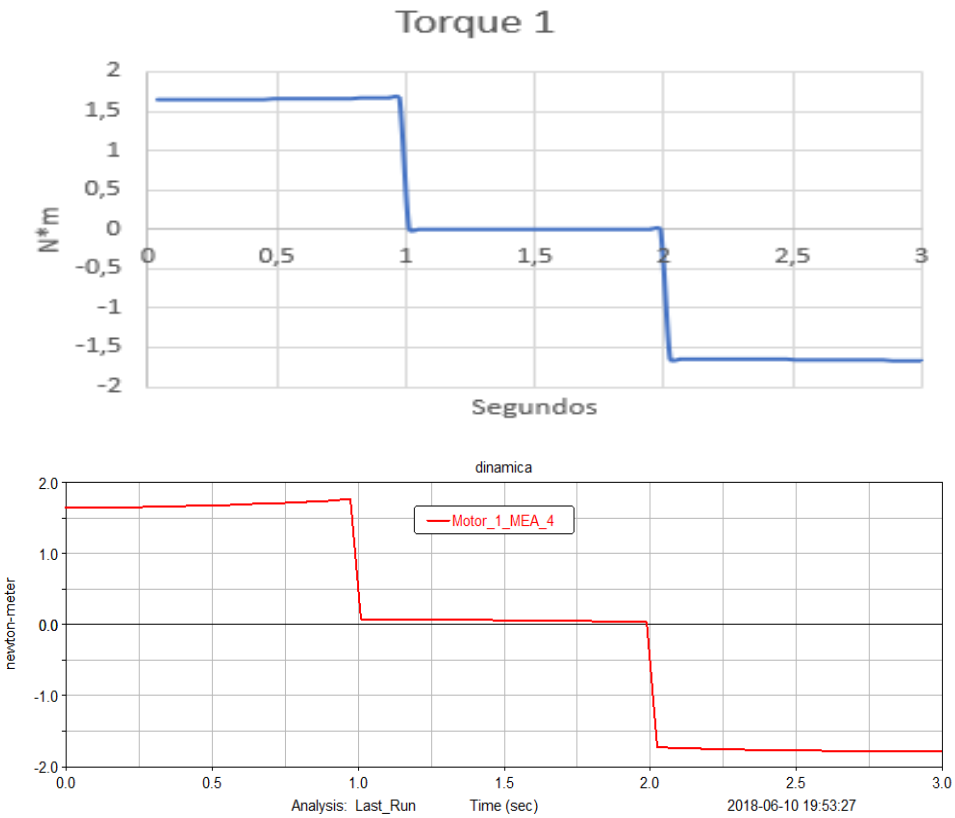


Figura 69: Par motor 1

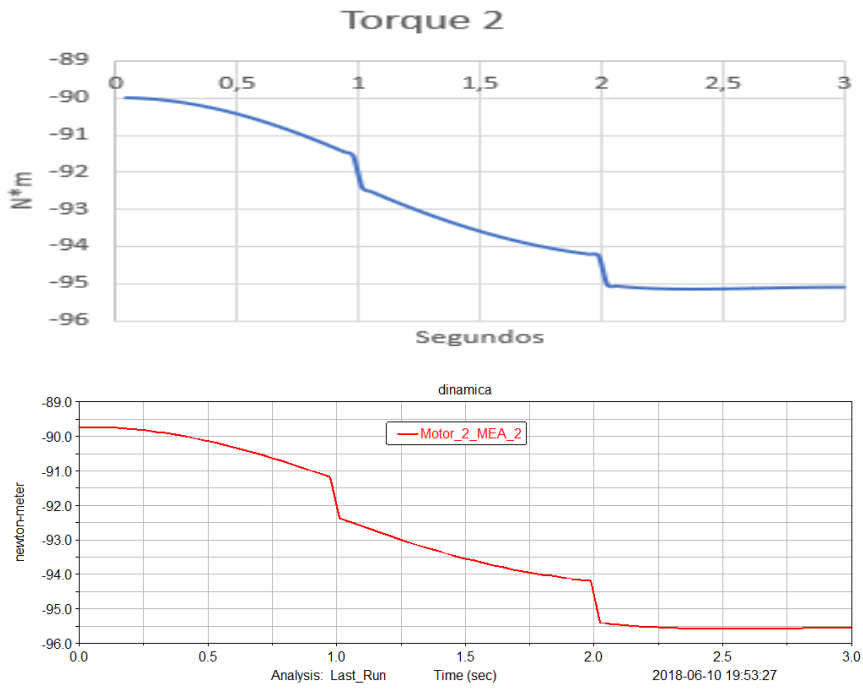


Figura 70: Par motor 2

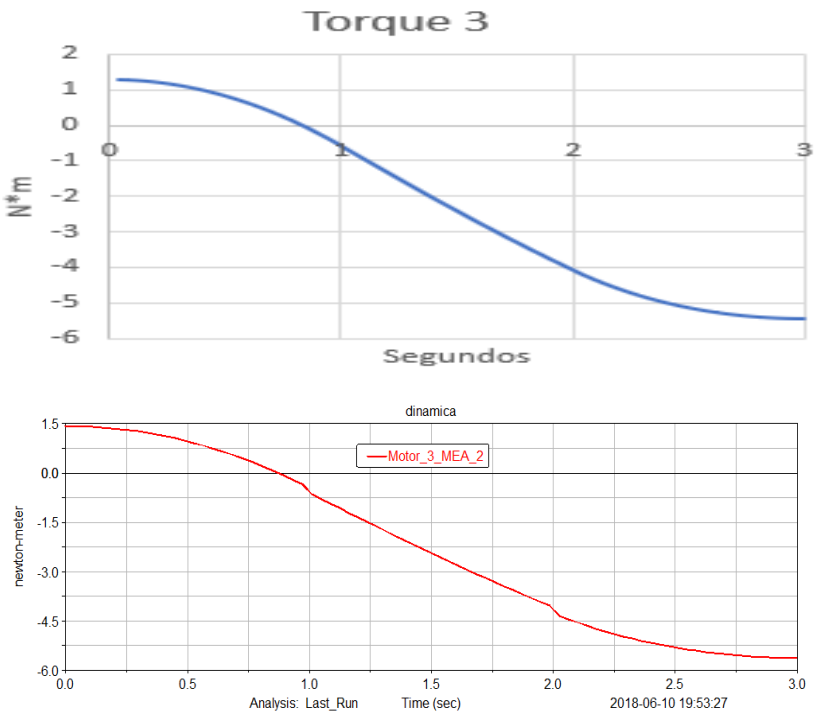


Figura 71: Par motor 3

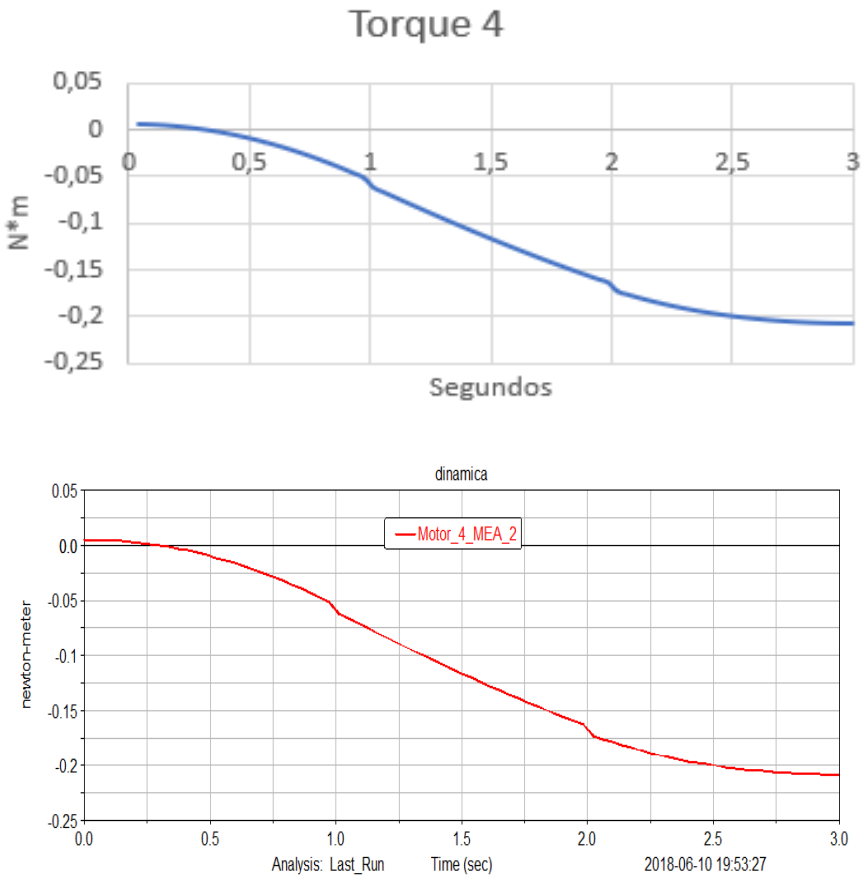


Figura 72: Par motor 4

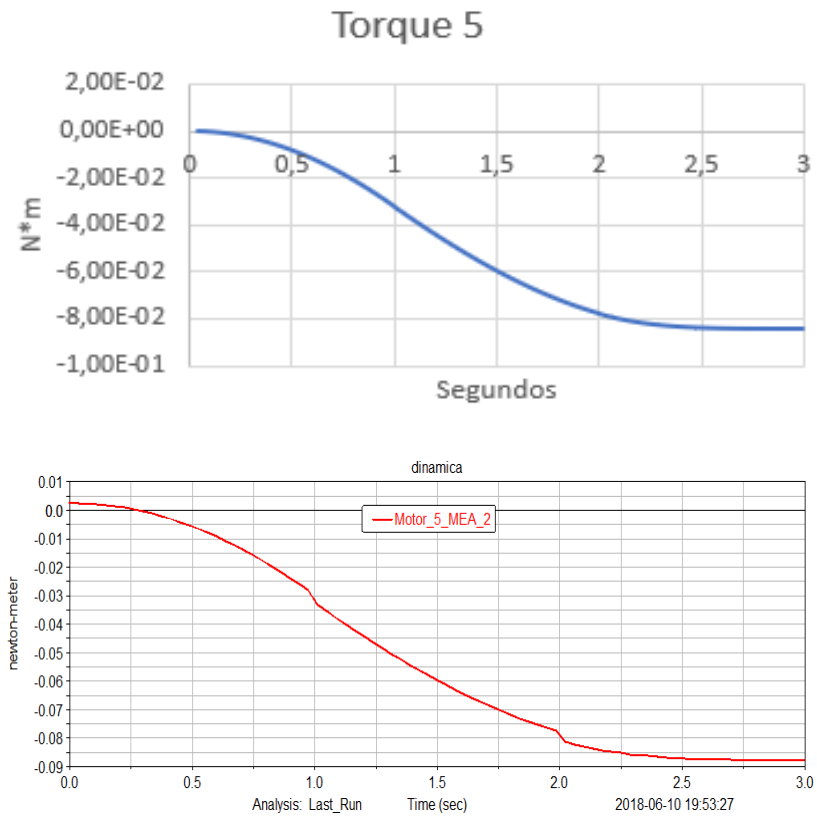


Figura 73: Par motor 5

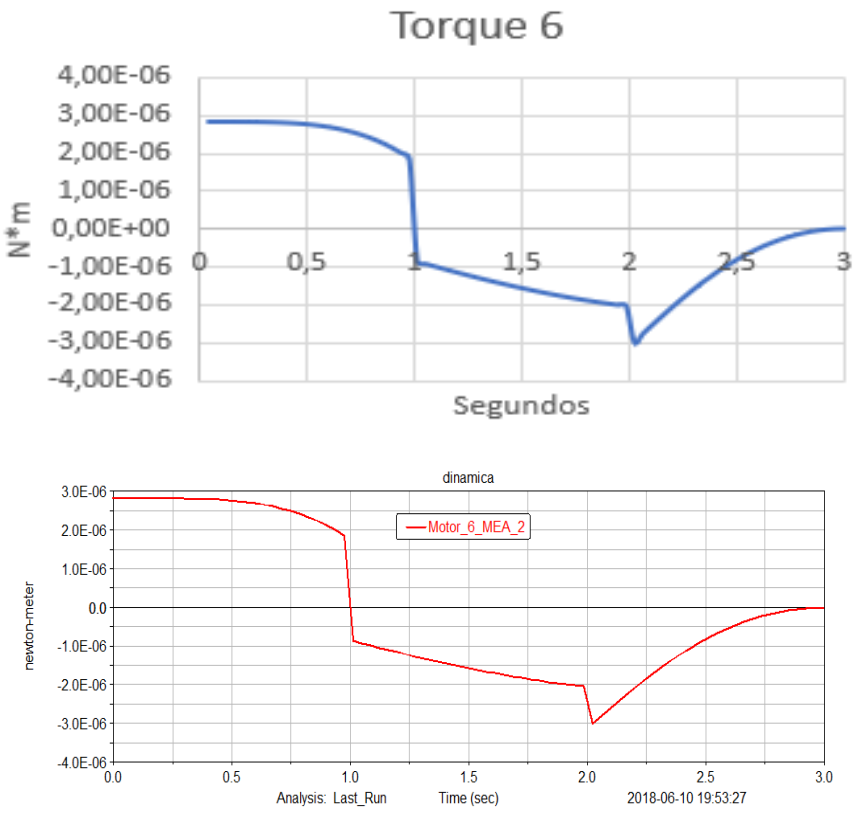


Figura 74: Par motor 6

Como se puede observar los resultados tanto de la simulación como del modelado matemático del robot PUMA 560, son coincidentes. Se ha realizado por tanto exitosamente la simulación y el modelado de la dinámica inversa, obteniendo por tanto los pares de cada uno de los respectivos motores del robot.

8 Conclusiones y futuros desarrollos

8.1 Conclusiones

Finalmente, cabe destacar que se han alcanzado los objetivos propuestos en este trabajo de fin de grado, los cuales se basaban en el estudio y simulación de la cinemática y la dinámica del brazo robot PUMA 560.

En este proyecto, se ha explicado y detallado paso a paso cómo se ha realizado dicha simulación y modelado matemático de la cinemática y dinámica, y se ha observado que los resultados coincidían en los dos programas, tanto en ADAMS View como en MATLAB. Estos resultados son coincidentes en todos los intervalos de tiempo especificados.

Ha sido un proyecto interesante en el cual se han utilizado muchas áreas de la ingeniería relacionándolas y trabajando con ellas. La importancia del presente proyecto se basa en el refuerzo de todos los conceptos de mecánica vectorial, localización espacial, cinemática y dinámica, así como el aprendizaje del software utilizado tanto de simulación como del software matemático.

Únicamente resta remarcar que ha sido un proyecto donde he reforzado y ampliado con creces los conceptos de programación, así como muchos otros conocimientos adquiridos durante la carrera enfocados al área de la dinámica y cinemática de los sistemas multicuerpo.

8.2 Futuros desarrollos

En este proyecto se ha trabajado con un robot antiguo, pero se han sentado las bases de un programa matemático que se puede desarrollar en MATLAB como herramienta de cálculo para una amplia gama de robots, realizando mínimas modificaciones, así como el uso del software ADAMS View para la simulación de cualquier mecanismo. Por otro lado, se ha realizado este proyecto de manera que pueda servir de algún modo como incentivo para ampliar la docencia de este campo de la ingeniería.

9 Presupuesto

En este apartado, se va a proceder al estudio económico que nos permitirá valorar cual es el presupuesto del presente proyecto. Para ello se realizarán una serie de consideraciones expuestas a continuación.

Para la correcta realización de dicho proyecto, se supondrá necesaria la compra del equipo que utilizaremos, así como los gastos de trabajo que tienen lugar en una oficina técnica.

Por lo tanto, tendremos en cuenta los siguientes apartados: Mano de obra y costes del equipo empleado.

9.1 Mano de obra

En el concepto de mano de obra, se tienen referidos los costes de un ingeniero trabajando como proyectista.

1. Tiempo de realización.

   El proyecto se ha realizado en un espacio de tiempo de aproximadamente 5 meses, tomando una media de 23 días trabajados por mes con una dedicación aproximada de 5 horas por día tendremos:

   4x5x23=460 horas de trabajo

   Basándose en el mercado laboral actual, estipularemos un precio horario de 16 euros por lo que tendremos el siguiente coste:

   460 horas x 16€/hora = 7360€

2. Pagos a la Seguridad Social.

   Estos costes se calculan en función de la cotización a la Seguridad social dependiendo de la categoría profesional.

   El coste de la mano de obra directa por los pagos a la Seguridad Social se obtiene de una manera muy sencilla, tendremos que multiplicar el número de cotización a la Seguridad Social dentro de la categoría profesional. En nuestro caso la base de cotización será de 4,5€/hora con lo que tendremos el siguiente cálculo.

   460 horas x 4,5€/hora = 2070€

   Como no se considerarán pagas extraordinarias y este proyecto solo ha sido desempeñado por un ingeniero, no se supondrá un coste de mano de obra directa, por lo que el coste total referido a la mano de obra será la suma de los apartados anteriores. Coste de la mano de obra= 9430€

9.2 Costes de equipo empleado

Para este proyecto se ha necesitado:

Ordenador portátil HP = 640 €
Licencia Microsoft Office = Incluida con el ordenador.
Licencia de Microsoft Windows= Incluida con el ordenador.
Licencia anual ADAMS View = Versión estudiante gratuita.
Licencia anual MATLAB estándar = Versión estudiante gratuita.
Amortización a 3 años portátil HP= (640 € / 3años) = 213,33 €/año
213,33 €/año / (365 días x 8 horas) = 0,0729 €/hora.
0,0729 €/hora*460 horas=33.53€.
Precios de equipos = 640-33.53=606.47 €

9.3 Costes totales

Los costes totales de este proyecto se obtendrán mediante la suma de los diferentes apartados que se han ido explicando anteriormente. A continuación, se adjunta el total del proyecto:

Costes de mano de obra 9430€
Precios de equipos 606.47€

Costes totales 10036.47€
13% Gastos generales 1304.74€
6% Beneficio Industrial 602.19€
Presupuesto base 11943.4€

21% IVA 2508.114€

Presupuesto total 14451.514€

10 Bibliografía

En el siguiente apartado se adjunta la bibliografía consultada para la realización del proyecto.

1. Mecánica de Robots – Vicente Mata Amela, Francisco Valero Chuliá, Juanignacio Cuadrado Iglesias - Servicio de Publicaciones Colección: Libro-Apunte número 16

2. MATLAB aplicado a la robótica y mecatrónica – Fernando Reyes Cortés – Editorial Alfaomega – Primera edición – (2012)

3. Apuntes de la signatura optativa "robótica" 4º de Ingeniería Mecánica – Universidad Politécnica de Valencia

4. Apuntes de asignatura "Dinámica del sistema Multicuerpo" 3º de Ingeniería Mecánica – Universidad Politécnica de Valencia

5. Apuntes de "robótica" Máster de Ingeniería Mecánica – Universidad Politécnica de Valencia.

6. A Search for Consensus Among Model Parameters Reported for the PUMA 560 Robot – Peter I. Corke, Brian Amstroung-Hélouvry

7. The Explicit Model and Inertial Parameters of the PUMA 560 Arm – Brian Amstrong, Oussama Khatib, Joel Burdick

Recursos electrónicos

1. https://grabcad.com/library/robot-puma-560 - Página de la cual se ha obtenido el archivo CAD del PUMA 560.

2. http://es.mathworks.com/help - Página oficial de la empresa proveedora de MATLAB – Apartado de ayuda.

3. https://www.draw.io/ - Página para diseñar diagramas de flujo.

4. https://en.wikipedia.org/wiki/Programmable_Universal_Machine_for_Assembly - Página de la Wikipedia relativa al PUMA.

A continuación, se muestra una tabla en la cual se cita la fuente de cada Figura utilizada en este proyecto.

Figura	Fuente
Figura 1: Ángulos de Euler ZXZ	Apuntes de Robótica, Máster en Ingeniería Mecánica
Figura 2: Ángulos de Euler ZYZ	Apuntes de Robótica, Máster en Ingeniería Mecánica
Figura 3: Alabeo, cabeceo y guiñada	Apuntes de Robótica, Máster en Ingeniería Mecánica
Figura 4: Transformación de coordenadas	Apuntes de Robótica, Máster en Ingeniería Mecánica
Figura 5: Cadena cinemática abierta	Apuntes de Robótica, Máster en Ingeniería Mecánica
Figura 6: Notación DH criterio de Paul para un par de revolución	Apuntes de Robótica, Máster en Ingeniería Mecánica
Figura 7: Notación DH eje X	Apuntes de Robótica, Máster en Ingeniería Mecánica
Figura 8: Notación DH eje Y	Apuntes de Robótica, Máster en Ingeniería Mecánica
Figura 9: Parámetros DH	Apuntes de Robótica, Máster en Ingeniería Mecánica
Figura 10: Aplicación método DH	Apuntes de Robótica, Máster en Ingeniería Mecánica
Figura 11: Cinemática directa, cinemática inversa	Apuntes de Robótica, Máster en Ingeniería Mecánica
Figura 12: Localización elemento terminal	Apuntes de Robótica, Máster en Ingeniería Mecánica
Figura 13: Muñeca esférica	Apuntes de Robótica, Máster en Ingeniería Mecánica
Figura 14: Posibles configuraciones del PUMA 560	Apuntes de Robótica, Optativa 4º de Ingeniería Mecánica
Figura 15: Representación vectores translación y rotación infinitesimal efector final	Apuntes de Robótica, Máster en Ingeniería Mecánica
Figura 16: Trayectoria de segmentos lineales con tramos parabólicos	Apuntes de Robótica, Máster en Ingeniería Mecánica
Figura 17: Dinámica de robots	Apuntes de Robótica, Máster en Ingeniería Mecánica
Figura 18: Vector $^{0}r_{OiP}$ representado en el PUMA 560	Apuntes de Robótica, Máster en Ingeniería Mecánica

Figura 19: Dimensiones principales y sistemas de referencia del PUMA 560	Mecánica de Robots, Vicente Mata, Juan Ignacio Cuadrado Iglesias, Francisco Valero Chuliá
Figura 20: DH PUMA 560 ADAMS View View vista 3D	Fuente Propia
Figura 21: DH PUMA 560 ADAMS View View vista alzado	Fuente Propia
Figura 22: DH PUMA 560 ADAMS View View vista planta	Fuente Propia
Figura 23: Características de los pares en ADAMS View	Fuente Propia
Figura 24: Función del primer motion en ADAMS View	Fuente Propia
Figura 25: Desplazamiento en X MATLAB y ADAMS View	Fuente propia
Figura 26: Desplazamiento en Y MATLAB y ADAMS View	Fuente propia
Figura 27: Desplazamiento en Z MATLAB y ADAMS View	Fuente propia
Figura 28: Ángulo Theta MATLAB y ADAMS View	Fuente propia
Figura 29: Ángulo PHI MATLAB y ADAMS View	Fuente propia
Figura 30: Ángulo PSI MATLAB y ADAMS View	Fuente propia
Figura 31: Velocidad lineal en X MATLAB y ADAMS View	Fuente propia
Figura 32: Velocidad lineal en Y MATLAB y ADAMS View	Fuente propia
Figura 33: Velocidad lineal en Z MATLAB y ADAMS View	Fuente propia
Figura 34: Velocidad angular en X MATLAB y ADAMS View	Fuente propia
Figura 35: Velocidad angular en Y MATLAB y ADAMS View	Fuente propia
Figura 36: Velocidad angular en Z MATLAB y ADAMS View	Fuente propia
Figura 37: Aceleración lineal en X MATLAB y ADAMS View	Fuente propia
Figura 38: Aceleración lineal en Y MATLAB y ADAMS View	Fuente propia
Figura 39: Aceleración lineal en Z MATLAB y ADAMS View	Fuente propia
Figura 40: Aceleración angular en X MATLAB y ADAMS View	Fuente propia
Figura 41: Aceleración angular en Y MATLAB y ADAMS View	Fuente propia
Figura 42: Aceleración angular en Z MATLAB y ADAMS View	Fuente propia
Figura 43: Unidades de medida utilizadas	

Figura 44: Desplazamiento en X MATLAB y ADAMS View	Fuente propia
Figura 45: Desplazamiento en Y MATLAB y ADAMS View	Fuente propia
Figura 46: Desplazamiento en Z MATLAB y ADAMS View	Fuente propia
Figura 47: Ángulo Theta MATLAB y ADAMS View	Fuente propia
Figura 48: Ángulo PHI MATLAB y ADAMS View	Fuente propia
Figura 49: Ángulo PSI MATLAB y ADAMS View	Fuente propia
Figura 50: Velocidad lineal en X MATLAB y ADAMS View	Fuente propia
Figura 51: Velocidad lineal en Y MATLAB y ADAMS View	Fuente propia
Figura 52: Velocidad lineal en Z MATLAB y ADAMS View	Fuente propia
Figura 53: Velocidad angular en X MATLAB y ADAMS View	Fuente propia
Figura 54: Velocidad angular en Y MATLAB y ADAMS View	Fuente propia
Figura 55: Velocidad angular en Z MATLAB y ADAMS View	Fuente propia
Figura 56: Aceleración lineal en X MATLAB y ADAMS View	Fuente propia
Figura 57: Aceleración lineal en Y MATLAB y ADAMS View	Fuente propia
Figura 58: Aceleración lineal en Z MATLAB y ADAMS View	Fuente propia
Figura 59: Aceleración angular en X MATLAB y ADAMS View	Fuente propia
Figura 60: Aceleración angular en Y MATLAB y ADAMS View	Fuente propia
Figura 61: Aceleración angular en Z MATLAB y ADAMS View	Fuente propia
Figura 62: Unidades de medida utilizadas en la cinemática inversa	Fuente Propia
Figura 63: Masa de las barras en kg	A Search for Consensus Among Model Parameters Reported for the PUMA 560 Robot – Peter I. Corke, Brian Amstroung-Hélouvry
Figura 64: Localización relativa de los Centros de Masas respecto a los S.D.R. de cada barra (mm)	A Search for Consensus Among Model Parameters Reported for the PUMA 560 Robot – Peter I. Corke, Brian Amstroung-Hélouvry
Figura 65: Momentos de inercia respecto a los Cetros de Masas de cada barra (Kg*m^2)	A Search for Consensus Among Model Parameters Reported for the PUMA 560

	Robot – Peter I. Corke, Brian Amstroung-Hélouvry
Figura 66: Parámetros másicos e inerciales en ADAMS View	Fuente Propia
Figura 67: Localización y orientación C.d.M en ADAMS View	Fuente Propia
Figura 68: Unidades de medida para la simulación dinámica	Fuente Propia
Figura 69: Torque motor 1	Fuente propia
Figura 70: Torque motor 2	Fuente propia
Figura 71: Torque motor 3	Fuente propia
Figura 72: Torque motor 4	Fuente propia
Figura 73: Torque motor 5	Fuente propia
Figura 74: Torque motor 6	Fuente propia