Trabajo Fin de Grado
Grado en Ingenieria en Tecnologías Industriales

Análisis Computacional Cinemático de Mecanismos Planos

Autor: Jesús Medrán Castro
Tutor: Carmen Madrigal Sánchez

Dep. Ingeniería Mecánica y Fabricación
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016

Trabajo Fin de Grado
Grado en Ingeniería en Tecnologías Industriales

Análisis Computacional Cinemático de Mecanismos Planos

Autor:
Jesús Medrán Castro

Tutor:
Carmen Madrigal Sánchez
Profesor titular

Dep. de Ingeniería Mecánica y Fabricación
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2016

Proyecto Fin de Carrera: Análisis Computacional Cinemático de Mecanismos Planos

Autor: Jesús Medrán Castro

Tutor: Carmen Madrigal Sánchez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2016

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

Dedico el presente trabajo a todas aquellas personas que siempre me han ayudado, mi familia y amigos. Por estar siempre a mi lado y por su constante apoyo y consejos que he recibido. También agradecer a todos aquellos profesores por guiarme y enseñarme.

Jesús Medrán Castro

Alumno del grado de Ingeniería en Tecnologías Industriales

Sevilla, 2016

Resumen

El objetivo principal de este documento es el estudio de la cinemática de mecanismos planos. Para ello se efectuará un análisis físico y matemático de los mecanismos planos con el fin de entender su esencia. Se buscarán generalidades y patrones que ayuden a la creación de un programa informático con el software de programación comercial *Matlab*. Finalmente se describirá cómo realizar un programa capaz de resolver la cinemática de mecanismos planos para ayudar a entender mejor cómo funcionan estos mecanismos, afianzar los conocimientos sobre esta materia y servir como asistente de resolución para comprobar los resultados obtenidos del cálculo a mano.

Índice

ÍNDICE DE FIGURAS

Notación

$\vec{\imath}, \vec{\jmath}, \vec{k}$	Vectores unitarios en las 3 direcciones de los ejes coordenados
$\overrightarrow{OP}$	Vector que va del punto O al punto P
dt	Derivada temporal
$\vec{r}_{i1}^{P}$	Vector de posición del punto P perteneciente al cuerpo i con respecto al cuerpo 1
$\vec{V}_{i1}^{P}$	Vector de velocidad lineal del cuerpo i en el punto P con respecto al cuerpo 1
$\vec{\omega}_{i1}$	Vector velocidad angular del cuerpo i con respecto al cuerpo 1
$\vec{a}_{i1}^{P}$	Vector de aceleración lineal del cuerpo i en el punto P con respecto al cuerpo 1
$\vec{\alpha}_{i1}$	Vector aceleración angular del cuerpo i con respecto al cuerpo 1
$\wedge$	Producto vectorial
sen	Función seno
tg	Función tangente
cos	Función coseno
$\overrightarrow{h_{p_i}}$	Vector de dirección perpendicular al par prismático 1 del cuerpo i
$\square, \triangle, O$	Representación esquemática e los pares prismáticos, de rodadura sin deslizamiento y de rotación respectivamente
$\forall$	Para todo
$\cdots$	Continuación
$\sum_{i,\ j}^{n,\ m}$	Sumatorio desde el término i hasta el término n y desde el término j al término m
$[m\ x\ n]$	Dimensiones de una matriz de m filas y n columnas
$\leq$	Menor o igual
$\geq$	Mayor o igual

1 INTRODUCCIÓN: OBJETIVO Y ALCANCE

Las máquinas no existirían sin nosotros, pero nuestra existencia ya no es posible sin ellas.

- Munari, Bruno -

El presente trabajo se presenta como un análisis de la cinemática de los mecanismos planos. Se estudia los pares cinemáticos más comunes que existen en el plano y que forman estos mecanismos. El objetivo principal es la búsqueda de la generalización de la resolución de los problemas de velocidad y aceleración de cualquier mecanismo que pueda modelizarse en el plano. Todo esto servirá a posteriori para la creación de un script de código ejecutable en el programa comercial *Matlab* que sea capaz de resolver la cinemática de un mecanismo dado.

Primeramente se explicará cuáles son los pares que se estudiarán. Se analizará su física tanto en términos de velocidad como en términos de aceleración. Se expresarán unas ecuaciones para generalizar cada par a cualquier tipo de cuerpo, de esta manera será más fácil la generalización de un mecanismo completo (un mecanismo es, al fin y al cabo, un conjunto de cuerpos unidos mediante diferentes pares cinemáticos). Los pares que se estudiarán serán tres: Par de rotación, par prismático y par de rodadura sin deslizamiento. Esto querrá decir que los mecanismos que se contemplarán en este documento solo podrán poseer los pares cinemáticos descritos o cualquier par cinemático cuya cinemática pueda modelarse (o simularse) por uno de estos tres pares anteriormente descritos.

En el capítulo 3 se analizará cómo plantear las ecuaciones que rigen la cinemática de un mecanismo genérico de manera que se pueda generalizar para cualquier mecanismo. De esta forma dado un *lazo de los pares* de un mecanismo (término que se explicará en profundidad en dicho capítulo) y la geometría completa, se es capaz de plantear todas las ecuaciones de velocidad y aceleración del mecanismo. Estas ecuaciones, como ya se sabe, poseen operaciones tales como productos vectoriales, productos escalares, sumas, restas, términos elevados al cuadrado y demás cálculos relativamente complejos. Se habla de esto ya que el objetivo último de este trabajo es la creación de un programa en *Matlab* capaz de resolver esas ecuaciones. *Matlab* es un programa de cálculo que está especializado en el uso y manejo de matrices y operaciones con matrices por tanto estas ecuaciones no podrá resolverlas de manera sencilla debido a su grado de complejidad.

Es aquí donde entra en juego el capítulo 4 donde se tratará de transformar estas ecuaciones a un formato matricial compatible con las capacidades de *Matlab*. Las ecuaciones que rigen la cinemática de los mecanismos se trasformarán a un formato matricial en el cual *Matlab* trabaja muy bien. Este proceso de transformación nos llevará a darnos cuenta de cómo actúan y que papel tienen cada par cinemático en las ecuaciones del mecanismo. Una vez 'traducido' el problema a un formato en el que *Matlab* pueda resolverlo es necesario crear un programa que sea capaz de hacerlo.

El código del programa se escribirá íntegramente en *Matlab*. Todas las funciones, cálculos y operaciones se

realizarán con *Matlab*. El problema completo posee un número considerable de variables de todo tipo, por lo que se hace necesario la creación de un método de organización de estos datos y cómo analizarlos. Se ha creado la estructura '*Meca*' en la cual se administran todas las variables del problema y hará más sencillo su transporte a lo largo de todos los subprogramas necesarios para la resolución del problema. Se debe definir también una secuencia de operaciones a seguir para el correcto desarrollo del programa. Es aquí donde se plantea la necesidad de un método de introducción de los datos necesarios para el cálculo del problema (geometría, velocidades y aceleraciones iniciales, etc.). Es por ello que se crea una interfaz que guie y ayude al usuario de cómo implementar el mecanismo que este quiere resolver. El programa creado se ha denominado *CAC 2D* (*Computational Analysis Cinematic of Mechanical 2D*), será este el encargado de recoger los datos, analizarlos y resolver el problema.

CAC 2D se plantea como un programa creado con la ayuda de *Matlab* que es capaz de recoger y analizar los datos del mecanismo al igual que de plantear sus ecuaciones y resolverlas. Pero para mejorar el uso de este programa por parte del usuario se ha incluido funciones adicionales muy útiles que permiten mejorar la experiencia de uso. Se han incluido funciones tales como poder guardar y cargar los datos de un mecanismo, la representación del esquema del mecanismo junto con su correspondiente campo de velocidades y aceleraciones, la creación automática de informes en pdf sobre los resultados del mecanismo, un manual de usuario para guiar a usuarios inexpertos y la simulación del movimiento natural del mecanismo entre otros.

Este programa debe ejecutarse en *Matlab 2012a* o versiones superiores siendo necesario siempre que *Matlab* este instalado en el equipo previamente. Por ello se ha creado también una versión de *CAC 2D* capaz de ser reproducida sin *Matlab* es decir que no necesita tener instalado *Matlab* en el equipo. Esta versión se basa en la aplicación *MATLAB Compiler Runtime (MCR)* que es capaz de realizar todas las operaciones y cálculos sin necesidad de tener instalado *Matlab*. Esta opción con MCR garantiza todas las operaciones del programa *CAC 2D* excepto una, la creación de informes en pdf. Debido a las licencias que usa MCR no puede crear documento pdf por lo que esta versión de *CAC 2D* no podrá crear informes del mecanismo.

2 BASES TEÓRICAS

Se define como Cinemática (del vocablo griego kineo, que significa movimiento) al estudio del movimiento independientemente de las fuerzas que lo producen. Se trata de una rama de la mecánica que se centra en estudiar las leyes del movimiento de los objetos sólidos sin considerar las causas que lo originan. La velocidad y la aceleración son las dos magnitudes principales en este estudio.

El conocimiento previo sobre la cinemática de mecanismos planos es muy amplio. Gran parte de la Física estudiada por el ser humano se ha centrado en el movimiento y relación de los sólidos. Estos conocimientos han servido para la creación de máquinas y mecanismos que estuvieran al servicio del hombre para mejorar su calidad de vida y la de su entorno. La mayoría de mecanismos existentes puede modelizarse de forma esquemática en el plano, para que su análisis inicial sea más fácil y ameno. Es aquí donde la cinemática de estos mecanismos planos cobra mucha importancia ya que presenta un método de análisis de movimiento capaz de predecir velocidades, aceleraciones e incluso futuras posiciones que puede adoptar el mecanismo.

Prácticamente todos los mecanismos que existen a nuestro alrededor han sido analizados cinemáticamente para su diseño. Todas las velocidades y aceleraciones que puedan llegar a alcanzar se han estudiado previamente gracias al análisis cinemático.

2.1. Método de las velocidades y aceleraciones relativas

Se considera un sólido rígido al que denominaremos *sólido 2* caracterizado por un triedro trirrectángulo de vectores unitarios $\{\vec{i}_2,\ \vec{j}_2 \text{ y } \vec{k}_2\}$. Por otro lado consideraremos un sólido fijo e inmóvil (*sólido 1*) con vectores unitarios $\{\vec{i},\ \vec{j} \text{ y } \vec{k}\}$ respecto del cual estudiaremos el movimiento del *sólido 2*.

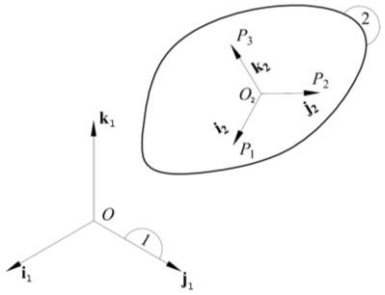


Figura 1

Se puede deducir que:

$$\vec{\imath}_2 = \overrightarrow{OP_1} - \overrightarrow{OO_2}$$

Derivando con respecto del tiempo se obtiene:

$$\frac{d\vec{\imath}_2}{dt} = \vec{V}_{21}^{P_2} - \vec{V}_{21}^{O_2}$$

La diferencia de velocidades de dos puntos de un sólido rígido están relacionados mediante la expresión:

$$\vec{V}_{21}^{P_2} = \vec{V}_{21}^{O_2} + \vec{\omega}_{21} \wedge \vec{\imath}_2$$

Aplicando las dos ecuaciones anteriormente descritas:

$$\frac{d\vec{\imath}_2}{dt} = \vec{\omega}_{21} \wedge \vec{\imath}_2$$

Generalizando este proceso para los otros dos vectores unitarios se llega a las conocidas como *fórmulas de Poisson.* A partir de ellas se puede calcular la expresión de la derivada con respecto al tiempo de cualquier vector ligado al *sólido 2* con referencia al *sólido 1*.

$$\begin{cases} \dfrac{d\vec{\imath}_2}{dt} = \vec{\omega}_{21} \wedge \vec{\imath}_2 \\ \dfrac{d\vec{\jmath}_2}{dt} = \vec{\omega}_{21} \wedge \vec{\jmath}_2 \\ \dfrac{d\vec{k}_2}{dt} = \vec{\omega}_{21} \wedge \vec{k}_2 \end{cases}$$

Estas ecuaciones serán muy útiles a la hora de averiguar las velocidades de los puntos de eslabones unidos entre sí.

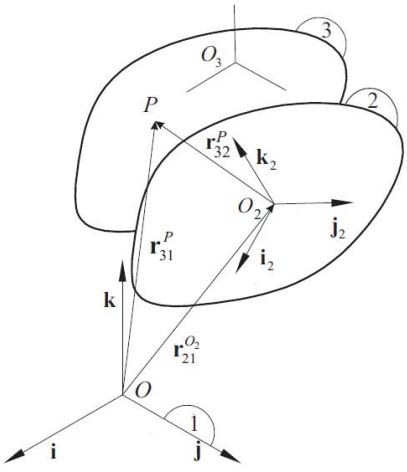


Figura 2

Para referirnos al vector de posición de un punto P perteneciente a un *sólido 3* y expresado respecto el *sólido 1* (eslabón fijo) usaremos la notación $\vec{r}_{31}^{P}$. La veriación en el tiempo de este vector describirá el movimiento de absoluto del punto P. Las derivadas con el tiempo de este vector serán las velocidades y aceleraciones absolutas de este punto P.

$$\left.\frac{d\vec{r}_{31}^{P}}{dt}\right|_1 = \vec{V}_{31}^{P} \quad ; \quad \left.\frac{d\vec{V}_{31}^{P}}{dt}\right|_1 = \left.\frac{d^2\vec{r}_{31}^{P}}{dt^2}\right|_1 = \vec{a}_{31}^{P}$$

De la misma forma si queremos expresar la posición del punto P perteneciente al *sólido 3* con respecto al *sólido 2* lo haremos con el término $\vec{r}_{32}^{P}$. Esto será la posición relativa del punto P con respecto al *sólido 2* y por tanto sus derivadas serán las velocidades y aceleraciones relativas al *sólido 2*.

$$\left.\frac{d\vec{r}_{32}^{P}}{dt}\right|_2 = \vec{V}_{32}^{P} \quad ; \quad \left.\frac{d\vec{V}_{32}^{P}}{dt}\right|_2 = \left.\frac{d^2\vec{r}_{31}^{P}}{dt^2}\right|_2 = \vec{a}_{32}^{P}$$

De la aplicación de estas reglas junto con las *fórmulas de Poisson* puede deducirse las velocidades lineales (y angulares) y las aceleraciones lineales (y angulares) de eslabones de un mismo mecanismo. Siendo así se explica la deducción de las velocidades lineales relativas a tres cuerpos. Las demás relaciones se expondrán ya deducidas ya que son muy parecidas a la primera deducción.

Siendo la igualdad vectorial:

$$\vec{r}_{31}^{P} = \vec{r}_{32}^{P} + \vec{r}_{21}^{O_2}$$

Derivando esta expresión con respecto al tiempo:

$$\left.\frac{d\vec{r}_{31}^{P}}{dt}\right|_1 = \left.\frac{d\vec{r}_{32}^{P}}{dt}\right|_1 + \left.\frac{d\vec{r}_{21}^{O_2}}{dt}\right|_1 \longrightarrow \vec{V}_{31}^{P} = \left.\frac{d\vec{r}_{32}^{P}}{dt}\right|_1 + \vec{V}_{21}^{O_2}$$

El primer término de la suma de la parte derecha de la igualdad puede desarrollarse:

$$\left.\frac{d\vec{r}_{32}^{P}}{dt}\right|_1 = \left.\frac{d\vec{r}_{32}^{P}}{dt}\right|_2 + \vec{\omega}_{21} \wedge \vec{r}_{32}^{P} = \vec{V}_{32}^{P} + \vec{\omega}_{21} \wedge \vec{r}_{32}^{P}$$

Si reagrupamos y ordenamos términos:

$$\vec{V}_{31}^{P} = \vec{V}_{32}^{P} + \vec{V}_{21}^{O_2} + \vec{\omega}_{21} \wedge \vec{r}_{32}^{P} \longrightarrow \vec{V}_{31}^{P} = \vec{V}_{32}^{P} + \vec{V}_{21}^{P}$$

La expresión anterior se puede generalizar y obtener:

$$\vec{V}_{ij}^{P} = \vec{V}_{ik}^{P} + \vec{V}_{kj}^{P}$$

A continuación pasamos a mostrar todas las restricciones de las velocidades y aceleraciones relativas de los eslabones de los mecanismos planos. Estas ecuaciones serán la base para el posterior cálculo de las relaciones de velocidad y aceleración de los eslabones de los mecanismos.

- Velocidades lineales

$$\vec{V}_{ij}^{P} = \vec{V}_{ik}^{P} + \vec{V}_{kj}^{P} \quad ; \quad \vec{V}_{i1}^{P} = \vec{V}_{i1}^{Q} + \vec{\omega}_{i1} \wedge \overrightarrow{QP}$$

- Velocidades angulares

$$\vec{\omega}_{ij} = \vec{\omega}_{ik} + \vec{\omega}_{kj}$$

- Aceleraciones lineales

$$\vec{a}_{ij}^{P} = \vec{a}_{ik}^{P} + \vec{a}_{kj}^{P} + 2\vec{\omega}_{kj} \wedge \vec{V}_{ik}^{P} \quad ; \quad \vec{a}_{i1}^{P} = \vec{a}_{i1}^{Q} + \vec{\alpha}_{i1} \wedge \overrightarrow{QP} - \vec{\omega}_{i1}^{2}\overrightarrow{QP}$$

- Aceleraciones angulares

$$\vec{\alpha}_{ij} = \vec{\alpha}_{ik} + \vec{\alpha}_{kj} + \vec{\omega}_{kj} \wedge \vec{\omega}_{ik} \xrightarrow{\text{En el caso plano}} \vec{\alpha}_{ij} = \vec{\alpha}_{ik} + \vec{\alpha}_{kj}$$

2.2. Cinemática del plano

Como ya se ha adelantado, los pares cinemáticos que se estudiarán serán tres específicos. A continuación se expone la física de cada par y cuáles son las restricciones que este impone en velocidad y aceleración. A su vez también se explica cómo generalizar cada par a dos cuerpos (i y j) cualesquiera. Estas ecuaciones serán de mucha ayuda a la hora de plantear el conjunto de ecuaciones de un mecanismo.

2.2.1 Par de rotación

Un par de cuerpos (o eslabones) unidos mediante un par de rotación comparten una serie de restricciones debidas a la naturaleza de dicho par. Estas restricciones son referentes al movimiento de una rotación de un cuerpo.

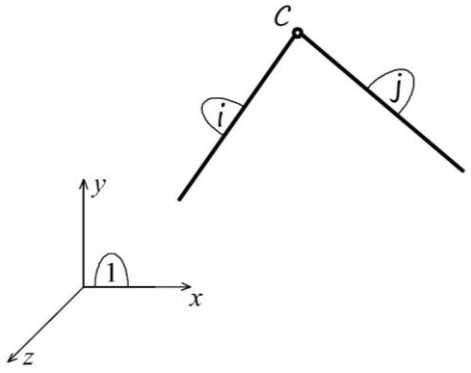


Figura 3

Las velocidades y aceleraciones relativas en la articulación son cero. Por consiguientes sus equivalentes también conforman la restricción de este par. Es decir, que la velocidad y aceleración relativa a cada cuerpo en el punto donde se encuentra el par de rotación es cero.

$$\vec{V}_{ij}^{C} = 0 \quad ; \quad \vec{a}_{ij}^{C} = 0$$

Su generalización para un par de cuerpos cualquiera sería:

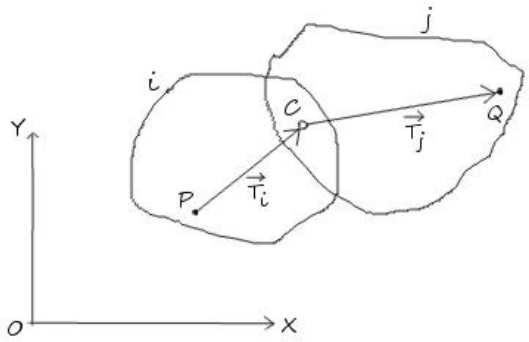


Figura 4

· Velocidad

Este par restringe la velocidad del punto de contacto de los dos cuerpos para que sean completamente iguales (en módulo dirección y sentido).

$$\begin{cases} \vec{V}_{i1}^{C} = \vec{V}_{i1}^{P} + \vec{\omega}_{i1} \wedge \overrightarrow{T_i} \\ \vec{V}_{j1}^{C} = \vec{V}_{j1}^{Q} - \vec{\omega}_{j1} \wedge \overrightarrow{T_j} \end{cases}$$

$$\underline{Cóndición\ del\ par}$$
$$\vec{V}_{i1}^{C} = \vec{V}_{j1}^{C}$$

$$\boxed{\vec{V}_{j1}^{Q} = \vec{V}_{i1}^{P} + \vec{\omega}_{i1} \wedge \overrightarrow{T_i} + \vec{\omega}_{j1} \wedge \overrightarrow{T_j}}$$

· Aceleración

Este para también impone igualdad de aceleraciones en el punto de contacto.

$$\begin{cases} \vec{a}_{i1}^{C} = \vec{a}_{i1}^{P} + \vec{\alpha}_{i1} \wedge \overrightarrow{T_i} - \vec{\omega}_{i1}^{2} \overrightarrow{T_i} \\ \vec{a}_{j1}^{C} = \vec{a}_{j1}^{Q} - \vec{\alpha}_{j1} \wedge \overrightarrow{T_j} + \vec{\omega}_{j1}^{2} \overrightarrow{T_j} \end{cases}$$

$$\underline{Cóndición\ del\ par}$$
$$\vec{a}_{i1}^{C} = \vec{a}_{j1}^{C}$$

$$\boxed{\vec{a}_{j1}^{Q} = \vec{a}_{i1}^{P} + \vec{\alpha}_{i1} \wedge \overrightarrow{T_i} + \vec{\alpha}_{j1} \wedge \overrightarrow{T_j} - \vec{\omega}_{i1}^{2} \overrightarrow{T_i} - \vec{\omega}_{j1}^{2} \overrightarrow{T_j}}$$

2.2.2 Par prismático

Las restricciones que impone un par prismático consisten en que el movimiento relativo de los dos eslabones debe realizarse en la dirección de la guía.

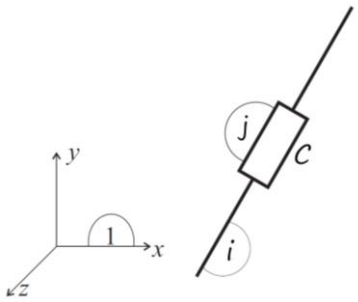


Figura 5

Su generalización para un par de cuerpos cualquiera sería:

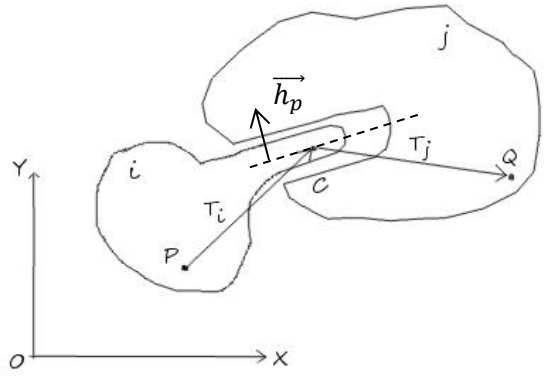


Figura 6

· <u>Velocidad</u>

Este par impone que las velocidades angulares de ambos eslabones sean las mismas y además impone que las componentes de las velocidades lineales perpendiculares a la dirección del par deben coincidir.

$$\begin{cases} \vec{V}_{i1}^{C} = \vec{V}_{i1}^{P} + \vec{\omega}_{i1} \wedge \overrightarrow{T_i} \\ \vec{V}_{j1}^{C} = \vec{V}_{j1}^{Q} - \vec{\omega}_{j1} \wedge \overrightarrow{T_j} \end{cases}$$

<u>Cóndición del par</u>

$$\begin{cases} \left(\vec{V}_{i1}^{C}\right) \cdot \overrightarrow{h_{p_i}} = \left(\vec{V}_{j1}^{C}\right) \cdot \overrightarrow{h_{p_i}} \\ \vec{\omega}_{i1} = \vec{\omega}_{j1} \end{cases}$$

$$\left(\vec{V}_{i1}^{P} + \vec{\omega}_{i1} \wedge \overrightarrow{T_i}\right) \cdot \overrightarrow{h_{p_i}} = \left(\vec{V}_{j1}^{Q} - \vec{\omega}_{j1} \wedge \overrightarrow{T_j}\right) \cdot \overrightarrow{h_{p_i}}$$
$$\vec{\omega}_{i1} = \vec{\omega}_{j1}$$

· <u>Aceleración</u>

En el caso de aceleración se imponen unas condiciones análogas a las impuestas en velocidad. Las aceleraciones angulares deben ser iguales y las componentes de las aceleraciones lineales perpendiculares a la dirección del par deben ser también iguales.

$$\begin{cases} \vec{a}_{i1}^{C} = \vec{a}_{i1}^{P} + \vec{\alpha}_{i1} \wedge \overrightarrow{T_i} - \vec{\omega}_{i1}^{2}\overrightarrow{T_i} \\ \vec{a}_{j1}^{C} = \vec{a}_{j1}^{Q} - \vec{\alpha}_{j1} \wedge \overrightarrow{T_j} + \vec{\omega}_{j1}^{2}\overrightarrow{T_j} \end{cases}$$

Cóndición del par

$$\begin{cases} (\vec{a}_{i1}^{C}) \cdot \overrightarrow{h_{p_i}} = (\vec{a}_{j1}^{C}) \cdot \overrightarrow{h_{p_i}} \\ \vec{\alpha}_{i1} = \vec{\alpha}_{j1} \end{cases}$$

$$\begin{gathered} (\vec{a}_{i1}^{P} + \vec{\alpha}_{i1} \wedge \overrightarrow{T_i} - \vec{\omega}_{i1}^{2}\overrightarrow{T_i}) \cdot \overrightarrow{h_{p_i}} = (\vec{a}_{j1}^{Q} - \vec{\alpha}_{j1} \wedge \overrightarrow{T_j} + \vec{\omega}_{j1}^{2}\overrightarrow{T_j}) \cdot \overrightarrow{h_{p_i}} \\ \vec{\alpha}_{i1} = \vec{\alpha}_{j1} \end{gathered}$$

2.2.3 Par de rodadura sin deslizamiento

Este tipo de par se asemeja a casos como el de engranajes o ruedas de fricción. Las restricciones que este impone en velocidad es que la velocidad relativa de ambos eslabones en el punto de contacto es cero. Esta restricción se asemeja a la impuesta por los pares de rotación anteriormente mencionados.

$$\vec{V}_{ij}^{C} = 0 \quad \rightarrow \quad \vec{V}_{i1}^{C} = \vec{V}j_{1}^{C}$$

Es por esta razón que el estudio de velocidades de eslabones unidos con un par de rodadura sin deslizamiento se puede sustituir por el estudio de los mismos eslabones pero esta vez unidos con un par de rotación. Las soluciones en velocidades de ambos problemas serán las mismas.

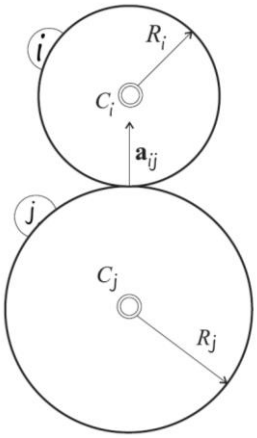


Figura 7

En cuanto a aceleración lo único que este par impone es que la aceleración relativa no es igual a cero. Es decir que el problema de aceleración en este caso no se puede asemejar al de un par de rotación.

$$\vec{a}_{ij}^{C} \neq 0$$

Existen hipótesis que se pueden aplicar a este estudio que aportan un valor de cómo sería la aceleración relativa en el punto de contacto. Esto se basa en el análisis de ruedas (o engranajes) que giran sin deslizar entre sí. Extrapolando esta idea a un cuerpo con un radio de curvatura instantáneo en cada momento se puede hacer un análisis de la aceleración relativa de este par de rodadura sin deslizamiento.

Su generalización para un par de cuerpos cualquiera sería:

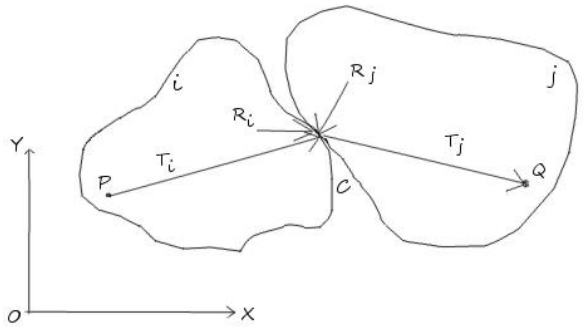


Figura 8

· Velocidad

Este par impone que la velocidad relativa del punto de contacto entre los dos eslabones sea cero. Esto hace que en cuanto al problema de velocidad las ecuaciones generales sean las mismas que el par de revolución visto anteriormente.

$$\begin{cases} \vec{V}_{i1}^{C} = \vec{V}_{i1}^{P} + \vec{\omega}_{i1} \wedge \overrightarrow{T_i} \\ \vec{V}_{j1}^{C} = \vec{V}_{j1}^{Q} - \vec{\omega}_{j1} \wedge \overrightarrow{T_j} \end{cases}$$

Cóndición del par

$$\vec{V}_{ij}^{C} = \vec{V}_{ji}^{C} = \vec{V}_{i1}^{C} - \vec{V}_{j1}^{C} = 0$$

$$\boxed{\vec{V}_{j1}^{Q} = \vec{V}_{i1}^{P} + \vec{\omega}_{i1} \wedge \overrightarrow{T_i} + \vec{\omega}_{j1} \wedge \overrightarrow{T_j}}$$

· Aceleración

En cuanto a aceleración no ocurrirá lo mismo. La aceleración lineal relativa al punto de contacto de ambos eslabones no tendrá por qué ser igual a cero. Esta aceleración relativa es conocida y dependerá únicamente de las velocidades angulares de los dos eslabones y de la disposición del par de rodadura sin deslizamiento.

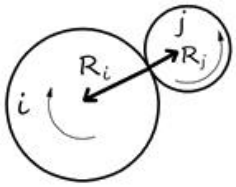
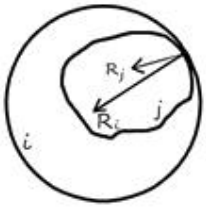
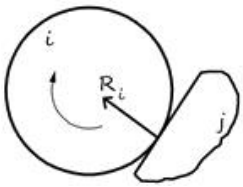
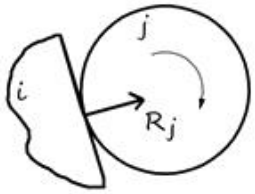
$$\begin{cases} \vec{a}_{i1}^{C} = \vec{a}_{i1}^{P} + \vec{\alpha}_{i1} \wedge \overrightarrow{T_i} - \vec{\omega}_{i1}^{2} \overrightarrow{T_i} \\ \vec{a}_{j1}^{C} = \vec{a}_{j1}^{Q} - \vec{\alpha}_{j1} \wedge \overrightarrow{T_j} + \vec{\omega}_{j1}^{2} \overrightarrow{T_j} \end{cases}$$

Cóndición del par

$$\vec{a}_{ij}^{C} = \vec{a}_{i1}^{P} - \vec{a}_{j1}^{Q} + \vec{\alpha}_{i1} \wedge \overrightarrow{T_i} + \vec{\alpha}_{j1} \wedge \overrightarrow{T_j} - \vec{\omega}_{i1}^{2} \overrightarrow{T_i} - \vec{\omega}_{j1}^{2} \overrightarrow{T_j} - 2\vec{\omega}_{j1} \wedge \vec{V}_{i1}^{C}$$

$$\vec{a}_{ij}^{C} = \left(\vec{\omega}_{i1} - \vec{\omega}_{j1}\right)^{2} \frac{R_i \, R_j}{R_i \pm R_j} \, \vec{\lambda}$$

Siendo :

Tipo 1	**Tipo 2**	**Tipo 3**	**Tipo 4**
Cóncavo - Convexo	Convexo - Cóncavo	Rueda - Plano	Plano - Rueda
			
$\vec{a}_{ij}^{C} = \left(\vec{\omega}_{i1} - \vec{\omega}_{j1}\right)^2 \frac{R_i\, R_j}{R_i - R_j}\, \vec{\lambda}$	$\vec{a}_{ij}^{C} = \left(\vec{\omega}_{i1} - \vec{\omega}_{j1}\right)^2 \frac{R_i\, R_j}{R_i + R_j}\, \vec{\lambda}$	$\vec{a}_{ij}^{C} = \left(\vec{\omega}_{i1} - \vec{\omega}_{j1}\right)^2 R_i\, \vec{\lambda}$	$\vec{a}_{ij}^{C} = \left(\vec{\omega}_{i1} - \vec{\omega}_{j1}\right)^2 R_j\, \vec{\lambda}$

Siendo $\vec{\lambda}$ el vector unitario que parte del par de rodadura sin deslizamiento y apunta al sólido i. Se observa que determinando el tipo de para de rodadura sin deslizamiento (y resolviendo previamente el problema de velocidad) se puede determinar el valor de $\vec{a}_{ij}^{C}$ para el punto de contacto.

3 MÉTODO DE PLANTEAMIENTO DE LAS ECUACIONES

El objetivo último de este documento es la creación de un código escrito en *Matlab* capaz de resolver la cinemática de mecanismos planos. Ya que se trata de un programa informático es conveniente buscar métodos de generalización del cálculo de la resolución de estos problemas. Mientras más general sea el método, más sencilla será su programación y más depurado quedará el código. Además en la búsqueda de la generalización estudiaremos mejor la matemática que rige los mecanismos planos y esto ayudará a su mejor comprensión.

3.1 Introducción al método y requisitos previos

Con motivo del estudio de los diferentes pares existentes en los mecanismos planos, sus diferentes combinaciones y con el objetivo de la generalización de las ecuaciones se expone a continuación una serie de explicaciones de como se ha llegado a generalizar la resolución del problema central de este proyecto.

Se partirán de casos particulares sencillos aumentando progresivamente la complejidad de los mecanismos hasta poder discernir una metodología general de resolución. Primero se explicará la metodología de resolución del problema de velocidad y posterior mente, cuando se llegue a la generalización del mismo se explicará la resolución del problema de aceleración.

Antes de empezar se impondrán unos requisitos básicos para el planteamiento del problema que servirán para la posterior generalización. Estos requisitos se entenderán mejor conforme se vaya avanzando en la explicación del método:

3.1.1 Requisitos

Para mejorar el proceso de generalización se exponen a continuación una serie de requisitos e hipótesis iniciales que deberán respetarse. Se trata de normar referentes a aspectos como la geometría o la definición de los pares que el usuario debe conocer a priori

- El mecanismo debe poder modelizarse en dos dimensiones. Los movimientos efectuados o con relación a una dimensión perpendicular al plano no se contemplan en este estudio.

- Debe conocerse de manera exhaustiva y completa toda la geometría básica del mecanismo. Todas las coordenadas de los pares, direcciones de los pares prismáticos, radios de curvatura, etc. deberá ser conocido.
- Los mecanismos estudiados deberán tener sentido lógico asumiendo esto como la capacidad de tener movimiento y no siendo similar a una estructura hiperestática. Como mínimo para que los resultados de esta generalización tengan sentido será poseer un mecanismo con al menos 1 grado de libertad o más.
- Los pares a estudio solo podrán ser modelados por tres tipos, par de rotación, par prismático y par de rodadura sin deslizamiento.
- Las condiciones iniciales de velocidad y aceleración deberán ser conocidas coincidiendo en número, como es lógico con los grados de libertad que posea el mecanismo.
- Todos los mecanismos deberán poseer un eslabón tipo bastidor, es decir sobre el cual se expresarán todos los términos de velocidades y aceleraciones relativas. Este eslabón se denominará con el número 1.
- No se contemplaran los pares prismáticos curvos.
- Siempre que sea posible se realizará una simplificación del mecanismo previa al estudio que este documento ofrece.

3.1.2 Definición de un grupo dentro de un lazo de un mecanismo

Llamaremos *"Grupo"* a un conjunto de cuerpos con una relación específica en un lazo. Este término se crea con la intención de mejorar y generalizar el estudio de lazos de un mecanismo. Para definir concretamente qué es un grupo dentro de un lazo se han determinado 3 reglas básicas que siempre deben cumplirse a la hora de determinar los grupos que componen un lazo.

Reglas básicas para determinar un grupo en un lazo de un mecanismo:

1. Deben existir tantos grupos como pares prismáticos tenga el lazo.
2. Cada grupo solo contendrá un solo par prismático.
3. Todos los cuerpos del mecanismo deberán formar parte de algún grupo, no se puede quedar ningún cuerpo sin formar parte de un grupo.

Para facilitar la comprensión se da un ejemplo previo de cómo se forma un grupo en un lazo sencillo. Sea el lazo dado por:

$$\int = \{1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 1\}$$

$$\int_p = \{O \quad \Delta \quad \square \quad O \quad \square \quad O \quad O\}$$

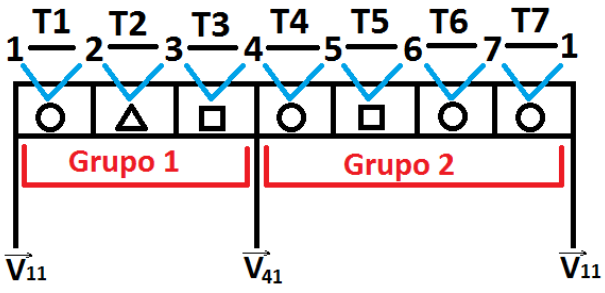


Figura 9

Como podemos observar básicamente se trata de hacer grupos siguiendo las 3 reglas anteriormente mencionadas. Visto desde otro punto de vista, se trata de hacer grupos en los que solo exista un cuadrado (Representación esquemática de un par prismático), dividiendo el mecanismo en tantos grupos como pares prismáticos tenga.

En este caso en el grupo 1 tendríamos los tramos $\overrightarrow{T_1}$, $\overrightarrow{T_2}$ y $\overrightarrow{T_3}$ y en el grupo 2 tendríamos los tramos restantes ($\overrightarrow{T_4}$, $\overrightarrow{T_5}$, $\overrightarrow{T_6}$ y $\overrightarrow{T_7}$). El grupo 1 estaría delimitado por las velocidades lineales $\vec{V}_{11}^{A}$ y $\vec{V}_{41}^{C}$ así como el grupo 2 estaría delimitado por las mismas velocidades lineales en distinto orden.

Se muestran a continuación diferentes lazos con distintas combinaciones de pares para afianzar la comprensión del lector sobre la creación de grupos.

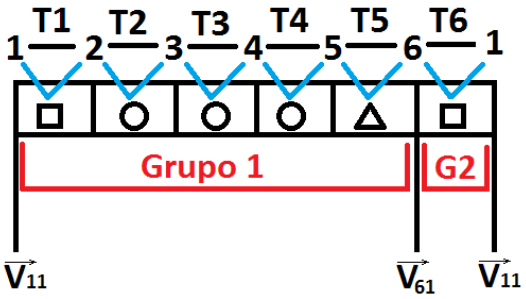


Figura 11

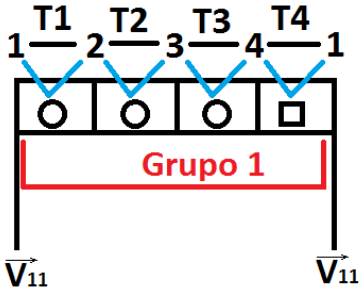


Figura 10

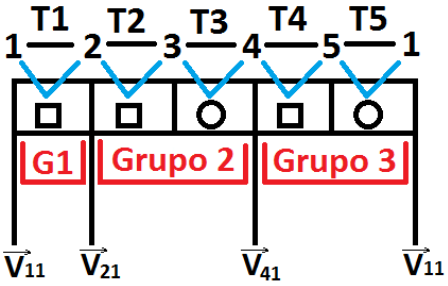


Figura 13

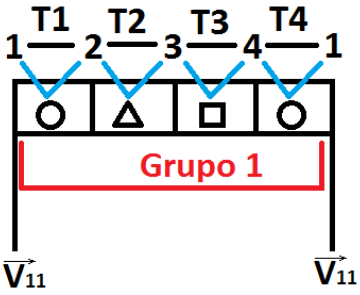


Figura 12

Una vez practicado la discretización del lazo en grupos de trabajo se continuará con la metodología para la generalización del problema de velocidad.

3.2 Definición previa de grados de libertad de un mecanismo

En este apartado se explicará cómo se calcula el número de gados de libertad de un mecanismo plano ya que resulta útil para la implantación del método de resolución.

Cualquier cuerpo que se estudie en el plano tiene, por definición 3 grados de libertad. Estos son precisamente, los dos movimientos en los dos ejes coordenados y el giro sobre el eje perpendicular al plano de estudio.

Las restricciones, en el caso que compete a este trabajo: Pares de revolución, pares prismáticos y pares de rodadura sin deslizamiento; restringen dos grados de libertad. Dicho de otra forma, cualquier par de restricción de los estudiados en este libro restará dos grados de libertad al mecanismo.

Partiendo de esa base, cualquier conjunto de cuerpos unidos mediante pares de restricción, que es a lo que llamaremos *mecanismo*, posee G grados de libertad de acuerdo con la siguiente ecuación:

$$G = 3 \cdot (N - 1) - 2 \cdot P$$

Siendo

$$G \rightarrow el\ número\ de\ grados\ de\ libertad\ del\ mecanismo$$
$$N \rightarrow el\ número\ de\ cuerpos\ totales\ del\ mecanismo\ (incluyendo\ el\ cuerpo\ fijo)$$
$$P \rightarrow el\ número\ de\ pares\ de\ restricción\ del\ mecanismo$$

$$P = Npr + Npp + Nprsd$$

Siendo $Npr, Npp\ y\ Nprsd$ el número de pares de revolución, prismáticos y de rodadura sin deslizamiento respectivamente.

Cabe decir que para implementar el método descrito en este trabajo es necesario introducir como vector de velocidades iniciales y aceleraciones iniciales uno tal que tenga al menos tantas componentes conocidas como grados de libertad tenga el lazo del mecanismo estudiado. Esto resulta obvio ya que estas velocidades y aceleraciones iniciales serán claves a la hora de resolver el problema de forma determinada.

3.3 Generalización del método. Velocidad

Partiendo de un problema sencillo como es un mecanismo 4 barras se plantean las ecuaciones que lo gobiernan.

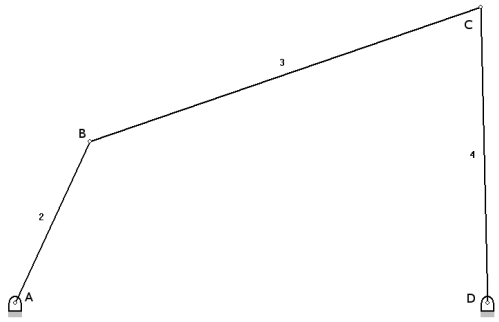


Figura 14

$$\vec{V}_{21}^{B} = \vec{V}_{11}^{A} + \vec{\omega}_{21} \wedge \overrightarrow{T_1}$$
$$\vec{V}_{31}^{C} = \vec{V}_{21}^{B} + \vec{\omega}_{31} \wedge \overrightarrow{T_2}$$
$$\vec{V}_{41}^{D} = \vec{V}_{31}^{C} + \vec{\omega}_{41} \wedge \overrightarrow{T_3} = 0$$

$$\vec{\omega}_{21} \wedge \overrightarrow{T_1} + \vec{\omega}_{31} \wedge \overrightarrow{T_2} + \vec{\omega}_{41} \wedge \overrightarrow{T_3} = 0$$

Los vectores $\vec{T}_i$ son los vectores denominados como *"Tramos"* que provienen de la geometría y con siempre conocidos.

Las conclusiones previas que podemos deducir son que, debemos resolver dos ecuaciones escalares, donde partimos de 3 incógnitas ($\vec{\omega}_{21}$, $\vec{\omega}_{31}\ y\ \vec{\omega}_{41}$) y para que el problema pueda tener solución para una posición dada debemos conocer una velocidad angular inicial. Siendo así tendríamos un problema cerrado con 2 ecuaciones escalares para dos incógnitas.

Pasamos ahora a analizar un mecanismo típico biela-manivela, el cual solo difiere del anterior en que el último par es un par prismático.

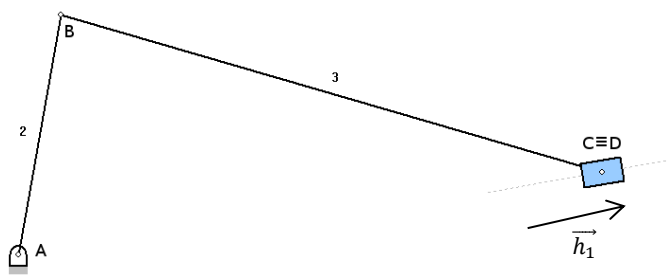


Figura 15

$$\vec{V}_{21}^{B} = \vec{V}_{11}^{A} + \vec{\omega}_{21} \wedge \overrightarrow{T_1}$$
$$\vec{V}_{31}^{C} = \vec{V}_{21}^{B} + \vec{\omega}_{31} \wedge \overrightarrow{T_2}$$
$$\vec{V}_{41}^{D} = \vec{V}_{31}^{C} + \vec{\omega}_{41} \wedge \overrightarrow{T_3}$$
$$(\vec{V}_{41}^{D})\overrightarrow{h_{p_1}} = 0\ ;\ \vec{\omega}_{41} = \vec{\omega}_{11} = 0$$

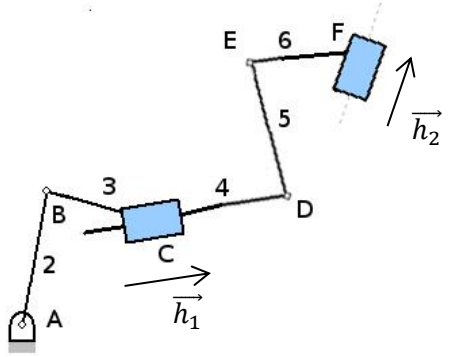
$$\left(\vec{\omega}_{21} \wedge \overrightarrow{T_1} + \vec{\omega}_{31} \wedge \overrightarrow{T_2} + \vec{\omega}_{41} \wedge \overrightarrow{T_3}\right)\overrightarrow{h_{p_1}} = 0$$

$$\vec{\omega}_{41} = \vec{\omega}_{11} = 0$$

Este problema ya varía del anterior puesto que hay que resolver una ecuación escalar solamente debido a que el par prismático iguala la velocidad angular del cuerpo 1 a la del cuerpo 4 (*Por restricciones iniciales el cuerpo 1 siempre será el bastidor. Ver Requisitos*). Para que se pueda resolver el sistema debemos conocer una velocidad inicial como dato.

Hasta este punto se pueden deducir dos hipótesis iniciales obvias: tendremos tantas incógnitas como números de cuerpos móviles tengamos (es decir sin contar con el bastidor) y deberemos conocer al menos tantas velocidades iniciales como grados de libertad tenga el mecanismo.

Complicando un poco el problema se plantea ahora un caso distinto. Simplemente se ha añadido dos cuerpos más al problema anterior unidos por un par prismático. En este caso sí que se mostrarán las ecuaciones de todos los pares.



$$\vec{V}_{21}^{B} = \vec{V}_{11}^{A} + \vec{\omega}_{21} \wedge \overrightarrow{T_1}$$

$$\vec{V}_{31}^{C} = \vec{V}_{21}^{B} + \vec{\omega}_{31} \wedge \overrightarrow{T_2}$$

$$\left(\vec{V}_{31}^{C} + \vec{\omega}_{31} \wedge \overrightarrow{CC}\right) \cdot \overrightarrow{h_{p_1}} = \left(\vec{V}_{41}^{D} - \vec{\omega}_{41} \wedge \overrightarrow{T_3}\right) \cdot \overrightarrow{h_{p_1}} \;\; ; \;\; \vec{\omega}_{31} = \vec{\omega}_{41}$$

$$\vec{V}_{51}^{E} = \vec{V}_{41}^{D} + \vec{\omega}_{51} \wedge \overrightarrow{T_4}$$

$$\vec{V}_{61}^{F} = \vec{V}_{51}^{E} + \vec{\omega}_{61} \wedge \overrightarrow{T_5}$$

$$\left(\vec{V}_{61}^{F} + \vec{\omega}_{61} \wedge \overrightarrow{FF}\right) \cdot \overrightarrow{h_{p_2}} = \left(\vec{V}_{11}^{A} - \vec{\omega}_{11} \wedge \overrightarrow{T_6}\right) \cdot \overrightarrow{h_{p_2}} \;\; ; \;\; \vec{\omega}_{61} = \vec{\omega}_{11} = 0$$

Figura 16

Podemos agrupar estas ecuaciones en 2 debido a que hay términos que se repiten en muchas de ellas. Obtenemos así dos ecuaciones escalares en este caso.

$$\left(\vec{\omega}_{21} \wedge \overrightarrow{T_1} + \vec{\omega}_{31} \wedge \overrightarrow{T_2}\right) \cdot \overrightarrow{h_{p_1}} = \left(\vec{V}_{41}^{D} - \vec{\omega}_{41} \wedge \overrightarrow{T_3}\right) \cdot \overrightarrow{h_{p_1}} \;\; ; \;\; \vec{\omega}_{31} = \vec{\omega}_{41}$$

$$\left(\vec{V}_{41}^{D} + \vec{\omega}_{51} \wedge \overrightarrow{T_4} + \vec{\omega}_{61} \wedge \overrightarrow{T_5}\right) \cdot \overrightarrow{h_{p_2}} = 0 \;\; ; \;\; \vec{\omega}_{61} = \vec{\omega}_{11} = 0$$

Reordenando los términos obtenemos lo siguiente.

$$\left(\vec{\omega}_{21} \wedge \overrightarrow{T_1} + \vec{\omega}_{31} \wedge \overrightarrow{T_2} + \vec{\omega}_{41} \wedge \overrightarrow{T_3}\right) \cdot \overrightarrow{h_{p_1}} = \left(\vec{V}_{41}^{D}\right) \cdot \overrightarrow{h_{p_1}} \;\; ; \;\; \vec{\omega}_{31} = \vec{\omega}_{41}$$

$$\left(\vec{\omega}_{51} \wedge \overrightarrow{T_4} + \vec{\omega}_{61} \wedge \overrightarrow{T_5}\right) \cdot \overrightarrow{h_{p_2}} = \left(-\vec{V}_{41}^{D}\right) \cdot \overrightarrow{h_{p_2}} \;\; ; \;\; \vec{\omega}_{61} = \vec{\omega}_{11} = 0$$

En este caso sí se pueden sacar muchas más hipótesis que podrían ayudar a la generalización. La primera y más importantes es que hay que resolver tantas ecuaciones escalares como pares prismáticos tenga el lazo (en este caso tenemos dos pares prismáticos pues deberemos resolver dos ecuaciones escalares). Otra hipótesis es que parece existir una relación entre los términos de la derecha de ambas ecuaciones y los términos de la izquierda. A la izquierda aparecen términos de velocidades angulares multiplicando a cada tramo y a la derecha aparecen términos de velocidades lineales.

Parece ser que cada ecuación esta multiplicada escalarmente por el vector perpendicular a la dirección de cada par prismático ($\overrightarrow{h_{p_1}}$ y $\overrightarrow{h_{p_2}}$).

Pasa exactamente lo mismo que en los casos anteriores, tenemos sistemas de ecuaciones escalares con tantas incógnitas angulares como cuerpos móviles tengamos en nuestro lazo. Pero en este caso vemos como además de las incógnitas angulares tenemos 2 incógnitas más de tipo velocidades lineales ($\vec{V}_{41}^{D}$). Es decir que este problema tiene 7 incógnitas ($\vec{\omega}_{21}$; $\vec{\omega}_{31}$; $\vec{\omega}_{41}$; $\vec{\omega}_{51}$; $\vec{\omega}_{61}$ y $\vec{V}_{41}^{D}$) y 4 ecuaciones lo cual hace coincidir los grados de libertad del mecanismo con las velocidades iniciales que debemos introducir para su resolución.

Con el fin de acercar al lector la comprensión del método a estudio y de generalizarlo para cualquier tipo de lazo de un mecanismo, se planteará un problema con una mayor dificultad que los resueltos en los apartados anteriores. El procedimiento que se seguirá será el mismo que hasta ahora, se plantearán todas y cada una de las ecuaciones que impone cada par de restricción y posteriormente se agruparan términos para determinar el conjunto de ecuaciones a resolver.

En este caso consideraremos un lazo con 12 cuerpos y pares de cualquier tipo estudiados (par de revolución, par prismático y par de rodadura sin deslizamiento).

En el caso del cálculo del problema de velocidades, que es el que ahora nos compete, las ecuaciones que rigen los pares de rodadura sin deslizamiento son las mismas que las que rigen los pares de rotación[1] . Por tanto, no se hará ninguna restricción entre estos dos tipos de pares. Esto es cierto solo para el problema de velocidad en un instante dado, cuando resolvamos el de aceleración tendremos en cuenta todas las diferencias y las definiciones de estos dos pares.

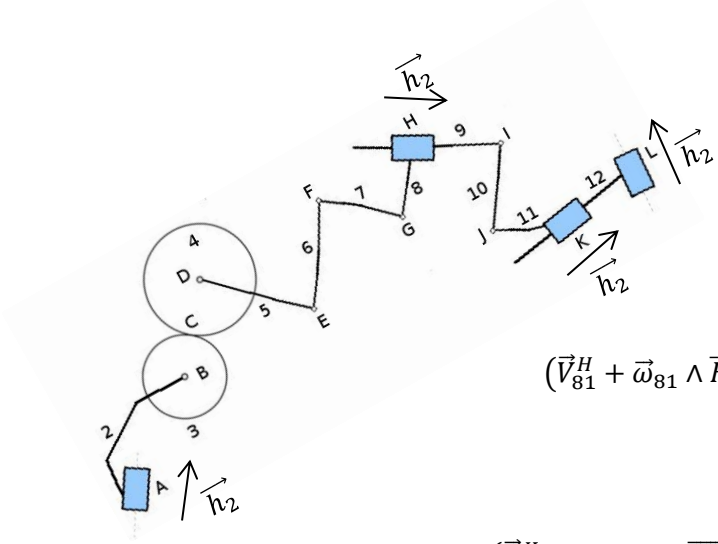


Figura 17

$$\left(\vec{V}_{11}^{A}+\vec{\omega}_{11}\wedge\overrightarrow{AA}\right)\cdot\overrightarrow{h_{p_1}}=\left(\vec{V}_{21}^{B}-\vec{\omega}_{21}\wedge\overrightarrow{T_1}\right)\cdot\overrightarrow{h_{p_1}}\ ;\ \vec{\omega}_{21}=\vec{\omega}_{11}=0$$

$$\vec{V}_{31}^{C}=\vec{V}_{21}^{B}+\vec{\omega}_{31}\wedge\overrightarrow{T_2}$$

$$\vec{V}_{41}^{D}=\vec{V}_{31}^{C}+\vec{\omega}_{41}\wedge\overrightarrow{T_3}$$

$$\vec{V}_{51}^{E}=\vec{V}_{41}^{D}+\vec{\omega}_{51}\wedge\overrightarrow{T_4}$$

$$\vec{V}_{61}^{F}=\vec{V}_{51}^{E}+\vec{\omega}_{61}\wedge\overrightarrow{T_5}$$

$$\vec{V}_{71}^{G}=\vec{V}_{61}^{F}+\vec{\omega}_{71}\wedge\overrightarrow{T_6}$$

$$\vec{V}_{81}^{H}=\vec{V}_{71}^{G}+\vec{\omega}_{81}\wedge\overrightarrow{T_7}$$

$$\left(\vec{V}_{81}^{H}+\vec{\omega}_{81}\wedge\overrightarrow{HH}\right)\cdot\overrightarrow{h_{p_2}}=\left(\vec{V}_{91}^{I}-\vec{\omega}_{91}\wedge\overrightarrow{T_8}\right)\cdot\overrightarrow{h_{p_2}}\ ;\ \vec{\omega}_{81}=\vec{\omega}_{91}$$

$$\vec{V}_{10,1}^{J}=\vec{V}_{91}^{I}+\vec{\omega}_{10,1}\wedge\overrightarrow{T_9}$$

$$\vec{V}_{11,1}^{K}=\vec{V}_{10,1}^{J}+\vec{\omega}_{11,1}\wedge\overrightarrow{T_{10}}$$

$$\left(\vec{V}_{11,1}^{K}+\vec{\omega}_{11,1}\wedge\overrightarrow{KK}\right)\cdot\overrightarrow{h_{p_3}}=\left(\vec{V}_{12,1}^{L}-\vec{\omega}_{12,1}\wedge\overrightarrow{T_{11}}\right)\cdot\overrightarrow{h_{p_3}}\ ;\ \vec{\omega}_{11,1}=\vec{\omega}_{12,1}$$

$$\left(\vec{V}_{12,1}^{L}+\vec{\omega}_{12,1}\wedge\overrightarrow{LL}\right)\cdot\overrightarrow{h_{p_4}}=\left(\vec{V}_{11}^{A}-\vec{\omega}_{11}\wedge\overrightarrow{T_{12}}\right)\cdot\overrightarrow{h_{p_4}}\ ;\ \vec{\omega}_{12,1}=\vec{\omega}_{11}=0$$

A pesar de lo complejo que pueda parecer, estas 12 ecuaciones se obtienen de forma sistemática aplicando el método de las velocidades relativas a cada par. Podemos agrupar, como ya hicimos en apartados anteriores, todos los términos repetidos y obtener así las ecuaciones principales para resolver este lazo. A la izquierda pondremos los términos angulares y a la derecha las velocidades lineales.

$$\left(\vec{\omega}_{21}\wedge\overrightarrow{T_1}+\vec{\omega}_{31}\wedge\overrightarrow{T_2}+\vec{\omega}_{41}\wedge\overrightarrow{T_3}++\vec{\omega}_{51}\wedge\overrightarrow{T_4}++\vec{\omega}_{61}\wedge\overrightarrow{T_5}+\vec{\omega}_{71}\wedge\overrightarrow{T_6}++\vec{\omega}_{81}\wedge\overrightarrow{T_7}\right)\cdot\overrightarrow{h_{p_1}}=\left(\vec{V}_{81}^{H}\right)\cdot\overrightarrow{h_{p_1}}$$

$$\vec{\omega}_{21}=\vec{\omega}_{11}=0$$

$$\left(\vec{\omega}_{91}\wedge\overrightarrow{T_8}+\vec{\omega}_{10,1}\wedge\overrightarrow{T_9}+\vec{\omega}_{11,1}\wedge\overrightarrow{T_{10}}\right)\cdot\overrightarrow{h_{p_2}}=\left(\vec{V}_{11,1}^{K}-\vec{V}_{81}^{H}\right)\cdot\overrightarrow{h_{p_2}}\ \ ;\ \ \vec{\omega}_{81}=\vec{\omega}_{91}$$

$$\left(\vec{\omega}_{12,1}\wedge\overrightarrow{T_{11}}\right)\cdot\overrightarrow{h_{p_3}}=\left(\vec{V}_{12,1}^{L}-\vec{V}_{11,1}^{K}\right)\cdot\overrightarrow{h_{p_3}}\ \ ;\ \ \vec{\omega}_{11,1}=\vec{\omega}_{12,1}$$

$$\left(\vec{V}_{12,1}^{L}\right)\cdot\overrightarrow{h_{p_4}}=0\ \ ;\ \ \vec{\omega}_{12,1}=\vec{\omega}_{11}=0$$

Volvemos a observar el mismo patrón que en los apartados anteriores. A la derecha un conjunto de multiplicaciones vectoriales de las velocidades angulares por los tramos y a la derecha la diferencia de dos

[1] En el caso de aceleraciones esto no se producirá

velocidades lineales. Todo ello añadiendo que todas las ecuaciones están multiplicadas por los vectores perpendiculares a las direcciones de los pares prismáticos de los cuatro pares.

A las ecuaciones de igualdad de velocidades angulares (tipo $\vec{\omega}_{i1} = \vec{\omega}_{j1} \quad \forall\, i,j$) las llamaremos *ecuaciones de condición prismática* y a las restantes las llamaremos *ecuaciones de grupo*. Estos nombres no tienen ninguna relevancia en el método, solo sirven como ataja para referirnos a cada tipo de ecuación ya que se pueden apreciar a simple vista dos tipos distintos.

Siendo así se puede afirmar previamente como hipótesis que existen tantas *ecuaciones de grupo* como pares prismáticos tenga el lazo. Lo mismo se podría afirmar para el número de *ecuaciones de condición prismática*. Así mismo se debe buscar una relación general y lógica para relacionar los términos de la derecha con los de la izquierda de cada ecuación así como establecer las pautas que existen para que se formen cada ecuación por separado dado un lazo concreto.

A continuación se exponen una serie de respuestas a las preguntas antes formadas. Estos resultados cabe decir que se han obtenido del estudio de varios mecanismos y diferentes combinaciones de los pares estudiados en un lazo. El procedimiento de generalización que a continuación se explica se ha logrado con una metodología similar a la desarrollada hasta ahora en este apartado. Se ha partido de lazos sencillos y posteriormente se han ido complicando hasta encontrar una pauta que pueda generalizarse para cualquier combinación de pares existente en un lazo dado. La generalización se ha hecho con vistas a una futura implementación en un programa de lectura de código tipo *Matlab* por ello se usan generalmente matrices y operaciones relativamente simples.

Pasamos ahora a exponer la generalización global del método para resolver el problema de velocidades en un lazo de un mecanismo. Primeramente resolveremos a modo de ejemplo visual el problema de la figura 17 del primer punto de este epígrafe. Después se indicará como resolver cualquier combinación de pares en un lazo.

Del problema ya planteado:

$$\left(\vec{\omega}_{21} \wedge \overrightarrow{T_1} + \vec{\omega}_{31} \wedge \overrightarrow{T_2} + \vec{\omega}_{41} \wedge \overrightarrow{T_3} + + \vec{\omega}_{51} \wedge \overrightarrow{T_4} + + \vec{\omega}_{61} \wedge \overrightarrow{T_5} + \vec{\omega}_{71} \wedge \overrightarrow{T_6} + + \vec{\omega}_{81} \wedge \overrightarrow{T_7}\right) \cdot \overrightarrow{h_{p_1}} = \left(\vec{V}_{81}^{H}\right) \cdot \overrightarrow{h_{p_1}}$$

$$\vec{\omega}_{21} = \vec{\omega}_{11} = 0$$

$$\left(\vec{\omega}_{91} \wedge \overrightarrow{T_8} + \vec{\omega}_{10,1} \wedge \overrightarrow{T_9} + \vec{\omega}_{11,1} \wedge \overrightarrow{T_{10}}\right) \cdot \overrightarrow{h_{p_2}} = \left(\vec{V}_{11,1}^{K} - \vec{V}_{81}^{H}\right) \cdot \overrightarrow{h_{p_2}} \quad ; \quad \vec{\omega}_{81} = \vec{\omega}_{91}$$

$$\left(\vec{\omega}_{12,1} \wedge \overrightarrow{T_{11}}\right) \cdot \overrightarrow{h_{p_3}} = \left(\vec{V}_{12,1}^{L} - \vec{V}_{11,1}^{K}\right) \cdot \overrightarrow{h_{p_3}} \quad ; \quad \vec{\omega}_{11,1} = \vec{\omega}_{12,1}$$

$$\left(\vec{V}_{12,1}^{L}\right) \cdot \overrightarrow{h_{p_4}} = 0 \quad ; \quad \vec{\omega}_{12,1} = \vec{\omega}_{11} = 0$$

Vimos que era necesario entender previamente la definición de *Grupo* para poder abordar el problema de forma genérica. A continuación se expone como desarrollar las ecuaciones a partir únicamente del conocimiento del *Lazo* ($\int$) y *Lazo de los pares* ($\int_p$).

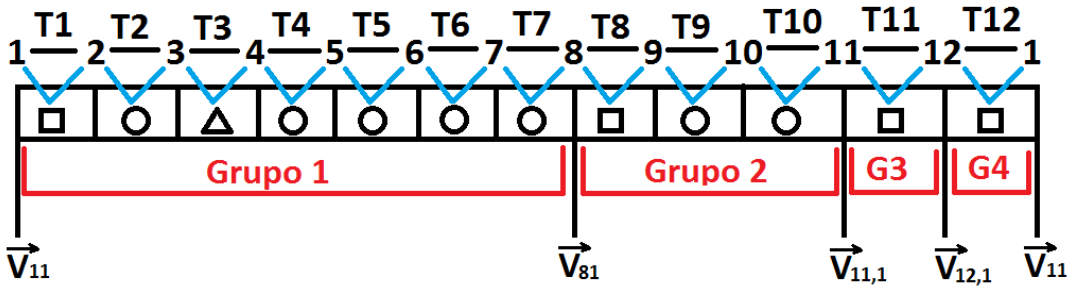


Figura 18

Tendremos tantas *ecuaciones de grupo* como grupos tengamos, y lo mismo para las *ecuaciones de condición prismática.* Cada tramo que forme parte de cada grupo impondrá un término de la parte izquierda de la igualdad en la *ecuación de grupo* a la que corresponda. Los términos de la derecha de estas ecuaciones serán iguales a las diferencias de velocidades entre las cuales se encuentra determinado el grupo. Esta diferencia se hará desde delante hacia atrás, es decir, al término de velocidad lineal que cierra el grupo se le restará el

término de velocidad lineal que lo inicia ("*Final menos principio*"). Por último, toda la ecuación de cada grupo, ira multiplicada escalarmente por el vector perpendicular a la dirección del par prismático ($\overrightarrow{h_{p_i}}$) que rige cada grupo (de ahí el hecho de imponer solo un par prismático por grupo).

Las sumas que aparecen en el lado izquierdo de las *ecuaciones de grupos* seguirán siempre la misma estructura. Siempre será la multiplicación escalar del vector correspondiente al tramo concreto de ese grupo por la velocidad angular del cuerpo que contiene a ese tramo. Se expone a modo de ejemplo:

Grupo 3: $\vec{T}_4 \ y \ \vec{T}_5 \longrightarrow \vec{\omega}_{51} \wedge \overrightarrow{T_4} + +\vec{\omega}_{61} \wedge \overrightarrow{T_5} \longrightarrow (\vec{\omega}_{51} \wedge \overrightarrow{T_4} + +\vec{\omega}_{61} \wedge \overrightarrow{T_5}) \cdot \overrightarrow{h_{p_3}} = \cdots$

$\vec{V}_{51} \ y \ \vec{V}_{61} \longrightarrow \vec{V}_{61} - \vec{V}_{51} \longrightarrow \ldots = (\vec{V}_{61} - \vec{V}_{51}) \cdot \overrightarrow{h_{p_3}}$

Para determinar el otro tipo de ecuaciones necesarias para resolver el problema, las *ecuaciones de condición prismática*, se determinan directamente de la visualización del esquema del *lazo de los pares* , es decir, si dos cuerpos están unidos por un par prismático es necesario que tengan iguales velocidades angulares ($\vec{\omega}_{i1} = \vec{\omega}_{j1}$).

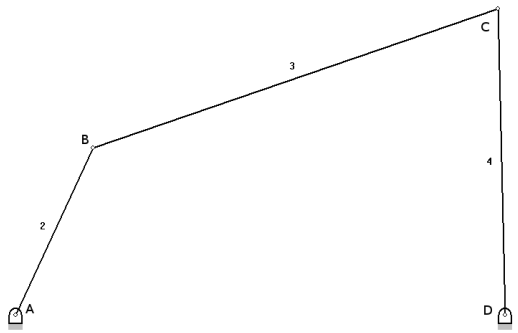
3.4 Generalización del método. Aceleración

A la hora de generalizar el método para resolver el problema de aceleración observaremos que la mecánica a seguir es siempre la misma con muy pocos puntos diferentes a incluir.

Como hemos hecho en el problema de velocidad, observaremos la resolución manual de varios ejemplos simples hasta que podamos llegar paso a paso a una generalización del método de aceleraciones. Los ejemplos serán los mismos expuestos en el apartado de velocidades.

Es necesario decir que previamente a resolver el problema de aceleración debe haberse resuelto el problema de velocidad y todas las velocidades (tanto angulares como lineales) deben ser conocidas.

El primer ejemplo muestra un caso sencillo de mecanismo plano de 4 barras (Figura 17). Las ecuaciones proporcionadas por el método de las aceleraciones relativas, y que servirán de base para la resolución del problema, se exponen a continuación.



$$\vec{a}_{21}^{B} = \vec{a}_{11}^{A} + \vec{\alpha}_{21} \wedge \overrightarrow{T_1} - \vec{\omega}_{21}^2 \overrightarrow{T_1}$$

$$\vec{a}_{31}^{C} = \vec{a}_{21}^{B} + \vec{\alpha}_{31} \wedge \overrightarrow{T_2} - \vec{\omega}_{31}^2 \overrightarrow{T_2}$$

$$\vec{a}_{41}^{D} = \vec{a}_{31}^{C} + \vec{\alpha}_{41} \wedge \overrightarrow{T_3} - \vec{\omega}_{41}^2 \overrightarrow{T_3} = 0$$

$$\vec{\alpha}_{21} \wedge \overrightarrow{T_1} + \vec{\alpha}_{31} \wedge \overrightarrow{T_2} + \vec{\alpha}_{41} \wedge \overrightarrow{T_3} - \vec{\omega}_{21}^2 \overrightarrow{T_1} - \vec{\omega}_{31}^2 \overrightarrow{T_2} - \vec{\omega}_{41}^2 \overrightarrow{T_3} = 0$$

Figura 19

Se observa que existe poca diferencia en cuanto a la estructura de las ecuaciones del problema de velocidad y del problema de aceleración en este caso sencillo. A la estructura típica del problema de velocidad se le restan unos términos nuevos relacionados con las velocidades angulares de los cuerpos El conjunto de ecuaciones se reduce a dos ecuaciones escalares y tiene 3 incógnitas ($\vec{\alpha}_{21}, \vec{\alpha}_{31}, \vec{\alpha}_{41}$), es decir, el problema tiene un solo grado de libertad al igual que lo tenía en el problema de velocidades (todas las velocidades ya son conocidas).

Analizamos ahora como resulta un problema con un par prismático como es un mecanismo biela-manivela. La única diferencia seria sustituir el último par por uno prismático.

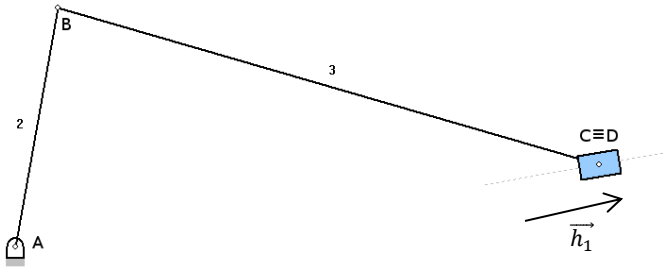


Figura 20

$$\vec{a}_{21}^{B} = \vec{a}_{11}^{A} + \vec{\alpha}_{21} \wedge \overrightarrow{T_1} - \vec{\omega}_{21}^2 \overrightarrow{T_1}$$
$$\vec{a}_{31}^{C} = \vec{a}_{21}^{B} + \vec{\alpha}_{31} \wedge \overrightarrow{T_2} - \vec{\omega}_{31}^2 \overrightarrow{T_2}$$
$$\vec{a}_{41}^{D} = \vec{a}_{31}^{C} + \vec{\alpha}_{41} \wedge \overrightarrow{T_3} - \vec{\omega}_{41}^2 \overrightarrow{T_3}$$
$$(\vec{a}_{41}^{D}) \cdot \overrightarrow{h_p}_1 = 0$$

$$\left(\vec{\alpha}_{21} \wedge \overrightarrow{T_1} + \vec{\alpha}_{31} \wedge \overrightarrow{T_2} + \vec{\alpha}_{41} \wedge \overrightarrow{T_3} - \vec{\omega}_{21}^2 \overrightarrow{T_1} - \vec{\omega}_{31}^2 \overrightarrow{T_2} - \vec{\omega}_{41}^2 \overrightarrow{T_3}\right) \cdot \overrightarrow{h_p}_1 = 0$$
$$\vec{\alpha}_{41} = \vec{\alpha}_{11} = 0$$

El conjunto ecuaciones se parece al calculado en el cuatro barras pero toda la ecuación esta multiplicada escalarmente por el vector perpendicular al par prismático ($\overrightarrow{h_p}_1$). Esta transformación también ocurría en el problema de velocidad. La otra ecuación (*ecuación de condición prismática*) es análoga a la de velocidad pero en este caso con aceleraciones angulares ($\vec{\alpha}_{41} = \vec{\alpha}_{11} = 0$).

En cómputo tendremos 2 ecuaciones escalares y 3 incógnitas, por tanto el mecanismo tendrá un solo grado de libertad, lo mismo que su problema análogo de velocidad.

Hasta este punto se siguen cumpliendo las dos hipótesis que se hicieron en el problema de velocidad. Se tienen tantas incógnitas en aceleraciones como número de cuerpos móviles (sin contar el eslabón 1) y se necesitan tantas aceleraciones iniciales como grados de libertad tiene el mecanismo para resolver el problema.

Pasamos a estudiar el tercer ejemplo que se vio en velocidades, un lazo con 6 eslabones y dos pares prismáticos. Con el conjunto de ecuaciones observaremos que se seguirá cumpliendo la misma estructura que en el ejemplo anterior.

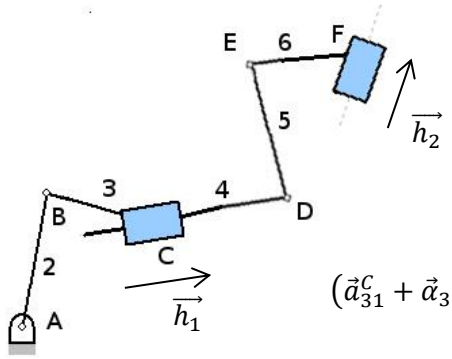


Figura 21

$$\vec{a}_{21}^{B} = \vec{a}_{11}^{A} + \vec{\alpha}_{21} \wedge \overrightarrow{T_1} - \vec{\omega}_{21}^2 \overrightarrow{T_1}$$
$$\vec{a}_{31}^{C} = \vec{a}_{21}^{B} + \vec{\alpha}_{31} \wedge \overrightarrow{T_2} - \vec{\omega}_{31}^2 \overrightarrow{T_2}$$
$$\left(\vec{a}_{31}^{C} + \vec{\alpha}_{31} \wedge \overrightarrow{CC} - \vec{\omega}_{31}^2 \overrightarrow{CC}\right) \cdot \overrightarrow{h_p}_1 = \left(\vec{a}_{41}^{D} - \vec{\alpha}_{41} \wedge \overrightarrow{T_3} + \vec{\omega}_{41}^2 \overrightarrow{T_3}\right) \cdot \overrightarrow{h_p}_1$$
$$\vec{a}_{51}^{E} = \vec{a}_{41}^{D} + \vec{\alpha}_{51} \wedge \overrightarrow{T_4} - \vec{\omega}_{51}^2 \overrightarrow{T_4}$$
$$\vec{a}_{61}^{F} = \vec{a}_{51}^{E} + \vec{\alpha}_{61} \wedge \overrightarrow{T_5} - \vec{\omega}_{61}^2 \overrightarrow{T_5}$$
$$\left(\vec{a}_{61}^{F} + \vec{\alpha}_{61} \wedge \overrightarrow{FF} - \vec{\omega}_{61}^2 \overrightarrow{FF}\right) \cdot \overrightarrow{h_p}_2 = \left(\vec{a}_{11}^{A} - \vec{\alpha}_{11} \wedge \overrightarrow{T_6} + \vec{\omega}_{11}^2 \overrightarrow{T_6}\right) \cdot \overrightarrow{h_p}_2$$
$$\vec{\alpha}_{31} = \vec{\alpha}_{41} \;\; ; \;\; \vec{\alpha}_{61} = \vec{\alpha}_{71} = \vec{\alpha}_{11} = 0$$

$$\left(\vec{\alpha}_{21} \wedge \overrightarrow{T_1} - \vec{\omega}_{21}^2 \overrightarrow{T_1} + \vec{\alpha}_{31} \wedge \overrightarrow{T_2} - \vec{\omega}_{31}^2 \overrightarrow{T_2}\right) \cdot \overrightarrow{h_p}_1 =$$

$$= \left(\vec{a}_{41}^D - \vec{\alpha}_{41} \wedge \overrightarrow{T_3} + \vec{\omega}_{41}^2 \overrightarrow{T_3}\right) \cdot \overrightarrow{h_p}_1$$

$$\left(\vec{a}_{41}^D + \vec{\alpha}_{51} \wedge \overrightarrow{T_4} - \vec{\omega}_{51}^2 \overrightarrow{T_4} + \vec{\alpha}_{61} \wedge \overrightarrow{T_5} - \vec{\omega}_{61}^2 \overrightarrow{T_5}\right) \cdot \overrightarrow{h_p}_2 = \left(-\vec{\alpha}_{11} \wedge \overrightarrow{T_6} + \vec{\omega}_{11}^2 \overrightarrow{T_6}\right) \cdot \overrightarrow{h_p}_2$$

$$\vec{\alpha}_{31} = \vec{\alpha}_{41} \;\; ; \;\; \vec{\alpha}_{61} = \vec{\alpha}_{71} = \vec{\alpha}_{11} = 0$$

Reordenando las ecuaciones podemos observar:

$$\left(\vec{\alpha}_{21} \wedge \overrightarrow{T_1} + \vec{\alpha}_{31} \wedge \overrightarrow{T_2} + \vec{\alpha}_{41} \wedge \overrightarrow{T_3} - \vec{\omega}_{21}^2 \overrightarrow{T_1} - \vec{\omega}_{31}^2 \overrightarrow{T_2} - \vec{\omega}_{41}^2 \overrightarrow{T_3}\right) \cdot \overrightarrow{h_p}_1 = \left(\vec{a}_{41}^D\right) \cdot \overrightarrow{h_p}_2$$

$$\left(\vec{\alpha}_{51} \wedge \overrightarrow{T_4} + \vec{\alpha}_{61} \wedge \overrightarrow{T_5} - \vec{\omega}_{51}^2 \overrightarrow{T_4} - \vec{\omega}_{61}^2 \overrightarrow{T_5}\right) \cdot \overrightarrow{h_p}_2 = \left(-\vec{a}_{41}^D\right) \cdot \overrightarrow{h_p}_2$$

$$\vec{\alpha}_{31} = \vec{\alpha}_{41} \;\; ; \;\; \vec{\alpha}_{61} = \vec{\alpha}_{71} = 0$$

Pasa lo mismo que pasaba en el método de velocidades. Se obtienen dos ecuaciones multiplicadas cada una por su correspondiente $\overrightarrow{h_p}_i$, en el lado izquierdo de la ecuación se tienen los términos de aceleraciones angulares multiplicados vectorialmente por los vectores de cada tramo, tal como pasaba en el problema de velocidad. Sin embargo, aparecen los nuevos términos de velocidades angulares elevadas al cuadrado y multiplicadas escalarmente por los vectores de los tramos. Estos nuevos términos restan a los anteriores. En la parte derecha de la igualdad se mantiene la estructura de aceleraciones lineales multiplicadas escalarmente por el vector perpendicular al respectivo par prismático.

Cuatro ecuaciones escalares, y 7 incógnitas hacen que sea necesarios conocer 3 aceleraciones iniciales coincidentes con el número de grados de libertad del mecanismo.

Por último analizamos el mecanismo de 12 eslabones con diversidad de pares (prismáticos, de rodadura sin deslizamiento y de revolución) que servirá para alcanzar una generalización mayor del método de aceleraciones.

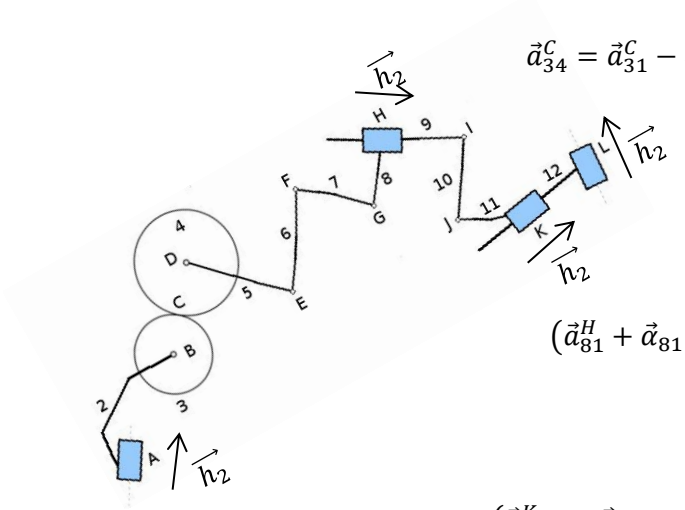


Figura 22

$$\left(\vec{a}_{11}^A + \vec{\alpha}_{11} \wedge \overrightarrow{AA} - \vec{\omega}_{11}^2 \overrightarrow{AA}\right) \cdot \overrightarrow{h_p}_1 = \left(\vec{a}_{21}^B - \vec{\alpha}_{21} \wedge \overrightarrow{T_1} + \vec{\omega}_{21}^2 \overrightarrow{T_1}\right) \cdot \overrightarrow{h_p}_1$$

$$\vec{a}_{31}^C = \vec{a}_{21}^B + \vec{\alpha}_{31} \wedge \overrightarrow{T_2} - \vec{\omega}_{31}^2 \overrightarrow{T_2}$$

$$\vec{a}_{34}^C = \vec{a}_{31}^C - \vec{a}_{41}^D + \vec{\alpha}_{31} \wedge \overrightarrow{CC} + \vec{\alpha}_{41} \wedge \overrightarrow{T_3} - \vec{\omega}_{31}^2 \overrightarrow{CC} - \vec{\omega}_{41}^2 \overrightarrow{T_3} - 2\vec{\omega}_{31} \wedge \vec{V}_{31}^C$$

$$\vec{a}_{51}^E = \vec{a}_{41}^D + \vec{\alpha}_{51} \wedge \overrightarrow{T_4} - \vec{\omega}_{51}^2 \overrightarrow{T_4}$$

$$\vec{a}_{61}^F = \vec{a}_{51}^E + \vec{\alpha}_{61} \wedge \overrightarrow{T_5} - \vec{\omega}_{61}^2 \overrightarrow{T_5}$$

$$\vec{a}_{71}^G = \vec{a}_{61}^F + \vec{\alpha}_{71} \wedge \overrightarrow{T_6} - \vec{\omega}_{71}^2 \overrightarrow{T_6}$$

$$\vec{a}_{81}^H = \vec{a}_{71}^G + \vec{\alpha}_{81} \wedge \overrightarrow{T_7} - \vec{\omega}_{81}^2 \overrightarrow{T_7}$$

$$\left(\vec{a}_{81}^H + \vec{\alpha}_{81} \wedge \overrightarrow{HH} - \vec{\omega}_{81}^2 \overrightarrow{HH}\right) \cdot \overrightarrow{h_p}_2 = \left(\vec{a}_{91}^I - \vec{\alpha}_{91} \wedge \overrightarrow{T_8} + \vec{\omega}_{91}^2 \overrightarrow{T_8}\right) \cdot \overrightarrow{h_p}_2$$

$$\vec{a}_{10,1}^J = \vec{a}_{91}^I + \vec{\alpha}_{10,1} \wedge \overrightarrow{T_9} - \vec{\omega}_{10,1}^2 \overrightarrow{T_9}$$

$$\vec{a}_{11,1}^K = \vec{a}_{10,1}^J + \vec{\alpha}_{11,1} \wedge \overrightarrow{T_{10}} - \vec{\omega}_{11,1}^2 \overrightarrow{T_{10}}$$

$$\left(\vec{a}_{11,1}^K + \vec{\alpha}_{11,1} \wedge \overrightarrow{KK} - \vec{\omega}_{11,1}^2 \overrightarrow{KK}\right) \cdot \overrightarrow{h_p}_3 = \left(\vec{a}_{12,1}^L - \vec{\alpha}_{12,1} \wedge \overrightarrow{T_{11}} + \vec{\omega}_{12,1}^2 \overrightarrow{T_{11}}\right) \cdot \overrightarrow{h_p}_3$$

$$\left(\vec{a}_{12,1}^L + \vec{\alpha}_{12,1} \wedge \overrightarrow{LL} - \vec{\omega}_{12,1}^2 \overrightarrow{LL}\right) \cdot \overrightarrow{h_p}_4 = \left(\vec{a}_{11}^A - \vec{\alpha}_{11} \wedge \overrightarrow{T_{12}} + \vec{\omega}_{11}^2 \overrightarrow{T_{12}}\right) \cdot \overrightarrow{h_p}_4$$

$$\vec{\alpha}_{11} = \vec{\alpha}_{21} = 0 \;\; ; \;\; \vec{\alpha}_{81} = \vec{\alpha}_{91} \;\; ; \;\; \vec{\alpha}_{11,1} = \vec{\alpha}_{12,1} \;\; ; \;\; \vec{\alpha}_{12,1} = \vec{\alpha}_{11} = 0$$

Si agrupamos las ecuaciones obtenemos

$$\left(\vec{\alpha}_{21} \wedge \overrightarrow{T_1} - \vec{\omega}_{21}^2 \overrightarrow{T_1} + \vec{\alpha}_{31} \wedge \overrightarrow{T_2} - \vec{\omega}_{31}^2 \overrightarrow{T_2} + \vec{\alpha}_{41} \wedge \overrightarrow{T_3} - \vec{\omega}_{41}^2 \overrightarrow{T_3} - \vec{a}_{34}^C - 2\vec{\omega}_{31} \wedge \vec{V}_{31}^C + \vec{\alpha}_{51} \wedge \overrightarrow{T_4} - \vec{\omega}_{51}^2 \overrightarrow{T_4} + \vec{\alpha}_{61} \wedge \overrightarrow{T_5} - \vec{\omega}_{61}^2 \overrightarrow{T_5} + \vec{\alpha}_{71} \wedge \overrightarrow{T_6} - \vec{\omega}_{71}^2 \overrightarrow{T_6} + \vec{\alpha}_{81} \wedge \overrightarrow{T_7} - \vec{\omega}_{81}^2 \overrightarrow{T_7}\right) \cdot \overrightarrow{h_{p_1}} = (\vec{a}_{81}^H) \cdot \overrightarrow{h_{p_1}}$$

$$\left(\vec{\alpha}_{91} \wedge \overrightarrow{T_8} - \vec{\omega}_{91}^2 \overrightarrow{T_8} + \vec{\alpha}_{10,1} \wedge \overrightarrow{T_9} - \vec{\omega}_{10,1}^2 \overrightarrow{T_9} + \vec{\alpha}_{11,1} \wedge \overrightarrow{T_{10}} - \vec{\omega}_{11,1}^2 \overrightarrow{T_{10}}\right) \cdot \overrightarrow{h_{p_2}} = \left(\vec{a}_{11,1}^K - \vec{a}_{81}^H\right) \cdot \overrightarrow{h_{p_2}}$$

$$\left(\vec{\alpha}_{12,1} \wedge \overrightarrow{T_{11}} - \vec{\omega}_{12,1}^2 \overrightarrow{T_{11}}\right) \cdot \overrightarrow{h_{p_3}} = \left(\vec{a}_{12,1}^L - \vec{a}_{11,1}^K\right) \cdot \overrightarrow{h_{p_3}}$$

$$\left(\vec{a}_{12,1}^L\right) \cdot \overrightarrow{h_{p_4}} = 0$$

$$\vec{\alpha}_{21} = 0 \;\; ; \;\; \vec{\alpha}_{81} = \vec{\alpha}_{91} \;\; ; \;\; \vec{\alpha}_{11,1} = \vec{\alpha}_{12,1} = 0$$

Reordenando los términos

$$\begin{pmatrix} \vec{\alpha}_{21} \wedge \overrightarrow{T_1} + \vec{\alpha}_{31} \wedge \overrightarrow{T_2} + \vec{\alpha}_{41} \wedge \overrightarrow{T_3} + \vec{\alpha}_{51} \wedge \overrightarrow{T_4} + \vec{\alpha}_{61} \wedge \overrightarrow{T_5} + \vec{\alpha}_{71} \wedge \overrightarrow{T_6} + \vec{\alpha}_{81} \wedge \overrightarrow{T_7} \\ + \\ -\vec{\omega}_{21}^2 \overrightarrow{T_1} - \vec{\omega}_{31}^2 \overrightarrow{T_2} - \vec{\omega}_{41}^2 \overrightarrow{T_3} - \vec{\omega}_{51}^2 \overrightarrow{T_4} - \vec{\omega}_{61}^2 \overrightarrow{T_5} - \vec{\omega}_{71}^2 \overrightarrow{T_6} - \vec{\omega}_{81}^2 \overrightarrow{T_7} \\ + \\ -\vec{a}_{34}^C - 2\vec{\omega}_{31} \wedge \vec{V}_{31}^C \end{pmatrix} \cdot \overrightarrow{h_{p_1}} = (\vec{a}_{81}^H) \cdot \overrightarrow{h_{p_1}}$$

$$\begin{pmatrix} \vec{\alpha}_{91} \wedge \overrightarrow{T_8} + \vec{\alpha}_{10,1} \wedge \overrightarrow{T_9} + \vec{\alpha}_{11,1} \wedge \overrightarrow{T_{10}} \\ + \\ -\vec{\omega}_{91}^2 \overrightarrow{T_8} - \vec{\omega}_{10,1}^2 \overrightarrow{T_9} - \vec{\omega}_{11,1}^2 \overrightarrow{T_{10}} \end{pmatrix} \cdot \overrightarrow{h_{p_2}} = \left(\vec{a}_{11,1}^K - \vec{a}_{81}^H\right) \cdot \overrightarrow{h_{p_2}}$$

$$\left(\vec{\alpha}_{12,1} \wedge \overrightarrow{T_{11}} - \vec{\omega}_{12,1}^2 \overrightarrow{T_{11}}\right) \cdot \overrightarrow{h_{p_3}} = \left(\vec{a}_{12,1}^L - \vec{a}_{11,1}^K\right) \cdot \overrightarrow{h_{p_3}}$$

$$0 = \left(\vec{a}_{12,1}^L\right) \cdot \overrightarrow{h_{p_4}}$$

$$\vec{\alpha}_{21} = 0 \;\; ; \;\; \vec{\alpha}_{81} = \vec{\alpha}_{91} \;\; ; \;\; \vec{\alpha}_{11,1} = \vec{\alpha}_{12,1} = 0$$

Tendremos en cuenta que el par de rodadura sin deslizamiento es de la tipología* 1, es decir, cóncavo – convexo. Por tanto la ecuación de las velocidades relativas en dicho par quedaría así:

$$\vec{a}_{34}^C = (\vec{\omega}_{31} - \vec{\omega}_{41})^2 \frac{R_3 R_4}{R_3 - R_4} \lambda$$

Se puede observar que parte de la estructura de las ecuaciones que rigen este problema en aceleraciones tienen cierto parecido a las ecuaciones del mismo problema en velocidad. Si bien se le incluyen unos términos adicionales y conocidos. La parte de las ecuaciones a la derecha de la igualdad sí mantiene su antigua estructura de diferencia de aceleraciones lineales multiplicadas por el vector perpendicular de par prismático de cada caso.

En cuanto a número de ecuaciones y grados de libertad se mantiene igual que en su homologo de velocidad. Tenemos 17 incógnitas y 8 ecuaciones por tanto el número de aceleraciones iniciales que debemos conocer para poder cerrar el problema es de 9. Este vuelve a coincidir con el número de grados de libertad del mecanismo.

Todos estos ejemplos realizados nos sirven para llegar a una posible generalización del método de aceleraciones que permita su posterior computarización. Al igual que pasaba en el problema de velocidad, el concepto de *Grupo* dentro del lazo del mecanismo también nos será muy útil. Recordando el concepto, un *Grupo* en un lazo de un mecanismo era aquel conjunto de eslabones en los que solo existía, entre ellos un par prismático. Usaremos este concepto de la misma forma en lo que lo usamos en el problema de velocidad, pero en este nuevo caso también nos ayudará a definir esos nuevos términos que diferencian la estructura de ambos problemas.

Como puede observarse, la definición de grupos para este último ejemplo propuesto es exactamente la misma que para el problema de velocidad. Esto se debe a que la definición de grupos dentro de un lazo no entiende de problemas de aceleración o de velocidad, simplemente es una agrupación que facilita los cálculos. Como se dijo en el apartado de velocidades, esta agrupación podrá ser o no única, pudiendo existir diferentes combinaciones de grupos dentro de un mismo lazo. Este hecho no afecta al cálculo ya que solo plantearía otras ecuaciones agrupadas de manera diferente pero similares.

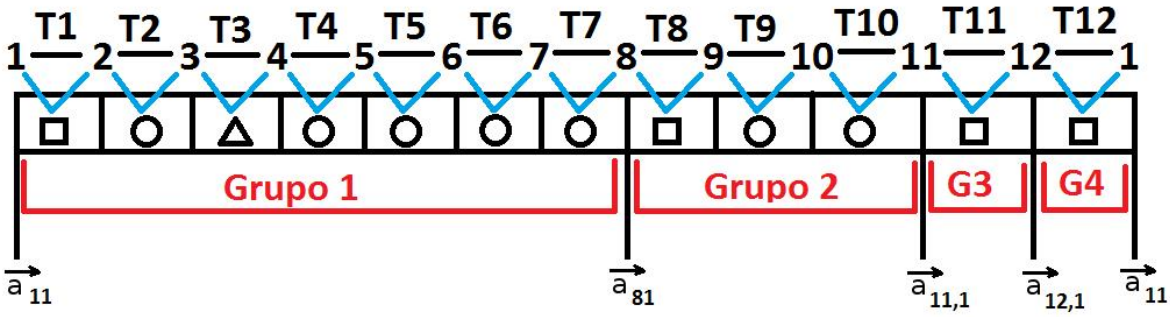


Figura 23

De la simple visión de la división en grupos del lazo de los pares del mecanismo se observa que tendremos 4 *ecuaciones de grupos* (siempre tendremos tantas *ecuaciones de grupos* como pares prismáticos existan).

La formación del conjunto de ecuaciones a partir, sólo, del lazo de los pares es muy similar a como se hacía en el problema de velocidad. Se expone a continuación un ejemplo:

Grupo 3: $\vec{T}_4$ y $\vec{T}_5$ $\longrightarrow$ $\vec{\alpha}_{51} \wedge \overrightarrow{T_4} + \vec{\alpha}_{61} \wedge \overrightarrow{T_5} - \vec{\omega}_{51}^2 \overrightarrow{T_4} - \vec{\omega}_{61}^2 \overrightarrow{T_5}$

$(\vec{\alpha}_{51} \wedge \overrightarrow{T_4} + \vec{\alpha}_{61} \wedge \overrightarrow{T_5} - \vec{\omega}_{51}^2 \overrightarrow{T_4} - \vec{\omega}_{61}^2 \overrightarrow{T_5}) \cdot \overrightarrow{h_{p_3}} = \cdots$

No existe ningún par de rodadura sin *deslizamiento dentro de este grupo*

$\vec{a}_{51}$ y $\vec{a}_{61}$

$\vec{a}_{61} - \vec{a}_{51} \longrightarrow \ldots = (\vec{a}_{61} - \vec{a}_{51}) \cdot \overrightarrow{h_{p_3}}$

En este ejemplo se demuestra que a estructura a seguir para construir las ecuaciones a partir de los grupos del lazo de los pares es muy similar al problema de aceleración. En lo único que varía es en añadir los términos tipo $\vec{\omega}_{i1}^2 \overrightarrow{T_j}$ restando en la parte derecha de la ecuación.

Si dentro del grupo a estudio, existiera un par de rodadura sin deslizamiento se deberá incluir dos términos más en la parte derecha de la ecuación. El proceso vuelve a ser fácilmente generalizable. Se hallan las ecuaciones del último ejemplo expuesto (12 eslabones y 12 pares, Figura 22) para poder observar este fenómeno.

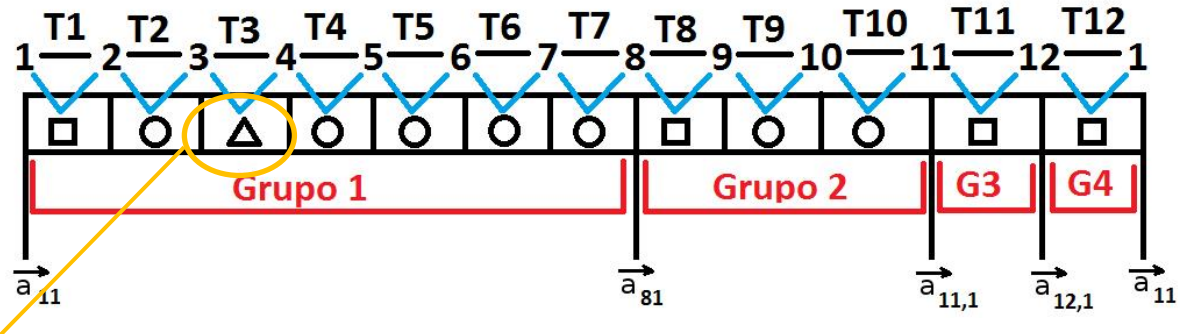


Figura 24

$$\left(\begin{array}{c}\vec{\alpha}_{21}\wedge\overrightarrow{T_1}+\vec{\alpha}_{31}\wedge\overrightarrow{T_2}+\vec{\alpha}_{41}\wedge\overrightarrow{T_3}+\vec{\alpha}_{51}\wedge\overrightarrow{T_4}+\vec{\alpha}_{61}\wedge\overrightarrow{T_5}+\vec{\alpha}_{71}\wedge\overrightarrow{T_6}+\vec{\alpha}_{81}\wedge\overrightarrow{T_7}\\ +\\ -\vec{\omega}_{21}^2\overrightarrow{T_1}-\vec{\omega}_{31}^2\overrightarrow{T_2}-\vec{\omega}_{41}^2\overrightarrow{T_3}-\vec{\omega}_{51}^2\overrightarrow{T_4}-\vec{\omega}_{61}^2\overrightarrow{T_5}-\vec{\omega}_{71}^2\overrightarrow{T_6}-\vec{\omega}_{81}^2\overrightarrow{T_7}\\ +\\ -\vec{a}_{34}^C-2\vec{\omega}_{31}\wedge\vec{V}_{31}^C\end{array}\right)\cdot\overrightarrow{h_{p_1}}=(\vec{a}_{81}^H)\cdot\overrightarrow{h_{p_1}}$$

En los grupos en los que se incluya un par de rodadura sin deslizamiento tan solo se incluirá en la parte derecha de la igualdad, dos términos restando referentes al lugar que ocupa dicho par ($-\vec{a}_{ij}^C-2\vec{\omega}_{i1}\wedge\vec{V}_{i1}^C$).

Por último, al igual que pasaba en el problema de velocidad, para determinar las *ecuaciones de condición prismática* para aceleraciones simplemente deberemos observar el posicionamiento de los pares prismáticos en el lazo de los pares e imponer la igualdad de aceleraciones angulares de los dos eslabones que una ($\vec{\alpha}_{i1}=\vec{\alpha}_{j1}$).

3.5 Restricciones finales y casos particualres

En este apartado se analizará unos casos particulares que difieren levemente del método planteado pero que sin embargo son muy comunes y merecen ser estudiados para poder implementarlos en el programa final. Analizaremos 2 casos particulares:

· Lazo sin ningún par prismático (Caso 1)

· Mecanismo con lazo abierto (Caso 2)

El método explicado es totalmente válido en estos dos ejemplos pero por sus peculiaridades es necesaria una mención ya que plantean casos que pueden provocar errores a la hora de su programación.

- **Caso 1:** Lazo sin ningún para prismático

  Como ya se ha explicado, la base del método es la división del lazo del mecanismo en grupos en los que sólo exista un par prismático. Esto es posible de realizar en cualquier lazo de un mecanismo, pero si este no posee ningún par prismático los procedimiento a seguir serán similares pero con sutiles diferencias.

  En principio definiremos un mecanismo sencillo como el de la figura 25, un cuatro barras. Este mecanismo solo posee un lazo y ningún par prismático. Las ecuaciones que lo rigen son:

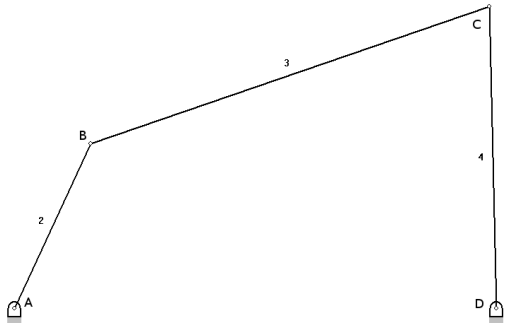


Figura 25

$$\vec{V}_{21}^{B} = \vec{V}_{11}^{A} + \vec{\omega}_{21} \wedge \overrightarrow{T_1}$$
$$\vec{V}_{31}^{C} = \vec{V}_{21}^{B} + \vec{\omega}_{31} \wedge \overrightarrow{T_2}$$
$$\vec{V}_{41}^{D} = \vec{V}_{31}^{C} + \vec{\omega}_{41} \wedge \overrightarrow{T_3} = 0$$

$$\vec{\omega}_{21} \wedge \overrightarrow{T_1} + \vec{\omega}_{31} \wedge \overrightarrow{T_2} + \vec{\omega}_{41} \wedge \overrightarrow{T_3} = 0$$

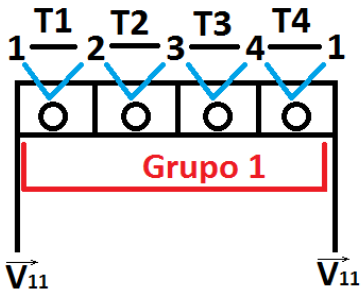


Figura 26

Aunque no exista ningún par prismático es necesario plantear un grupo en el mecanismo para poder continuar con el procedimiento del método. Es por eso que se define todo el lazo como un único grupo sin ser gobernado por ningún par prismático.

Las ecuaciones se plantearían de forma idéntica solo que los términos de la izquierda no irían multiplicados por ningún vector $\overrightarrow{h_{p_k}}$ siendo las ecuaciones a resolver del tipo:

$$\vec{\omega}_{21} \wedge \overrightarrow{T_1} + \vec{\omega}_{31} \wedge \overrightarrow{T_2} + \vec{\omega}_{41} \wedge \overrightarrow{T_3} + \cdots = 0$$

$$\begin{pmatrix} -T_{1y} \\ T_{1x} \end{pmatrix} \omega_{21} + \begin{pmatrix} -T_{2y} \\ T_{2x} \end{pmatrix} \omega_{31} + \begin{pmatrix} -T_{3y} \\ T_{3x} \end{pmatrix} \omega_{41} + \cdots = 0$$

$$\begin{pmatrix} -T_{1y} & -T_{2y} & -T_{3y} & \cdots & -T_{jy} \\ T_{1x} & T_{2x} & T_{3x} & \cdots & T_{jx} \end{pmatrix} \begin{pmatrix} \omega_{21} \\ \omega_{31} \\ \omega_{41} \\ \vdots \\ \omega_{i1} \end{pmatrix} = 0$$

A estas ecuaciones habrá que añadirle las restricciones de las velocidades iniciales impuestas al mecanismo.

El problema de aceleración presentaría la misma singularidad. Existiría un solo grupo sin ningún par prismático. Las ecuaciones que rigen la aceleración se presentan a continuación

$$\vec{a}_{21}^{B} = \vec{a}_{11}^{A} + \vec{\alpha}_{21} \wedge \overrightarrow{T_1} - \vec{\omega}_{21}^{2} \overrightarrow{T_1}$$
$$\vec{a}_{31}^{C} = \vec{a}_{21}^{B} + \vec{\alpha}_{31} \wedge \overrightarrow{T_2} - \vec{\omega}_{31}^{2} \overrightarrow{T_2}$$
$$\vec{a}_{41}^{D} = \vec{a}_{31}^{C} + \vec{\alpha}_{41} \wedge \overrightarrow{T_3} - \vec{\omega}_{41}^{2} \overrightarrow{T_3} = 0$$

$$\vec{\alpha}_{21} \wedge \overrightarrow{T_1} + \vec{\alpha}_{31} \wedge \overrightarrow{T_2} + \vec{\alpha}_{41} \wedge \overrightarrow{T_3} - \vec{\omega}_{21}^{2} \overrightarrow{T_1} - \vec{\omega}_{31}^{2} \overrightarrow{T_2} - \vec{\omega}_{41}^{2} \overrightarrow{T_3} = 0$$

En un caso genérico se podría observar:

$$\left(\vec{\alpha}_{21} \wedge \overrightarrow{T_1} + \vec{\alpha}_{31} \wedge \overrightarrow{T_2} + \vec{\alpha}_{41} \wedge \overrightarrow{T_3} + \cdots\right) - \left(\vec{\omega}_{21}^2 \overrightarrow{T_1} + \vec{\omega}_{31}^2 \overrightarrow{T_2} + \vec{\omega}_{41}^2 \overrightarrow{T_3} + \cdots\right) = 0$$

$$\begin{pmatrix} -T_{1y} \\ T_{1x} \end{pmatrix} \alpha_{21} + \begin{pmatrix} -T_{2y} \\ T_{2x} \end{pmatrix} \alpha_{31} + \begin{pmatrix} -T_{3y} \\ T_{3x} \end{pmatrix} \alpha_{41} + \cdots - \begin{pmatrix} T_{1_x}\omega_{21}^2 + T_{2_x}\omega_{31}^2 + T_{3_x}\omega_{41}^2 + \cdots \\ T_{1_y}\omega_{21}^2 + T_{2_y}\omega_{31}^2 + T_{3_y}\omega_{41}^2 + \cdots \end{pmatrix} = 0$$

$$\begin{pmatrix} -T_{1y} & -T_{2y} & -T_{3y} & \cdots & -T_{jy} \\ T_{1x} & T_{2x} & T_{3x} & \cdots & T_{jx} \end{pmatrix} \begin{pmatrix} \alpha_{21} \\ \alpha_{31} \\ \alpha_{41} \\ \vdots \\ \alpha_{i1} \end{pmatrix} - \begin{pmatrix} T_{1_x}\omega_{21}^2 + T_{2_x}\omega_{31}^2 + T_{3_x}\omega_{41}^2 + \cdots + T_{j_x}\omega_{i1}^2 \\ T_{1_y}\omega_{21}^2 + T_{2_y}\omega_{31}^2 + T_{3_y}\omega_{41}^2 + \cdots + T_{j_y}\omega_{i1}^2 \end{pmatrix} = 0$$

Como pasaba en velocidades, a estas ecuaciones sería necesario añadirles las condiciones de aceleraciones iniciales impuestas al lazo del mecanismo.

- **Caso 2:** Mecanismo con lazo abierto

Como ya se ha hablado hasta ahora, el método planteado necesita de un lazo del mecanismo para crear grupos a partir de los cuales plantear las ecuaciones que rigen la cinemática plana. Todos los lazos que se han visto hasta ahora son del mismo tipo lo que se denomina "*Lazos cerrados*". Estos tipos de lazos se llaman así porque inician en el eslabón 1 o bastidor y finalizan en el mismo eslabón 1. Las ecuaciones y el método empleado para deducir las ecuaciones para este tipo de lazos son válidas pero, ¿qué pasaría si el lazo fuera abierto?

Se denomina "*Lazo abierto*" aquel que aunque inicia en el eslabón 1 o bastidor no acaba en él, es decir finaliza en otro cuerpo que no está unido al eslabón 1. En la industria muchos ejemplos de mecanismos que poseen lazos abiertos los encontramos en los brazos robóticos.

A continuación se plantarán dos ejemplos muy similares, uno con un lazo cerrado y otro con un lazo abierto. Se intentará deducir cómo a partir del método explicado se pueden también resolver mecanismos con lazos abiertos imponiendo unas pequeñas restricciones.

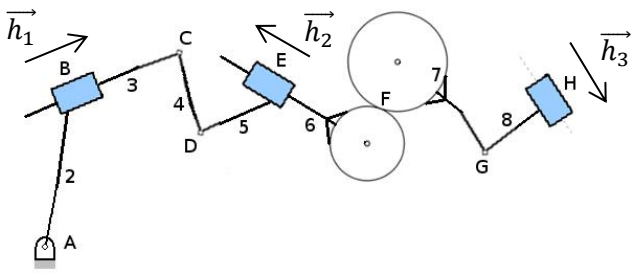


Figura 27

$$\vec{V}_{21}^{B} = \vec{V}_{11}^{A} + \vec{\omega}_{21} \wedge \overrightarrow{T_1}$$

$$\left(\vec{V}_{21}^{B} + \vec{\omega}_{21} \wedge \overrightarrow{BB}\right) \cdot \overrightarrow{h_{p_1}} = \left(\vec{V}_{31}^{C} - \vec{\omega}_{31} \wedge \overrightarrow{T_2}\right) \cdot \overrightarrow{h_{p_1}}$$

$$\vec{V}_{41}^{D} = \vec{V}_{31}^{C} + \vec{\omega}_{41} \wedge \overrightarrow{T_3}$$

$$\vec{V}_{51}^{E} = \vec{V}_{41}^{D} + \vec{\omega}_{51} \wedge \overrightarrow{T_4}$$

$$\left(\vec{V}_{51}^{E} + \vec{\omega}_{51} \wedge \overrightarrow{EE}\right) \cdot \overrightarrow{h_{p_2}} = \left(\vec{V}_{61}^{F} - \vec{\omega}_{61} \wedge \overrightarrow{T_5}\right) \cdot \overrightarrow{h_{p_2}}$$

$$\vec{V}_{71}^{G} = \vec{V}_{61}^{F} + \vec{\omega}_{71} \wedge \overrightarrow{T_6}$$

$$\vec{V}_{81}^{H} = \vec{V}_{71}^{G} + \vec{\omega}_{81} \wedge \overrightarrow{T_7}$$

$$\left(\vec{V}_{81}^{H} + \vec{\omega}_{81} \wedge \overrightarrow{HH}\right) \cdot \overrightarrow{h_{p_3}} = \left(\vec{V}_{11}^{A} - \vec{\omega}_{11} \wedge \overrightarrow{T_8}\right) \cdot \overrightarrow{h_{p_3}}$$

$$\vec{\omega}_{21} = \vec{\omega}_{31} \;\; ; \;\; \vec{\omega}_{51} = \vec{\omega}_{61} \;\; ; \;\; \vec{\omega}_{81} = \vec{\omega}_{11} = 0$$

Este lazo posee 8 cueros y 8 restricciones por lo que tiene 5 grados de libertad. Si agrupamos y ordenamos las ecuaciones:

$$\left(\vec{\omega}_{21} \wedge \overrightarrow{T_1} + \vec{\omega}_{31} \wedge \overrightarrow{T_2} + \vec{\omega}_{41} \wedge \overrightarrow{T_3} + \vec{\omega}_{51} \wedge \overrightarrow{T_4}\right) \cdot \overrightarrow{h_{p_1}} = \left(\vec{V}_{51}^{E}\right) \cdot \overrightarrow{h_{p_1}}$$

$$\left(\vec{\omega}_{61} \wedge \overrightarrow{T_5} + \vec{\omega}_{71} \wedge \overrightarrow{T_6} + \vec{\omega}_{81} \wedge \overrightarrow{T_7}\right) \cdot \overrightarrow{h_{p_2}} = \left(\vec{V}_{81}^{H} - \vec{V}_{51}^{E}\right) \cdot \overrightarrow{h_{p_2}}$$

$$\left(\vec{V}_{81}^{H}\right) \cdot \overrightarrow{h_{p_3}} = 0$$

$$\vec{\omega}_{21} = \vec{\omega}_{31} \;;\; \vec{\omega}_{51} = \vec{\omega}_{61} \;;\; \vec{\omega}_{81} = \vec{\omega}_{11} = 0$$

Se tienen 6 ecuaciones escalares con 11 incógnitas ($\vec{\omega}_{21}, \vec{\omega}_{31}, \vec{\omega}_{41}, \vec{\omega}_{51}, \vec{\omega}_{61}, \vec{\omega}_{71}, \vec{\omega}_{81}, \vec{V}_{51}^{E}, \vec{V}_{81}^{H}$). Se analizará ahora un mecanismo similar pero esta vez con el lazo abierto. Se trata del mismo mecanismo que ha sido abierto por su último eslabón.

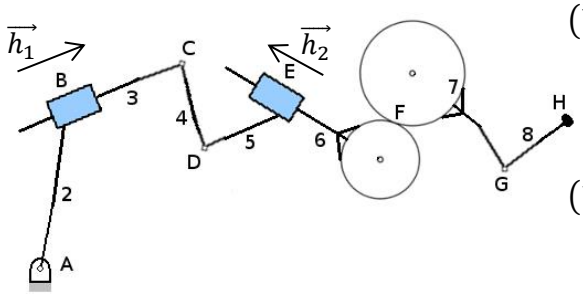


Figura 28

$$\vec{V}_{21}^{B} = \vec{V}_{11}^{A} + \vec{\omega}_{21} \wedge \overrightarrow{T_1}$$

$$\left(\vec{V}_{21}^{B} + \vec{\omega}_{21} \wedge \overrightarrow{BB}\right) \cdot \overrightarrow{h_{p_1}} = \left(\vec{V}_{31}^{C} - \vec{\omega}_{31} \wedge \overrightarrow{T_2}\right) \cdot \overrightarrow{h_{p_1}}$$

$$\vec{V}_{41}^{D} = \vec{V}_{31}^{C} + \vec{\omega}_{41} \wedge \overrightarrow{T_3}$$

$$\vec{V}_{51}^{E} = \vec{V}_{41}^{D} + \vec{\omega}_{51} \wedge \overrightarrow{T_4}$$

$$\left(\vec{V}_{51}^{E} + \vec{\omega}_{51} \wedge \overrightarrow{EE}\right) \cdot \overrightarrow{h_{p_2}} = \left(\vec{V}_{61}^{F} - \vec{\omega}_{61} \wedge \overrightarrow{T_5}\right) \cdot \overrightarrow{h_{p_2}}$$

$$\vec{V}_{71}^{G} = \vec{V}_{61}^{F} + \vec{\omega}_{71} \wedge \overrightarrow{T_6}$$

$$\vec{V}_{81}^{H} = \vec{V}_{71}^{G} + \vec{\omega}_{81} \wedge \overrightarrow{T_7}$$

$$\vec{\omega}_{21} = \vec{\omega}_{31} \;;\; \vec{\omega}_{51} = \vec{\omega}_{61}$$

En este caso poseemos una restricción menos por lo que el mecanismo pasa a tener 7 grados de libertad en total. Si volvemos a reordenar las ecuaciones obtendremos:

$$\left(\vec{\omega}_{21} \wedge \overrightarrow{T_1} + \vec{\omega}_{31} \wedge \overrightarrow{T_2} + \vec{\omega}_{41} \wedge \overrightarrow{T_3} + \vec{\omega}_{51} \wedge \overrightarrow{T_4}\right) \cdot \overrightarrow{h_{p_1}} = \left(\vec{V}_{51}^{E}\right) \cdot \overrightarrow{h_{p_1}}$$

$$\left(\vec{\omega}_{61} \wedge \overrightarrow{T_5} + \vec{\omega}_{71} \wedge \overrightarrow{T_6} + \vec{\omega}_{81} \wedge \overrightarrow{T_7}\right) \cdot \overrightarrow{h_{p_2}} = \left(\vec{V}_{81}^{H} - \vec{V}_{51}^{E}\right) \cdot \overrightarrow{h_{p_2}}$$

$$\vec{\omega}_{21} = \vec{\omega}_{31} \;;\; \vec{\omega}_{51} = \vec{\omega}_{61}$$

Este problema tiene 4 ecuaciones escalares con 11 incógnitas las mismas que en su homólogo en con lazo cerrado. Como puede apreciarse las ecuaciones que rigen ambos mecanismos son muy parecidas manteniendo unas diferencias que hacen que el método no pueda aplicarse tal cual.

Esto se ha salvado gracias a la aplicación de una serie de pasos que convierten el método descrito en una herramienta para resolver las ecuaciones de un mecanismo con un lazo abierto. Cabe decir a estas alturas que el programa implementa todo este método tiene en cuenta si el lazo es abierto o cerrado y es capaz de implantar, de forma automática, todos los cambios que a continuación se explicarán. Una vez detectemos un mecanismo con un lazo abierto deberemos seguir una serie de pasos:

1. Primero deberemos identificar el mecanismo como lazo abierto. Se distinguirá fácilmente ya que el lazo no acabará en el eslabón 1 además contará con un número de restricciones una unidad menor que el número de eslabones totales del lazo (Si se trata de un lazo cerrado el número de eslabones totales coincidirá con el número de pares de restricción totales).

2. Incluiremos de forma directa un par prismático como última restricción del lazo, que una el último cuerpo con el eslabón 1 o bastidor. De esta forma el lazo ahora estará cerrado por un par prismático pudiendo así aplicar el método estudiado para hallar sus ecuaciones cinemáticas.

3. Aplicaremos el método que se ha explicado en esta sección de forma completa y análoga a todos los casos explicados. Para ello deberemos establecer los grupos en el lazo de pares del mecanismo y con ellos formar las *ecuaciones de grupos* y *las ecuaciones de condiciones prismáticas.* Ya que hemos incluido un par prismático en la última posición, el último eslabón será un grupo en sí mismo dando una ecuación tipo $\left(\vec{V}_{i1}^{P}\right) \cdot \overrightarrow{h_{p}}_{k} = 0$.

4. También deberemos incluir, ya que el método así lo impone, la *ecuación de condición prismática* del último par que hemos impuesto. Esto arrojaría una ecuación más del tipo $\vec{\omega}_{i1} = \vec{\omega}_{11} = 0$.

5. El nuevo problema planteado no es el reflejo del problema real a resolver. El problema real tiene $(n+2)$ grados de libertad y $(n+2)$ ecuaciones y el problema inventado tiene solo n grados de libertad y n ecuaciones. Esto debe solucionarse para garantizar la resolución de las ecuaciones reales del problema. Una vez planteadas estas ecuaciones pasaremos a eliminar aquellos factores que no distinguen el mecanismo real del inventado para plantear las ecuaciones. Por ello eliminaremos la última *ecuación de grupo* que era del tipo $\left(\vec{V}_{i1}^{P}\right) \cdot \overrightarrow{h_{p}}_{k} = 0$ y así mismo también eliminaremos la última *ecuación de condición prismática* que es de tipo $\vec{\omega}_{i1} = \vec{\omega}_{11} = 0$. Esta eliminación de dos ecuaciones escalares del mecanismo aporta la condición de añadir dos grados de libertad extra en el mecanismo. Estos dos grados de libertad deben ser las componentes de $\vec{V}_{i1}^{P}$.

Este mecanismo para implementar el método a un lazo abierto es útil para la posterior programación del mismo. De esta forma simplemente se incluirán unas restricciones adicionales en el código y este podrá resolverlo. Si bien se establece como condición necesaria que se impongan como datos de velocidades y aceleraciones iniciales los todos términos de los vectores $\vec{V}_{i1}^{P}$ y $\vec{a}_{i1}^{P}$ refiriéndonos a las velocidades y aceleraciones del último punto del eslabón que está libre en lazo del mecanismo. Es decir que deberemos conocer de manera obligada estas velocidades y aceleraciones iniciales.

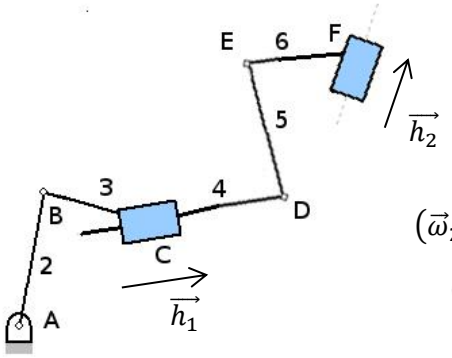
4 PROCEDIMIENTO DE RESOLUCIÓN MATRICIAL

Ena vez explicado el procedimiento por que cual obtener las ecuaciones del mecanismo a partir del lazo de pares ($\int_p$) y de la geometría, se explicará ahora cómo resolver este conjunto de ecuaciones de forma sencilla y sistemática. El procedimiento se explicará para resolver un problema de velocidad y posterior mente se explicará cómo resolver el de aceleración. Llegados a ese punto se expondrá extensamente el hecho de que la resolución del problema de aceleración es análogo a la manera de resolver el problema de velocidad, solo que se incluirán unos términos adecuados a las ecuaciones. Por ser así, se explica primero y en profundidad el problema de velocidad y después cómo generalizarlo para resolver el de aceleración.

4.1 Velocidad

Primeramente, y como ya se ha ido hablando en otros apartados, analicemos la estructura global de las ecuaciones que rigen el mecanismo. Partimos del conocimiento del lazo de los pares ($\int_p$), la disposición de todos los eslabones, así como toda la geometría necesaria que implique el mecanismo.

Con estos datos podremos obtener las ecuaciones que gobiernan el lazo a estudio del mecanismo. Tendremos tantas *ecuaciones de grupos* como pares prismáticos tenga el lazo del mecanismo y el mismo número de *ecuaciones de condición* prismática que lo mencionado anteriormente. Las *ecuaciones de grupos* tienen siempre la misma estructura. Se trata de una serie de elementos sumados que multiplican al vector perpendicular a la dirección del par prismático del grupo a la izquierda de la igualdad, y una diferencia de velocidades lineales que también está multiplicada por ese mismo vector a la derecha del igual. Los elementos anteriormente nombrados y que pertenecen a esas sumas de la izquierda del igual son siempre multiplicaciones vectoriales de términos de vectores de tramo y velocidades



$$\left(\vec{\omega}_{21} \wedge \overrightarrow{T_1} + \vec{\omega}_{31} \wedge \overrightarrow{T_2} + \vec{\omega}_{41} \wedge \overrightarrow{T_3}\right) \cdot \overrightarrow{h_{p_1}} = \left(\vec{V}_{41}^{D}\right) \cdot \overrightarrow{h_{p_1}} \;\; ; \;\; \vec{\omega}_{31} = \vec{\omega}_{41}$$

$$\left(\vec{\omega}_{51} \wedge \overrightarrow{T_4} + \vec{\omega}_{61} \wedge \overrightarrow{T_5}\right) \cdot \overrightarrow{h_{p_2}} = \left(\vec{V}_{41}^{D}\right) \cdot \overrightarrow{h_{p_2}} \;\; ; \;\; \vec{\omega}_{61} = \vec{\omega}_{11} = 0$$

Figura 29

Centrándonos en este grupo de ecuaciones (*ecuaciones de grupos*) podemos simplificar las multiplicaciones vectoriales y escalares que en ellas se realizan por productos entre matrices. Se va a escoger una multiplicación genérica para poder explicar lo expuesto anteriormente:

$$(\vec{\omega}_{i1} \wedge \overrightarrow{T_j}) \cdot \overrightarrow{h_{p_k}} \qquad Siendo \quad \overrightarrow{T_j} = \begin{pmatrix} T_{j_x} \\ T_{j_y} \\ 0 \end{pmatrix} \;;\; \overrightarrow{h_{p_k}} = \begin{pmatrix} h_{p_{kx}} \\ h_{p_{ky}} \\ 0 \end{pmatrix} \;;\; \vec{\omega}_{i1} = \begin{pmatrix} 0 \\ 0 \\ \omega_{i1} \end{pmatrix}$$

$$\begin{vmatrix} \vec{\imath} & \vec{\jmath} & \vec{k} \\ 0 & 0 & \omega_{i1} \\ T_{j_x} & T_{j_y} & 0 \end{vmatrix} \cdot \begin{pmatrix} h_{p_{kx}} \\ h_{p_{ky}} \end{pmatrix} = \begin{pmatrix} -T_{j_y} \\ T_{j_x} \end{pmatrix} \cdot \begin{pmatrix} h_{p_{kx}} \\ h_{p_{ky}} \end{pmatrix} \omega_{i1} = \left(-T_{jy} h_{p_{kx}} + T_{jx} h_{p_{ky}}\right) \omega_{i1} = TH_{jk} \omega_{i1}$$

$$TH_{jk} = -T_{jy} h_{p_{kx}} + T_{jx} h_{p_{ky}}$$

Como se puede observar la multiplicaciones vectoriales y escalares se transforman en algo sencillo, un número (TH_{jk}), multiplicado por ω_{i1} .Se puede, entonces, conociendo la definición de TH_{jk} y el tramo del que se esté hablado conocer el valor de este término. Se hace indispensable conocer qué dos términos se están multiplicando (j y k) y que velocidad angular es la relacionada (i). Para visualizarlo mejor se transformarán las ecuaciones de problema anterior planteado con las ecuaciones deducidas en este apartado.

$$\left(\vec{\omega}_{21} \wedge \overrightarrow{T_1} + \vec{\omega}_{31} \wedge \overrightarrow{T_2} + \vec{\omega}_{41} \wedge \overrightarrow{T_3}\right) \cdot \overrightarrow{h_{p_1}} = \left(\vec{V}_{41}^{D}\right) \cdot \overrightarrow{h_{p_1}} \;;\; \vec{\omega}_{31} = \vec{\omega}_{41}$$

$$\left(\vec{\omega}_{51} \wedge \overrightarrow{T_4} + \vec{\omega}_{61} \wedge \overrightarrow{T_5}\right) \cdot \overrightarrow{h_{p_2}} = \left(\vec{V}_{41}^{D}\right) \cdot \overrightarrow{h_{p_2}} \;;\; \vec{\omega}_{61} = \vec{\omega}_{11} = 0$$

$$TH_{11}\omega_{21} + TH_{21}\omega_{31} + TH_{31}\omega_{41} = \left(\vec{V}_{41}^{D}\right) \cdot \overrightarrow{h_{p_1}} \;;\; \vec{\omega}_{31} = \vec{\omega}_{41}$$

$$TH_{42}\omega_{51} + TH_{52}\omega_{61} = \left(\vec{V}_{41}^{D}\right) \cdot \overrightarrow{h_{p_2}} \;;\; \vec{\omega}_{61} = \vec{\omega}_{11} = 0$$

$$TH_{11} = -T_{1y} h_{p_{1x}} + T_{1x} h_{p_{1y}} \qquad TH_{21} = -T_{2y} h_{p_{1x}} + T_{2x} h_{p_{1y}} \qquad TH_{31} = -T_{3y} h_{p_{1x}} + T_{3x} h_{p_{1y}}$$

$$TH_{42} = -T_{4y} h_{p_{2x}} + T_{4x} h_{p_{2y}} \qquad TH_{52} = -T_{5y} h_{p_{2x}} + T_{5x} h_{p_{2y}}$$

Se recuerda en este punto que, conociendo toda la geometría los términos TH_{jk} son todos conocidos. No ocurre lo mismo con los términos de las velocidades angulares (ω_{i1}). Estos últimos, podrán ser o no conocidos, solo en algún caso pero en general son desconocidos puesto que representan las incógnitas a resolver en nuestro problema de velocidad junto con ciertas velocidades lineales.

Ya se ha visto que es posible transformar los términos de la derecha en sumatorios de multiplicaciones en las cuales se encuentran las incógnitas del problema. Antes de seguir con lo homólogo en la parte derecha de la igualdad se expondrá la transformación a forma matricial de la parte izquierda primero.

Si partimos de un caso general de ecuaciones para un lazo de un mecanismo:

$$\begin{cases} \left(\vec{\omega}_{21} \wedge \vec{T}_1 + \vec{\omega}_{31} \wedge \vec{T}_4 + \vec{\omega}_{41} \wedge \vec{T}_3 + \cdots\right) \cdot \overrightarrow{h_{p_1}} = \cdots \\ \left(\vec{\omega}_{61} \wedge \vec{T}_5 + \vec{\omega}_{71} \wedge \vec{T}_6 + \cdots\right) \cdot \overrightarrow{h_{p_2}} = \cdots \\ \left(\vec{\omega}_{91} \wedge \vec{T}_8 + \cdots\right) \cdot \overrightarrow{h_{p_3}} = \cdots \\ \vdots \\ \vdots \\ \vdots \end{cases} \rightarrow \begin{cases} \left(\sum_{i,\ j}^{n,\ m} (\vec{\omega}_{i1} \wedge \vec{T}_j)\right) \cdot \overrightarrow{h_{p_k}} = \cdots \\ \vdots \\ \vdots \\ \vdots \end{cases}$$

Siendo $i\,, j\; y\; k$ los índices que reflejan las correspondientes velocidades angulares, tramos y direcciones perpendiculares a los pares prismáticos respectivamente. Habrá tantas ecuaciones como pares prismáticos tenga el lazo.

Se obtendría la siguiente matriz TH.

$$\begin{cases} TH_{11}\omega_{21} + TH_{21}\omega_{31} + TH_{31}\omega_{41} + \cdots = \cdots \\ TH_{52}\omega_{61} + TH_{62}\omega_{71} + \cdots = \cdots \\ TH_{83}\omega_{91} + \cdots = \cdots \\ \cdot \\ \cdot \\ \cdot \end{cases} \rightarrow \begin{cases} \sum_{i,\ j}^{n,\ m} \left(TH_{jk}\omega_{i1}\right) = \cdots \\ \cdot \\ \cdot \\ \cdot \end{cases}$$

$$\begin{pmatrix} TH_{11} & \cdots & TH_{m1} \\ \vdots & \ddots & \vdots \\ TH_{1k} & \cdots & TH_{mk} \end{pmatrix} \begin{pmatrix} \omega_{21} \\ \omega_{31} \\ \cdot \\ \cdot \\ \cdot \\ \omega_{n1} \end{pmatrix} = TH\,W = \cdots$$

Para ilustrar lo explicado anteriormente se expone un ejemplo. Se transformará el conjunto de ecuaciones del lazo de la figura 1.

$$\begin{matrix} TH_{11}\omega_{21} + TH_{21}\omega_{31} + TH_{31}\omega_{41} = \cdots \\ TH_{42}\omega_{51} + TH_{52}\omega_{61} = \cdots \end{matrix} \qquad \begin{pmatrix} TH_{11} & TH_{21} & TH_{31} & 0 & 0 \\ 0 & 0 & 0 & TH_{42} & TH_{52} \end{pmatrix} W = \begin{pmatrix} \cdots \\ \cdots \end{pmatrix}$$

Pasamos ahora a transformar la parte de la derecha de la igualdad de las ecuaciones. El procedimiento es análogo. En este caso tendremos una resta de dos multiplicaciones de los términos de un vector de velocidad y un vector de dirección.

$$\begin{cases} \cdots = \left(\vec{V}_{41}^{D}\right) \cdot \overrightarrow{h_{p_1}} \\ \cdots = \left(\vec{V}_{61}^{F} - \vec{V}_{41}^{D}\right) \cdot \overrightarrow{h_{p_2}} \\ \cdots = \left(\vec{V}_{81}^{H} - \vec{V}_{61}^{F}\right) \cdot \overrightarrow{h_{p_3}} \\ \cdot \\ \cdot \\ \cdot \end{cases} \rightarrow \begin{cases} \left(\vec{V}_{v1}^{P2} - \vec{V}_{u1}^{P1}\right) \cdot \overrightarrow{h_{p_k}} \\ \cdot \\ \cdot \\ \cdot \end{cases}$$

$$\left(\vec{V}_{41}^{D}\right) \cdot \overrightarrow{h_{p_1}} = V_{41_x}^{D} h_{p1_x} + V_{41_y}^{D} h_{p1_y} \qquad \left(\vec{V}_{61}^{F} - \vec{V}_{41}^{D}\right) \cdot \overrightarrow{h_{p_2}} = -h_{p2_x} V_{41_x}^{D} - h_{p2_y} V_{41_y}^{D} + h_{p2_x} V_{61_x}^{F} + h_{p2_y} V_{61_y}^{F}$$

$$\left(\vec{V}_{81}^{H} - \vec{V}_{61}^{F}\right) \cdot \overrightarrow{h_{p_3}} = -h_{p3_x} V_{61_x}^{F} - h_{p3_y} V_{61_y}^{F} + h_{p3_x} V_{81_x}^{H} + h_{p3_y} V_{81_y}^{H}$$

$$\begin{cases} V_{41_x}^{D} h_{p1_x} + V_{41_y}^{D} h_{p1_y} \\ -h_{p2_x} V_{41_x}^{D} - h_{p2_y} V_{41_y}^{D} + h_{p2_x} V_{61_x}^{F} + h_{p2_y} V_{61_y}^{F} \\ -h_{p3_x} V_{61_x}^{F} - h_{p3_y} V_{61_y}^{F} + h_{p3_x} V_{81_x}^{H} + h_{p3_y} V_{81_y}^{H} \\ \cdot \\ \cdot \\ \cdot \end{cases}$$

Si expresamos las ecuaciones anteriores en forma matricial obtendremos dos matrices. La primera se formularía con las direcciones perpendiculares de los pares prismáticos, a esta matriz la llamaremos H. La segunda matriz que diferenciamos se trata de un vector con las incógnitas de las velocidades lineales de los pares prismáticos del mecanismo, lo llamaremos V.

$$\begin{pmatrix} h_{p1_x} & h_{p1_y} & 0 & 0 & 0 & 0 & \cdots \\ -h_{p2_x} & -h_{p2_y} & h_{p2_x} & h_{p2_y} & 0 & 0 & \cdots \\ 0 & 0 & -h_{p3_x} & -h_{p3_y} & h_{p3_x} & h_{p3_y} & \\ & & & \vdots & & & \ddots \end{pmatrix} \begin{pmatrix} V_{41_x}^D \\ V_{41_y}^D \\ V_{61_x}^F \\ V_{61_y}^F \\ V_{81_x}^H \\ V_{81_y}^H \\ \vdots \end{pmatrix} = \begin{pmatrix} \overrightarrow{h_{p_1}}' & 0 & 0 & \cdots \\ -\overrightarrow{h_{p_2}}' & \overrightarrow{h_{p_2}}' & 0 & \cdots \\ 0 & -\overrightarrow{h_{p_3}}' & \overrightarrow{h_{p_3}}' & \\ & \vdots & & \ddots \end{pmatrix} \begin{pmatrix} \vec{V}_{41}^D \\ \vec{V}_{61}^F \\ \vec{V}_{81}^H \\ \vdots \end{pmatrix}$$

$$\begin{pmatrix} \overrightarrow{h_{p_1}}' & 0 & 0 & \cdots \\ -\overrightarrow{h_{p_2}}' & \overrightarrow{h_{p_2}}' & 0 & \cdots \\ 0 & -\overrightarrow{h_{p_3}}' & \overrightarrow{h_{p_3}}' & \\ & \vdots & & \ddots \end{pmatrix} = H \qquad \begin{pmatrix} \vec{V}_{41}^D \\ \vec{V}_{61}^F \\ \vec{V}_{81}^H \\ \vdots \end{pmatrix} = V$$

Una vez ya obtenidas las expresiones matriciales de las ecuaciones que rigen el comportamiento del lazo del mecanismo pasamos a realizar un análisis completo del problema. En la parte de la izquierda de la igualdad tendríamos la multiplicación de la matriz TH por el vector de incógnitas de las velocidades angulares, W. En el otro lado tenemos algo análogo. La matriz H, multiplica al vector V de incógnitas de velocidades lineales.

Hacemos ahora una breve descripción cualitativa de las dos matrices que aparecen a cada lado de la igualdad.

$Matriz\ TH$: Esta matriz se forma con el conocimiento de los tamos década grupo del mecanismo. Su valor es único para el instante de estudio y sólo depende de términos referentes a la geometría del mecanismo. Su tamaño sería un, $[m\ x\ n]$ determinando m como el número de pares prismáticos que tenga el lazo del mecanismos (o también el número de grupos que tenga el lazo, ya que coinciden numéricamente). El número de columnas, n, se puede determinar ya que se trata del número total de tramos que dispone el mecanismo (quitando el tramo que engloba al sólido uno si este tramo existiera). Por tanto una definición cualitativa de TH sería:

$$TH \rightarrow [m\ x\ n] = \begin{cases} m = Número\ de\ pares\ prismáticos \\ n = Número\ de\ tramos\ del\ mecanismo \\ (menos\ el\ tramo\ que\ englobe\ el\ sólido\ 1) \end{cases}$$

$$\rightarrow \begin{cases} si\ el\ lazo\ es\ cerrado \rightarrow n = Nbody - 1 \\ si\ el\ lazo\ es\ abierto \rightarrow\ n = Nbody \end{cases}$$

Siendo Nbody el número de cuerpos totales que compone el lazo

$Matriz\ H$: Se trata de una matriz construida únicamente por los vectores perpendiculares de los pares prismáticos del lazo. Sus dimensiones $[p\ x\ q]$, se pueden determinar fácilmente. El número de filas (p) es igual e número de pares prismáticos que tiene el mecanismo, es decir que coincide numéricamente con m. Por otro lado el número de columnas (q) es igual al número de eslabones totales.

$$H \rightarrow \ [p\ x\ q] = \begin{cases} p = m = Número\ de\ pares\ prismáticos \\ \\ q = \ Número\ de\ eslabones\ totales \end{cases}$$

Las incógnitas que tenemos en dicha ecuación son tanto el vector W de velocidades angulares como el vector V de velocidades lineales. Estos dos vectores se pueden agrupar en un único vector columna de incógnitas que llamaremos X_V para referenciarlo al problema de velocidad.

$$TH\ W = H\ V \qquad (TH\ \vdots\ H)\begin{pmatrix} W \\ \cdots \\ V \end{pmatrix} = 0$$

A su vez también definimos la matriz A la cual resulta de la combinación de las matrices TH y H. El problema de velocidad queda reducido por el momento a resolver el problema sencillo matricial:

$$A\ X_V = 0$$

Aun así debemos añadir que esta ecuación no refleja el conjunto total de ecuaciones ya que las llamadas $ecuaciones\ de\ condición\ prismática$ no han sido aún añadidas. También hay que señalar que queda particularizar la ecuación resultante a las condiciones de velocidades iniciales de nuestro lazo.

- **Ecuaciones de condición prismática**

Como ya se ha visto en anteriores apartados, estas ecuaciones surgen de los pares prismáticos del lazo de estudio. La condición que imponen es la de igualdad de velocidades angulares entre los dos eslabones que une el par prismático.

$$\vec{\omega}_i = \vec{\omega}_j$$

Estas ecuaciones junto con las *ecuaciones de grupo* rigen el movimiento del mecanismo particularizadas para el instante a estudio (velocidades y aceleraciones angulares). Por tanto deben de ser incluidas en el proceso de generalización y en la ecuación a resolver. Es decir, se hace necesario estudiar cómo estas ecuaciones modifican la matriz A definida en el apartado anterior.

Las columnas de la matriz A están relacionadas con las velocidades angulares y lineales del lazo del mecanismo y que para nosotros son unas incógnitas. Más concretamente las *ecuaciones de condición prismática* están relacionadas con las velocidades angulares por lo que, estas ecuaciones no afectaran a las columnas relacionadas con las velocidades lineales. Es decir, solo modificarán la matriz TH anteriormente descrita.

La forma en la que la modifica es fácil de ver, tan solo hay que imponer la condición de igualdad sobre las columnas de las velocidades angulares que si tengan impuesta una relación prismática. Por ejemplo:

$$\begin{pmatrix} TH_{11} & TH_{21} & TH_{31} & 0 & 0 \\ 0 & 0 & 0 & TH_{42} & TH_{52} \end{pmatrix}\begin{pmatrix} \omega_{21} \\ \omega_{31} \\ \omega_{41} \\ \omega_{51} \\ \omega_{61} \end{pmatrix}$$

$$\vec{\omega}_{31} = \vec{\omega}_{41}\ ;\ \vec{\omega}_{61} = \vec{\omega}_{11} = 0$$

$$TH_{11}\omega_{21} + TH_{21}\omega_{31} + TH_{31}\omega_{41} \rightarrow TH_{11}\omega_{21} + (TH_{21} + TH_{31})\omega_{31}$$
$$TH_{42}\omega_{51} + TH_{52}\omega_{61} \rightarrow TH_{42}\omega_{51}$$

$$\begin{pmatrix} TH_{11} & TH_{21}+TH_{31} & 0 & 0 & 0 \\ 0 & 0 & 0 & TH_{42} & 0 \end{pmatrix} \begin{pmatrix} \omega_{21} \\ \omega_{31} \\ \omega_{41} \\ \omega_{51} \\ \omega_{61} \end{pmatrix}$$

Como puede observarse solo se trata de sumar las columnas i y j que tengan impuesta una condición prismática del tipo $\vec{\omega}_i = \vec{\omega}_j$. Se puede realizar en el sentido que se quiera pero por comodidad y con vistas a la implementación en *Matlab* la suma se expresa en la columna i y la columna j se iguala a cero. También cabe decir que aquellos pares que unan un eslabón con el bastidor o *eslabón 1* sus velocidades angular serán igual a 0 por lo que en el caso anterior se iguala esa columna a cero. Llamaremos a la nueva matriz TH con el nombre TH^{cp} que recuerda a que ya se ha implantado la condición prismática. Por tanto la nueva matriz A se definiría:

$$TH^{cp}\, W = H\, V$$

$$(TH^{cp} \vdots H)\begin{pmatrix} W \\ \cdots \\ V \end{pmatrix} = 0$$

$$A_{cp}\, X_V = 0$$

La resolución de la ecuación matricial anterior no daría otra cosa que la solución trivial $X_V = (0\ 0\ 0\ \ldots\ 0)$ por tanto es necesario encontrar una matriz B que nos sirva como ayuda para poder resolver este problema y convertirlo en el tipo:

$$C\, X_V = B$$

Problema matricial ya con solución distinta de la trivial. Es decir tenemos que encontrar la relación entre las matrices A_{cp}, B y C. Es precisamente las condiciones de velocidad de entrada las que nos ayudan a resolver este problema.

- **Transformación del sistema compatible indeterminado a un sistema compatible determinado**

Partiremos del conocimiento del problema matricial anterior y del vector de velocidades de entrada.

$$(TH^{cp} \vdots H)\begin{pmatrix} W \\ \cdots \\ V \end{pmatrix} = 0 \longrightarrow \begin{pmatrix} TH_{11}^{cp} & TH_{12}^{cp} & \ldots & TH_{1m}^{cp} & H_{11} & H_{12} & \ldots & H_{1q} \\ TH_{21}^{cp} & \ddots & & \vdots & H_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ TH_{n1}^{cp} & \ldots & \ldots & TH_{nm}^{cp} & H_{p1} & \ldots & \ldots & H_{pq} \end{pmatrix} \begin{pmatrix} \omega_{21} \\ \omega_{31} \\ \vdots \\ V_{41_x}^{D} \\ V_{41_y}^{D} \\ V_{61_x}^{F} \\ \vdots \end{pmatrix} = 0$$

O lo que es lo mismo:

$$\begin{cases} TH_{11}\omega_{21} + (TH_{21} + TH_{31})\omega_{31} - V^D_{41_x}h_{p1_x} - V^D_{41_y}h_{p1_y} = 0 \\ TH_{42}\omega_{51} + TH_{52}\omega_{61} - h_{p2_x}V^D_{41_x} - h_{p2_y}V^D_{41_y} + h_{p2_x}V^F_{61_x} + h_{p2_y}V^F_{61_y} = 0 \\ \cdot \\ \cdot \\ \cdot \end{cases}$$

Del problema conocemos el vector de velocidades de entrada que lo representaremos por ϕ_V. Este vector tendrá exactamente la misma longitud que el vector X_V. Sus componentes serán numéricos siendo cero las componentes de estrada que no se conocen o que no forman parte de las velocidades de entrada conocidas.

$$\begin{cases} \omega_{31} = 3\ rad/s \\ \vec{V}^D_{41} = \begin{pmatrix} 4 \\ ? \end{pmatrix} m/s \\ \vec{V}^F_{61} = \begin{pmatrix} ? \\ 5 \end{pmatrix} m/s \end{cases} \longrightarrow \quad \phi_V = \begin{pmatrix} W \\ \cdots \\ V \end{pmatrix} = \begin{pmatrix} \omega_{21} \\ \omega_{31} \\ \vdots \\ V^D_{41_x} \\ V^D_{41_y} \\ V^F_{61_x} \\ \vdots \end{pmatrix} = \begin{pmatrix} 0 \\ 3 \\ \vdots \\ 4 \\ 0 \\ 0 \\ 5 \\ \vdots \end{pmatrix}$$

$$\begin{cases} TH_{11}\omega_{21} + (TH_{21} + TH_{31}) \cdot 3 - 4 \cdot h_{p1_x} - V^D_{41_y}h_{p1_y} = 0 \\ TH_{42}\omega_{51} + TH_{52}\omega_{61} - h_{p2_x} \cdot 4 - h_{p2_y}V^D_{41_y} + h_{p2_x}V^F_{61_x} + h_{p2_y} \cdot 5 = 0 \\ \cdot \\ \cdot \\ \cdot \end{cases}$$

$$\begin{cases} TH_{11}\omega_{21} - V^D_{41_y}h_{p1_y} = -(TH_{21} + TH_{31}) \cdot 3 + 4 \cdot h_{p1_x} \\ TH_{42}\omega_{51} + TH_{52}\omega_{61} - h_{p2_y}V^D_{41_y} + h_{p2_x}V^F_{61_x} = h_{p2_x} \cdot 4 - h_{p2_y} \cdot 5 \\ \cdot \\ \cdot \\ \cdot \end{cases}$$

La consecuencia directa de aplicar el vector de velocidades de entrada es la deducción de unos términos conocidos que pueden pasarse al lado derecho de la ecuación. Este hecho hace que la solución el problema sea determinada y distinta de la trivial.

$$B = -\begin{pmatrix} TH^{cp}_{11} & TH^{cp}_{12} & \dots & TH^{cp}_{1m} & H_{11} & H_{12} & \dots & H_{1q} \\ TH^{cp}_{21} & \ddots & & \vdots & H_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ TH^{cp}_{n1} & \dots & \dots & TH^{cp}_{nm} & H_{p1} & \dots & \dots & H_{pq} \end{pmatrix} \begin{pmatrix} 0 \\ 3 \\ \vdots \\ 4 \\ 0 \\ 0 \\ 5 \\ \vdots \end{pmatrix} = -(TH^{cp} \ \vdots \ H)\varphi_V = -A_{cp}\phi_V$$

Como ocurría en la implementación de las condiciones prismáticas, es necesario eliminar de la parte izquierda

de las ecuaciones todos los términos que hemos pasado al lado derecho. Las columnas que pertenecen a las velocidades conocidas (es decir distintas de cero) del vector de velocidades de entrada se deberán igualar a cero.

$$\begin{pmatrix} TH_{11}^{cp} & TH_{12}^{cp} & \dots & TH_{1m}^{cp} & H_{11} & H_{12} & \dots & H_{1q} \\ TH_{21}^{cp} & \ddots & & \vdots & H_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ TH_{n1}^{cp} & \dots & \dots & TH_{nm}^{cp} & H_{p1} & \dots & \dots & H_{pq} \end{pmatrix} = C$$

$$0 \qquad\qquad 0$$

Por tanto el problema de velocidad queda planteado y provisto de una solución única y determinada.

$$C\, X_V = B$$

El vector de soluciones X_V poseerá unos determinados datos numéricos para todas las incógnitas del problema pero, por haber impuesto unas velocidades de entrada, los términos relacionados con esas velocidades de entrada dentro de este vector X_V resultan ser cero. Por tanto cabe decir que será necesario imponer que esas velocidades (las cuales conocemos desde el principio) en el vector de solución X_V.

4.2 Aceleración

Resolviendo el problema anterior obtendremos las velocidades incógnitas del problema de velocidad. Es necesario, llegada esta altura, plantear como plantear y resolver las ecuaciones matriciales que rigen el problema de aceleraciones. A modo de ejemplo aclaratorio se expone el mismo problema que se presentó el inicio del este epígrafe pero, en este caso, resolveremos el problema de aceleraciones.

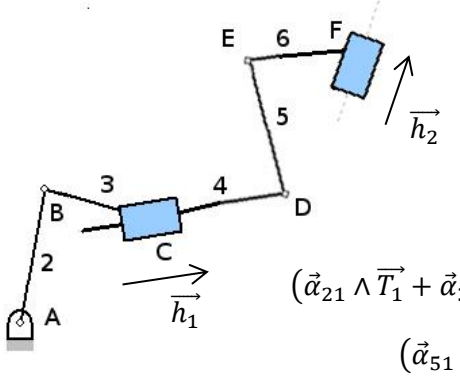


Figura 30

$$\left(\vec{\alpha}_{21} \wedge \overrightarrow{T_1} + \vec{\alpha}_{31} \wedge \overrightarrow{T_2} + \vec{\alpha}_{41} \wedge \overrightarrow{T_3} - \vec{\omega}_{21}^2 \overrightarrow{T_1} - \vec{\omega}_{31}^2 \overrightarrow{T_2} - \vec{\omega}_{41}^2 \overrightarrow{T_3}\right) \cdot \overrightarrow{h_p}_1 = (\vec{\alpha}_{41}^D) \cdot \overrightarrow{h_p}_1$$

$$\left(\vec{\alpha}_{51} \wedge \overrightarrow{T_4} + \vec{\alpha}_{61} \wedge \overrightarrow{T_5} - \vec{\omega}_{51}^2 \overrightarrow{T_4} - \vec{\omega}_{61}^2 \overrightarrow{T_5}\right) \cdot \overrightarrow{h_p}_2 = (-\vec{\alpha}_{41}^D) \cdot \overrightarrow{h_p}_2$$

$$\vec{\alpha}_{31} = \vec{\alpha}_{41} \;\; ; \;\; \vec{\alpha}_{61} = \vec{\alpha}_{71} = 0$$

Estas ecuaciones ya se han deducido en el apartado anterior cuando se explicó el método para formar las ecuaciones. Si bien las incógnitas siguen siendo las aceleraciones angulares ($\vec{\alpha}_{i1}$) y las aceleraciones lineales ($\vec{a}_{j1}^P$), se explica a continuación como transformar estas ecuaciones a forma matricial resoluble por *Matlab.* Por ejemplo:

$$\left(\vec{\alpha}_{i1} \wedge \vec{T}_j\right) \cdot \overrightarrow{h_p}_k \qquad Siendo:\ \overrightarrow{T_j} = \begin{pmatrix} T_{j_x} \\ T_{j_y} \\ 0 \end{pmatrix} \;\; ; \;\; \overrightarrow{h_p}_k = \begin{pmatrix} h_{p_{kx}} \\ h_{p_{ky}} \\ 0 \end{pmatrix} \;\; ; \;\; \vec{\alpha}_{i1} = \begin{pmatrix} 0 \\ 0 \\ \alpha_{i1} \end{pmatrix}$$

$$\begin{vmatrix} \vec{\imath} & \vec{\jmath} & \vec{k} \\ 0 & 0 & \alpha_{i1} \\ T_{j_x} & T_{j_y} & 0 \end{vmatrix} \cdot \begin{pmatrix} h_{p_{kx}} \\ h_{p_{ky}} \end{pmatrix} = \begin{pmatrix} -T_{j_y} \\ T_{j_x} \end{pmatrix} \cdot \begin{pmatrix} h_{p_{kx}} \\ h_{p_{ky}} \end{pmatrix} \alpha_{i1} = \left(-T_{jy} h_{p_{kx}} + T_{jx} h_{p_{ky}} \right) \alpha_{i1} = TH_{jk} \alpha_{i1}$$

$$TH_{jk} = -T_{jy} h_{p_{kx}} + T_{jx} h_{p_{ky}}$$

Tal y como pasaba con el problema de velocidad, el término TH_{jk} es exactamente el mismo y su construcción es la misma. Por tanto podemos empezar transformando las ecuaciones que rigen el problema de aceleración con estos nuevos términos

$$TH_{11}\alpha_{21} + TH_{21}\alpha_{31} + TH_{31}\alpha_{41} + \left(-\vec{\omega}_{21}^2 \overrightarrow{T_1} - \vec{\omega}_{31}^2 \overrightarrow{T_2} - \vec{\omega}_{41}^2 \overrightarrow{T_3} \right) \cdot \overrightarrow{h_{p_1}} = (\vec{a}_{41}^D) \cdot \overrightarrow{h_{p_1}}$$

$$TH_{42}\alpha_{51} + TH_{52}\alpha_{61} + \left(-\vec{\omega}_{51}^2 \overrightarrow{T_4} - \vec{\omega}_{61}^2 \overrightarrow{T_5} \right) \cdot \overrightarrow{h_{p_2}} = (-\vec{a}_{41}^D) \cdot \overrightarrow{h_{p_2}}$$

$$\vec{\alpha}_{31} = \vec{\alpha}_{41} \;\; ; \;\; \vec{\alpha}_{61} = \vec{\alpha}_{71} = 0$$

Como podemos observar la estructura que aparece es parecida a la que aparecía en el mismo problema en el caso de velocidad. Se trata de tantas *ecuaciones de grupos* como pares prismáticos tenga el mecanismo cada una formada a la izquierda por un conjunto de elementos TH_{ij} que multiplican a las aceleraciones angulares y a la derecha unas restas de aceleraciones angulares multiplicadas por los respectivos vectores perpendiculares a los pares prismáticos. Pero es destacable que, en este caso de aceleraciones, aparecen una serie de términos restando en la parte izquierda de la ecuación que no aparecían en el problema de velocidad. Se trata de unos términos del tipo $\vec{\omega}_{i1}^2 \overrightarrow{T_J}$ que están multiplicados por los vectores perpendiculares a cada respectivo par prismático. Cabe decir que todos estos términos no representan ninguna incógnita, son todos conocidos ya que se parte de haber resuelto el problema de velocidad.

Es decir que la estructura de las ecuaciones del problema de aceleración, por el momento, es muy parecida a las del problema de velocidad pero sumando, en la parte derecha de las ecuaciones una serie de términos conocidos.

Pasemos ahora a analizar cómo es la estructura matricial de estos nuevos términos y donde casan en el método de transformación a forma matricial.

$$\left(\vec{\omega}_{i1}^2 \overrightarrow{T_J}\right) \cdot \overrightarrow{h_{p_k}} \qquad\qquad Siendo: \; \overrightarrow{T_J} = \begin{pmatrix} T_{j_x} \\ T_{j_y} \\ 0 \end{pmatrix} \;\; ; \;\; \overrightarrow{h_{p_k}} = \begin{pmatrix} h_{p_{kx}} \\ h_{p_{ky}} \\ 0 \end{pmatrix} \;\; ; \;\; \vec{\omega}_{i1} = \begin{pmatrix} 0 \\ 0 \\ \omega_{i1} \end{pmatrix}$$

$$\left(\omega_{i1}^2 \begin{pmatrix} T_{jx} \\ T_{jy} \end{pmatrix} \right) \cdot \overrightarrow{h_{p_k}} = \begin{pmatrix} T_{j_x} \\ T_{j_y} \end{pmatrix} \cdot \begin{pmatrix} h_{p_{kx}} \\ h_{p_{ky}} \end{pmatrix} \omega_{i1} = \left(T_{jx} h_{p_{kx}} + T_{jy} h_{p_{ky}} \right) \omega_{i1} = WTH_{ijk}$$

$$WTH_{ijk} = \left(T_{jx} h_{p_{kx}} + T_{jy} h_{p_{ky}} \right) \omega_{i1}$$

El nuevo término WTH_{ijk} será conocido y poseerá una estructura matricial con tantas filas como grupos existan (tantas como número de pares prismáticos) y tantas columnas como términos haya en total el todos los grupos.

A continuación se van a exponer las ecuaciones matriciales generales para cualquier mecanismo añadiendo los nuevos términos que impone el problema de aceleración. Cabe decir que aún no se ha descrito cómo cambiaría el problema de aceleraciones si aparecen pares de rodadura sin deslizamiento en el lazo del mecanismo. Se describirá posteriormente ya que resulta más fácil a la hora de comprender este proceso.

$$\begin{cases} (\vec{\alpha}_{21} \wedge \vec{T}_1 + \vec{\alpha}_{31} \wedge \vec{T}_4 + \vec{\alpha}_{41} \wedge \vec{T}_3 + \cdots) \cdot \overrightarrow{h_{p_1}} - (\vec{\omega}_{21}^2 \overrightarrow{T_1} + \vec{\omega}_{31}^2 \overrightarrow{T_2} + \vec{\omega}_{41}^2 \overrightarrow{T_3} + \cdots) \cdot \overrightarrow{h_{p_1}} = \cdots \\ (\vec{\alpha}_{61} \wedge \vec{T}_5 + \alpha_{71} \wedge \vec{T}_6 + \cdots) \cdot \overrightarrow{h_{p_2}} - (\vec{\omega}_{61}^2 \overrightarrow{T_5} + \vec{\omega}_{71}^2 \overrightarrow{T_6} + \cdots) \cdot \overrightarrow{h_{p_2}} = \cdots \\ (\vec{\alpha}_{91} \wedge \vec{T}_8 + \cdots) \cdot \overrightarrow{h_{p_3}} - (\vec{\omega}_{91}^2 \overrightarrow{T_8} + \cdots) \cdot \overrightarrow{h_{p_3}} = \cdots \\ \vdots \end{cases}$$

$$\rightarrow \begin{cases} \left(\sum_{i,\ j}^{n,\ m} (\vec{\alpha}_{i1} \wedge \vec{T}_j) \right) \cdot \overrightarrow{h_{p_k}} - \left(\sum_{i,\ j}^{n,\ m} (\vec{\omega}_{i1}^2 \vec{T}_j) \right) \cdot \overrightarrow{h_{p_k}} = \cdots \\ \vdots \end{cases}$$

Siendo i , j y k los índices que reflejan las correspondientes velocidades y aceleraciones angulares, tramos y direcciones perpendiculares a los pares prismáticos respectivamente. Habrá tantas ecuaciones como pares prismáticos tenga el lazo.

Si se aplican las definiciones de TH y de WTH obtendríamos algo así:

$$\begin{cases} (TH_{11}\alpha_{21} + TH_{21}\alpha_{31} + TH_{31}\alpha_{41} + \cdots) - (WTH_{211} + WTH_{321} + WTH_{431} + \cdots) = \cdots \\ (TH_{52}\alpha_{61} + TH_{62}\alpha_{71} + \cdots) - (WTH_{652} + WTH_{762} + \cdots) = \cdots \\ (TH_{83}\alpha_{91} + \cdots) - (WTH_{983} + \cdots) = \cdots \\ \vdots \end{cases}$$

$$\rightarrow \begin{cases} \sum_{i,\ j}^{n,\ m} (TH_{jk}\alpha_{i1}) - \sum_{i,\ j}^{n,\ m} (WTH_{ijk}) = \cdots \\ \vdots \end{cases}$$

$$\begin{pmatrix} TH_{11} & \cdots & TH_{m1} \\ \vdots & \ddots & \vdots \\ TH_{1k} & \cdots & TH_{mk} \end{pmatrix} \begin{pmatrix} \alpha_{21} \\ \alpha_{31} \\ \vdots \\ \alpha_{n1} \end{pmatrix} - \begin{pmatrix} \sum_{i,\ j}^{n,\ m} (WTH_{ij1}) \\ \vdots \\ \sum_{i,\ j}^{n,\ m} (WTH_{ijk}) \end{pmatrix} = TH\ \alpha - WTH = \cdots$$

Para ilustrar lo explicado anteriormente se expone un ejemplo. Se transformará el conjunto de ecuaciones del lazo de la figura 1.

$$\begin{gathered} (TH_{11}\alpha_{21} + TH_{21}\alpha_{31} + TH_{31}\alpha_{41}) - (WTH_{211} + WTH_{321} + WTH_{431}) = \cdots \\ (TH_{42}\alpha_{51} + TH_{52}\alpha_{61}) - (WTH_{542} + WTH_{652}) = \cdots \end{gathered}$$

$$\downarrow$$

$$\begin{pmatrix} TH_{11} & TH_{21} & TH_{31} & 0 & 0 \\ 0 & 0 & 0 & TH_{42} & TH_{52} \end{pmatrix} \alpha - \begin{pmatrix} WTH_{211} + WTH_{321} + WTH_{431} \\ WTH_{542} + WTH_{652} \end{pmatrix} = \begin{pmatrix} \cdots \\ \cdots \end{pmatrix}$$

Pasamos ahora a transformar la parte de la derecha de la igualdad de las ecuaciones tal y como se hizo en el problema de velocidad. El procedimiento es análogo. En este caso tendremos una resta de dos multiplicaciones de los términos de un vector de aceleración y un vector de dirección.

$$\begin{cases} \cdots = (\vec{a}_{41}^{D}) \cdot \overrightarrow{h_{p_1}} \\ \cdots = (\vec{a}_{61}^{F} - \vec{a}_{41}^{D}) \cdot \overrightarrow{h_{p_2}} \\ \cdots = (\vec{a}_{81}^{H} - \vec{a}_{61}^{F}) \cdot \overrightarrow{h_{p_3}} \\ \quad . \\ \quad . \\ \quad . \end{cases} \rightarrow \begin{cases} (\vec{a}_{v1}^{P2} - \vec{a}_{u1}^{P1}) \cdot \overrightarrow{h_{p_k}} \\ \quad . \\ \quad . \\ \quad . \end{cases}$$

$$(\vec{a}_{41}^{D}) \cdot \overrightarrow{h_{p_1}} = a_{41_x}^{D} h_{p1_x} + a_{41_y}^{D} h_{p1_y} \qquad (\vec{a}_{61}^{F} - \vec{a}_{41}^{D}) \cdot \overrightarrow{h_{p_2}} = -h_{p2_x} a_{41_x}^{D} - h_{p2_y} a_{41_y}^{D} + h_{p2_x} a_{61_x}^{F} + h_{p2_y} a_{61_y}^{F}$$

$$(\vec{a}_{81}^{H} - \vec{a}_{61}^{F}) \cdot \overrightarrow{h_{p_3}} = -h_{p3_x} a_{61_x}^{F} - h_{p3_y} a_{61_y}^{F} + h_{p3_x} a_{81_x}^{H} + h_{p3_y} a_{81_y}^{H}$$

$$\begin{cases} a_{41_x}^{D} h_{p1_x} + a_{41_y}^{D} h_{p1_y} \\ -h_{p2_x} a_{41_x}^{D} - h_{p2_y} a_{41_y}^{D} + h_{p2_x} a_{61_x}^{F} + h_{p2_y} a_{61_y}^{F} \\ -h_{p3_x} a_{61_x}^{F} - h_{p3_y} a_{61_y}^{F} + h_{p3_x} a_{81_x}^{H} + h_{p3_y} a_{81_y}^{H} \\ \quad . \\ \quad . \\ \quad . \end{cases}$$

Si expresamos las ecuaciones anteriores en forma matricial obtendremos dos matrices. La primera se formularía con las direcciones perpendiculares de los pares prismáticos, a esta matriz *H* es exactamente la misma que se definió en el problema de velocidad. La segunda matriz que diferenciamos se trata de un vector con las incógnitas de las aceleraciones lineales de los pares prismáticos del mecanismo, lo llamaremos *A*.

$$\begin{pmatrix} h_{p1_x} & h_{p1_y} & 0 & 0 & 0 & 0 & & & \\ -h_{p2_x} & -h_{p2_y} & h_{p2_x} & h_{p2_y} & 0 & 0 & \cdot & \cdot & \cdot \\ 0 & 0 & -h_{p3_x} & -h_{p3_y} & h_{p3_x} & h_{p3_y} & & & \\ & & & \cdot & & & \cdot & & \\ & & & \cdot & & & & \cdot & \\ & & & \cdot & & & & & \cdot \end{pmatrix} \begin{pmatrix} a_{41_x}^{D} \\ a_{41_y}^{D} \\ a_{61_x}^{F} \\ a_{61_y}^{F} \\ a_{81_x}^{H} \\ a_{81_y}^{H} \\ \cdot \\ \cdot \\ \cdot \end{pmatrix} = \begin{pmatrix} \overrightarrow{h_{p_1}}' & 0 & 0 & & & \\ -\overrightarrow{h_{p_2}}' & \overrightarrow{h_{p_2}}' & 0 & \cdot & \cdot & \cdot \\ 0 & -\overrightarrow{h_{p_3}}' & \overrightarrow{h_{p_3}}' & & & \\ & \cdot & & \cdot & & \\ & \cdot & & & \cdot & \\ & \cdot & & & & \cdot \end{pmatrix} \begin{pmatrix} \vec{a}_{41}^{D} \\ \vec{a}_{61}^{F} \\ \vec{a}_{81}^{H} \\ \cdot \\ \cdot \\ \cdot \end{pmatrix}$$

$$\begin{pmatrix} \overrightarrow{h_{p_1}}' & 0 & 0 & & & \\ -\overrightarrow{h_{p_2}}' & \overrightarrow{h_{p_2}}' & 0 & \cdot & \cdot & \cdot \\ 0 & -\overrightarrow{h_{p_3}}' & \overrightarrow{h_{p_3}}' & & & \\ & \cdot & & \cdot & & \\ & \cdot & & & \cdot & \\ & \cdot & & & & \cdot \end{pmatrix} = H \qquad \begin{pmatrix} \vec{a}_{41}^{D} \\ \vec{a}_{61}^{F} \\ \vec{a}_{81}^{H} \\ \cdot \\ \cdot \\ \cdot \end{pmatrix} = A$$

La matriz H es exactamente igual para el caso de velocidad que para el caso de aceleración. El vector de incógnitas de aceleraciones lineales A mantiene la misma estructura que su homólogo en velocidades. Siendo así y asumiendo que el procedimiento la gran similitud de ambos problemas se expone a continuación una justificación de cómo resultaría el problema global de aceleraciones para lazos de mecanismos planos.

$$TH\ \alpha = H\ A + WTH \qquad (TH\ \vdots\ H)\begin{pmatrix}\alpha\\ \cdots\\ A\end{pmatrix} = WTH$$

Estas son las ecuaciones matriciales que resultan de agrupar las incógnitas de aceleración en un solo vector. Como se recordará del problema de velocidades a todo estas ecuaciones hay que implantarles las *ecuaciones de condición prismática* del mecanismo $(\vec{\alpha}_{i1} = \vec{\alpha}_{j1})$. Una vez añadidas (recordemos que esto se basaba en sumar columnas de la matriz $(TH\ \vdots\ H)$ e igualar a ceros las columnas utilizadas) deberemos también implantar las condiciones de velocidades iniciales impuestas al mecanismo. Todo esto en el problema de velocidad representaba una transformación de las ecuaciones matriciales hasta llegar a:

$$C\ X_V = B$$

En este caso de aceleraciones el problema queda muy similar. La matriz definida C no varía pero en la aparte de la derecha de la igualdad se incluyen unos términos provenientes tanto de implantar las aceleraciones iniciales como de la matriz WTH ya definida.

$$CX_a = D + WTH$$

Siendo:

$$D = -\begin{pmatrix} TH_{11}^{cp} & TH_{12}^{cp} & \dots & TH_{1m}^{cp} & H_{11} & H_{12} & \dots & H_{1q} \\ TH_{21}^{cp} & \ddots & & \vdots & H_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ TH_{n1}^{cp} & \dots & \dots & TH_{nm}^{cp} & H_{p1} & \dots & \dots & H_{pq} \end{pmatrix}\begin{pmatrix}0\\3\\\vdots\\4\\0\\0\\5\\\vdots\end{pmatrix} = -(TH^{cp}\ \vdots\ H)\varphi_a = -A_{cp}\varphi_a$$

El vector φ_a se define como el vector de aceleraciones iniciales impuestas al mecanismo.

El problema parece quedar cerrado con estas ecuaciones pero es necesario advertir que estas solo sirven para lazos de mecanismos compuestos solo por pares de rotación y pares prismáticos. Es decir, todavía queda introducir a estas ecuaciones las restricciones de los pares de rodadura sin deslizamiento. Se ha elegido estas alturas del texto para incluirlas y no antes porque partiendo de la idea que se ha querido enseñar en este punto: *el problema de aceleraciones es muy similar al de velocidades, solo cambia la implementación de las aceleraciones lineales e incluir la matriz WTH como suma en el término independiente.* Se estudiará ahora un lazo que contenga un par de rodadura sin deslizamiento para poder entender cómo añadir esas ecuaciones a nuestro caso general. El ejemplo elegido es la Figura 20 del capítulo 3.

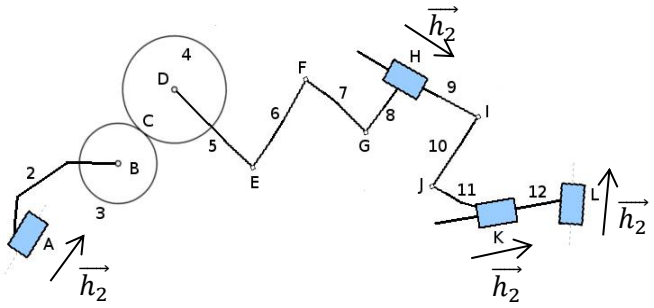


Figura 31

$$\begin{pmatrix} \vec{\alpha}_{21} \wedge \overrightarrow{T_1} + \vec{\alpha}_{31} \wedge \overrightarrow{T_2} + \vec{\alpha}_{41} \wedge \overrightarrow{T_3} + \vec{\alpha}_{51} \wedge \overrightarrow{T_4} + \vec{\alpha}_{61} \wedge \overrightarrow{T_5} + \vec{\alpha}_{71} \wedge \overrightarrow{T_6} + \vec{\alpha}_{81} \wedge \overrightarrow{T_7} \\ + \\ -\vec{\omega}_{21}^2 \overrightarrow{T_1} - \vec{\omega}_{31}^2 \overrightarrow{T_2} - \vec{\omega}_{41}^2 \overrightarrow{T_3} - \vec{\omega}_{51}^2 \overrightarrow{T_4} - \vec{\omega}_{61}^2 \overrightarrow{T_5} - \vec{\omega}_{71}^2 \overrightarrow{T_6} - \vec{\omega}_{81}^2 \overrightarrow{T_7} \\ + \\ -\vec{a}_{34}^C - 2\vec{\omega}_{41} \wedge \vec{V}_{31}^C \end{pmatrix} \cdot \overrightarrow{h_{p_1}} = (\vec{a}_{81}^H) \cdot \overrightarrow{h_{p_1}}$$

$$\begin{pmatrix} \vec{\alpha}_{91} \wedge \overrightarrow{T_8} + \vec{\alpha}_{10,1} \wedge \overrightarrow{T_9} + \vec{\alpha}_{11,1} \wedge \overrightarrow{T_{10}} \\ + \\ -\vec{\omega}_{91}^2 \overrightarrow{T_8} - \vec{\omega}_{10,1}^2 \overrightarrow{T_9} - \vec{\omega}_{11,1}^2 \overrightarrow{T_{10}} \end{pmatrix} \cdot \overrightarrow{h_{p_2}} = \left(\vec{a}_{11,1}^K - \vec{a}_{81}^H\right) \cdot \overrightarrow{h_{p_2}}$$

$$\left(\vec{\alpha}_{12,1} \wedge \overrightarrow{T_{11}} - \vec{\omega}_{12,1}^2 \overrightarrow{T_{11}}\right) \cdot \overrightarrow{h_{p_3}} = \left(\vec{a}_{12,1}^L - \vec{a}_{11,1}^K\right) \cdot \overrightarrow{h_{p_3}}$$

$$0 = \left(\vec{a}_{12,1}^L\right) \cdot \overrightarrow{h_{p_4}}$$

$$\vec{\alpha}_{21} = 0 \;\; ; \;\; \vec{\alpha}_{81} = \vec{\alpha}_{91} \;\; ; \;\; \vec{\alpha}_{11,1} = \vec{\alpha}_{12,1} = 0$$

Como se puede observar las ecuaciones tienen la misma estructura con la salvedad de incluir dos términos restando en la parte izquierda de la primera ecuación. Esto se debe a que, para ese grupo, existe un par de rodadura sin deslizamiento. Este hecho puede generalizarse, es decir, que para un grupo en el que exista uno o varios pares de rodadura sin deslizamiento se deberán incluir por cada para dos términos restando del tipo $\vec{a}_{ij}^C + 2\vec{\omega}_{j1} \wedge \vec{V}_{i1}^C$ siendo $i\ y\ j$ los respectivos eslabones que une el par de rodadura sin deslizamiento. En los grupos en los que no exista un par de rodadura sin deslizamiento no deberá añadirse ningún término adicional a parte de los propios ya explicados.

Como un lazo genérico puede poseer cualquier número de grupos y tener distribuidos cualquier combinación de pares de rodadura sin deslizamiento en ellos, pasaremos a describir cómo serían los términos $\left(\vec{a}_{ij}^C + 2\vec{\omega}_{j1} \wedge \vec{V}_{i1}^C\right) \cdot \overrightarrow{h_{p_k}}$ en forma matricial. De esta forma podremos cerrar por completo el problema de aceleración.

$$\left(\vec{a}_{ij}^C + 2\vec{\omega}_{j1} \wedge \vec{V}_{i1}^C\right) \cdot \overrightarrow{h_{p_k}}$$

$$Siendo:\ \vec{V}_{i1}^C = \begin{pmatrix} V_{i1_x}^C \\ V_{i1_y}^C \\ 0 \end{pmatrix} \;\; ; \;\; \overrightarrow{h_{p_k}} = \begin{pmatrix} h_{p_{kx}} \\ h_{p_{ky}} \\ 0 \end{pmatrix} \;\; ; \;\; \vec{\omega}_{j1} = \begin{pmatrix} 0 \\ 0 \\ \omega_{j1} \end{pmatrix} \;\; ; \;\; \vec{a}_{ij}^C = \begin{pmatrix} a_{ij_x}^C \\ a_{ij_y}^C \\ 0 \end{pmatrix}$$

$$De\ forma\ generica \rightarrow \;\; \vec{a}_{ij}^C = \left(\vec{\omega}_{i1} - \vec{\omega}_{j1}\right)^2 \frac{R_i R_j}{R_i \pm R_j} \vec{\lambda}$$

La aceleración $\vec{a}_{ij}^{C}$ se definirá como la aceleración lineal relativa en el punto de contacto del par de rodadura sin rozamiento. Este dato será conocido y vendrá definido por la tipología del par de rodadura sin deslizamiento que se dé. El vector unitario de dirección $\vec{\lambda}$ también deberá definirse y siempre será un vector perpendicular a la dirección natura de rodadura del par apuntando al eslabón anterior en la secuencia natural de descripción del lazo del mecanismo.

$$\left(\vec{a}_{ij}^{C} + 2\vec{\omega}_{j1} \wedge \vec{V}_{i1}^{C}\right) \cdot \overrightarrow{h_{p}}_{k} = \left(\vec{a}_{ij}^{C}\right) \cdot \overrightarrow{h_{p}}_{k} + \left(2\vec{\omega}_{j1} \wedge \vec{V}_{i1}^{C}\right) \cdot \overrightarrow{h_{p}}_{k}$$

$$\left(\vec{a}_{ij}^{C}\right) \cdot \overrightarrow{h_{p}}_{k} = {a_{ij}^{C}}_{x} h_{p_{kx}} + {a_{ij}^{C}}_{y} h_{p_{ky}} = AH_{ijk}$$

$$\left(2\vec{\omega}_{j1} \wedge \vec{V}_{i1}^{C}\right) \cdot \overrightarrow{h_{p}}_{k} = 2\begin{vmatrix} \vec{\imath} & \vec{\jmath} & \vec{k} \\ 0 & 0 & \omega_{i1} \\ {V_{i1}^{C}}_{x} & {V_{i1}^{C}}_{y} & 0 \end{vmatrix} \cdot \begin{pmatrix} h_{p_{kx}} \\ h_{p_{ky}} \end{pmatrix} = 2\begin{pmatrix} -{V_{i1}^{C}}_{y} \\ {V_{i1}^{C}}_{x} \end{pmatrix} \omega_{i1} \cdot \begin{pmatrix} h_{p_{kx}} \\ h_{p_{ky}} \end{pmatrix} =$$

$$= 2\left(-{V_{i1}^{C}}_{y} h_{p_{kx}} + {V_{i1}^{C}}_{x} h_{p_{ky}}\right) \omega_{i1} = WVH_{ijk}$$

Se definen por tanto dos nuevos términos conocidos que son necesarios pasar a forma matricial para poder incluirlos en el cálculo. Estos dos nuevos términos serán dos vectores columna con tantas filas como grupos tenga el mecanismo.

$$\begin{pmatrix} \sum_{i,\ j}^{n,\ m} \left(AH_{ij1}\right) \\ \vdots \\ \sum_{i,\ j}^{n,\ m} \left(AH_{ijk}\right) \end{pmatrix} = AH \qquad \begin{pmatrix} \sum_{i,\ j}^{n,\ m} \left(WVH_{ij1}\right) \\ \vdots \\ \sum_{i,\ j}^{n,\ m} \left(WVH_{ijk}\right) \end{pmatrix} = WVH$$

Si bien puede demostrarse, estos dos nuevos términos irán sumando al término independiente de la ecuación general del problema de aceleración. Queda de esta manera cerrado el problema y generalizado para los tres tipos de pares a estudio en este documento.

$$\boxed{CX_a = D + WTH + AH + WVH}$$

A modo de ejemplo se resolverán las ecuaciones del último ejemplo expuesto en este capítulo.

Esta sería la matriz TH que multiplica al vector de incógnitas α.

$$\begin{pmatrix} TH_{11} & TH_{12} & TH_{13} & TH_{14} & TH_{15} & TH_{16} & TH_{17} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & TH_{28} & TH_{29} & TH_{2,10} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & TH_{3,11} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} = TH$$

Dónde:

$$TH_{11} = \begin{pmatrix} -T_{1y} \\ T_{1x} \end{pmatrix} \cdot \begin{pmatrix} h_{p_{1x}} \\ h_{p_{1y}} \end{pmatrix} \quad ; \quad TH_{28} = \begin{pmatrix} -T_{8y} \\ T_{8x} \end{pmatrix} \cdot \begin{pmatrix} h_{p_{2x}} \\ h_{p_{2y}} \end{pmatrix} \quad \cdots etc.$$

A continuación se muestra el llamado vector WTH:

$$\begin{pmatrix} WTH_{211} + WTH_{321} + WTH_{431} + WTH_{541} + WTH_{651} + WTH_{761} + WTH_{871} \\ WTH_{982} + WTH_{10,92} + WTH_{11,10,2} \\ WTH_{12,11,3} \\ 0 \end{pmatrix} = WTH$$

Dónde:

$$WTH_{211} = \left(T_{1x}h_{p_{1x}} + T_{1y}h_{p_{1y}}\right)\omega_{21} \;\; ; \;\; WTH_{982} = \left(T_{8x}h_{p_{2x}} + T_{8y}h_{p_{2y}}\right)\omega_{91} \;\; \cdots \;\; etc.$$

La matriz H, correspondiente a la distribución de los pares prismáticos sería:

$$\begin{pmatrix} h_{p1_x} & h_{p1_y} & 0 & 0 & 0 & 0 & 0 & 0 \\ -h_{p2_x} & -h_{p2_y} & h_{p2_x} & h_{p2_y} & 0 & 0 & 0 & 0 \\ 0 & 0 & -h_{p3_x} & -h_{p3_y} & h_{p3_x} & h_{p3_y} & 0 & 0 \\ 0 & 0 & 0 & 0 & -h_{p4_x} & -h_{p4_y} & h_{p4_x} & h_{p4_y} \end{pmatrix} = H$$

Llegados este punto es necesario imponer las ecuaciones de condición prismática ($\vec{\alpha}_{i1} = \vec{\alpha}_{j1}$) a nuestra matriz TH. También calcularemos el término independiente obtenido de la aplicación de las condiciones iniciales de aceleración (vector φ_a) y al que denominaremos D.

$$\vec{\alpha}_{21} = 0 \;\; ; \;\; \vec{\alpha}_{81} = \vec{\alpha}_{91} \;\; ; \;\; \vec{\alpha}_{11,1} = \vec{\alpha}_{12,1} = 0$$

$$\begin{pmatrix} 0 & TH_{12} & TH_{13} & TH_{14} & TH_{15} & TH_{16} & TH_{17} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & TH_{28} & 0 & TH_{29} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} = TH^{cp}$$

$$(TH^{cp} \;\; \vdots \;\; H)\varphi_a = D$$

Cómo el mecanismo posee un par de rodadura sin deslizamiento es necesario calcular los vectores AH y WVH con los que podremos cerrar el problema.

$$\begin{pmatrix} (\vec{a}_{34}^{C}) \cdot \overrightarrow{h_{p_1}} \\ 0 \\ 0 \\ 0 \end{pmatrix} = AH \qquad \begin{pmatrix} 2\left(\vec{\omega}_{41} \wedge \vec{V}_{31}^{C}\right) \cdot \overrightarrow{h_{p_1}} \\ 0 \\ 0 \\ 0 \end{pmatrix} = WVH$$

$$\vec{a}_{34}^{C} = (\vec{\omega}_{31} - \vec{\omega}_{41})^2 \frac{R_3 R_4}{R_3 - R_4}\vec{\lambda}$$

Con todas estas matrices ya podemos resolver el problema de aceleración. Simplemente se planteará la siguiente ecuación donde X_a es el vector de incógnitas de aceleración del mecanismo.

$$C = (TH^{cp} \quad \vdots \quad H)$$

$$CX_a = D + WTH + AH + WVH$$

5 EXPLICACIÓN DEL CÓDIGO DEL PROGRAMA

Una vez descritas las ecuaciones que rigen la cinemática de mecanismos plana y cómo resolverlas de manera generalizada, se plantea la creación de un programa de cálculo capaz de llevar acabo la manipulación y resolución de mecanismos planos.

5.1 Introducción al método y requisitos previos

El programa denominado *CAC 2D* (*Computational Analysis Cinematic of Mechanical 2D*) es creado para poder analizar mecanismos planos y resolver su cinemática (Problema de velocidad y problema de aceleración) así como poder representarlos, guardarlos, generar informes de los mecanismos incluso de simular su movimiento en el tiempo.

Se trata de un software creado a partir de código compilado en *Matlab*, un programa de cálculo y programación con diversas aplicaciones en diferentes ámbitos, destacando por su potencia en el cálculo de matrices. Es por eso que los métodos que se usarán serán enfocados a resolver el problema de modo matricial, aprovechando así la gran versatilidad de *Matlab.*

Todo se ha escrito y programado en el lenguaje de programación de *Matlab.* El programa creado cuenta con una interfaz gráfica que ayuda al usuario a introducir los datos y analizar los resultados. El programa es un cómputo de varios subprogramas encargados de tareas distintas con el objetivo de trabajar en común para resolver la cinemática del mecanismo plano a estudio.

En las siguientes hojas se explicará, de manera amena, la programación utilizada y como se ha realizado y ensamblado todo para crear *CAC 2D.* La filosofía de programación ha sido la división en programas principales y estos a su vez, compuestos por programas auxiliares de ayuda y complementación.

Se muestra a continuación una lista de los programas de código en *Matlab* utilizados discretizándolos por categorías:

5.1.1 Listado de funciones principales del programa CAC 2D

- **Funciones básicas previas**
  1. BucleAC.m (función que modifica los parámetros si se trata de un bucle abierto)
  2. Relaciones.m (Se crean los subgrupos "*Rela*" en la estructura *Meca*)
  3. CreaGrupos.m (Discretiza el lazo de los pares en grupos)

4. CreaTramos.m (Implanta los vectores de los tramos en *Meca* a partir de la geometría)

- **Funciones del problema de velocidad**
  1. NOPP.m (Resuelve el problema de velocidades en el caso particular de no existir pares prismáticos)
  2. MatrizTH.m (Genera la matriz TH)
  3. MatrizHV.m (Genera la matriz H)
  4. CondPrismatico.m (Implementa las condiciones prismáticas en la matriz M)
  5. CondVGDL.m (Implementa las condiciones de las velocidades iniciales conocidas)
  6. CondPrisSolV.m (Implementa las igualdades de los pares prismáticos en la solución del problema de velocidades)
  7. VelocRela.m (Desarrolla el método de las velocidades relativas para hallar las velocidades de todos los pares del mecanismo)

- **Funciones del problema de aceleración**
  1. MatrizWTH.m (Genera la matriz WTH)
  2. MatrizAH. m (Genera la matriz AH, solo si existen pares de rodadura sin deslizamiento)
  3. MatrizWVH.m (Genera la matriz WVH, solo si existen pares de rodadura sin deslizamiento)
  4. CondAGDL.m (Implementa las condiciones de las aceleraciones iniciales conocidas)
  5. CondPrisSolA.m (Implementa las igualdades de los pares prismáticos en la solución del problema de aceleraciones)
  6. AceleRela.m (Desarrolla el método de las aceleraciones relativas para hallar las aceleraciones de todos los pares del mecanismo)
  7. ProbAcel.m (Función capaz de calcular totas las aceleraciones del mecanismo)

- **Funciones para la representación gráfica**
  1. MuestraPares.m (Función que representa gráficamente los pares del mecanismo en la posición dada)
  2. MuestraCuerpos.m (Función que representa gráficamente los eslabones del mecanismo en la posición dada)
  3. MuestraVelocidades.m (Función que representa gráficamente los vectores velocidad de los pares del mecanismo en la posición dada)
  4. MuestraAceleraciones.m (Función que representa gráficamente los vectores aceleración de los pares del mecanismo en la posición dada)
  5. HazDibujo.m (Función capaz de realizar un esquema gráfico del mecanismo completo)

- **Funciones para la representación de la simulación de movimiento**

  (Las mismas funciones utilizadas para la representación gráfica del mecanismo más cuatro funciones adicionales)

  1. NuevasCoord.m (Función que calcula las coordenadas del punto siguiente en la simulación temporal del mecanismo)
  2. NuevasGDL.m (Función que calcula las nuevas velocidades y aceleraciones para la siguiente posición del mecanismo)
  3. Calculador.m (Función que resuelve el problema de velocidad y aceleración junto con las funciones vistas anteriormente)

4. HazSimulacion.m (Función capaz de realizar una simulación del movimiento del mecanismo)

- **Funciones auxiliares y de ayuda**
  1. CreaChar.m (Función que genera líneas 'char' de las velocidades y aceleraciones iniciales que se deben introducir en el mecanismo para poder ser resuelto)
  2. CompIni.m (Función que realiza 4 comprobaciones previas al cálculo referentes a la introducción de los datos)
  3. CompruebaVel.m (Función que comprueba la validez de los resultados obtenidos del problema de velocidad)
  4. CompruebaAce.m (Función que comprueba la validez de los resultados obtenidos del problema de aceleración)
  5. Guarda_Archivo.m (Función que implementa la opción de guardado de los datos del mecanismo en una carpeta)
  6. Carga_Archivo.m (Función que es capaz de recuperar los datos de un mecanismo previamente guardado)
  7. ListadoCuerpos.m (Función que genera una cadena 'char' con una lista de todos los eslabones que tiene el mecanismo)
  8. ListadoRelaciones.m (Función que genera una cadena 'char' con una lista de todos las relaciones que tiene el mecanismo)
  9. TransLazoPares.m (Función que transforma el lazo de los pares en el problema de velocidad en caso de existir un par de rodadura sin deslizamiento)
  10. CreaInforme.m (Función capaz de crear un documento 'pdf' con los datos de la resolución del mecanismo)
  11. Resuelve.m (Función capaz de resolver la cinemática global del mecanismo)

- **Funciones referentes a la interfaz gráfica**
  1. Prueba11.m (Interfaz principal de introducción de datos del programa)
  2. Entrada.m (Interfaz de inicio del programa)
  3. AyudaDirecPRSD (Interfaz de ayuda para la introducción de las direcciones de los pares de rodadura sin deslizamiento)
  4. AyudaRadios.m (Interfaz de ayuda para la introducción de los radios de curvatura de los pares de rodadura sin deslizamiento)
  5. PreSimulacion.m (Interfaz para la introducción de los datos previos a la simulación)
  6. MuestraResultados.m (Interfaz para la muestra de los resultados obtenidos del cálculo del problema así como su representación gráfica)
  7. AcercaDe.m (Interfaz de información del programa y de contacto con su creador)
  8. Condiciones_de_uso.m (Interfaz que muestra las condiciones de uso del programa al usuario)

5.2 Definiciones previas

Para facilitar la entrada, manipulación, cálculo y expresión de todos los datos referentes al mecanismo a estudio se ha creado una variable capaz de cumplir con todas estas expectativas. El programa de codificación *Matlab* nos permite la creación de multitud de tipos distintos de variables (variables entras, variables vectoriales, matriciales, variables tipo cadena de texto, etc.) por ello se ha creado la variable tipo estructura de

nombre "*Meca*". Esta variable estructura guardará todos los datos necesarios de nuestro mecanismo así como variables auxiliares que sean necesarias para la buena ejecución de código y del programa.

La variable "Meca" poseerá una lista de subvariables más que podrán ser de todo tipo. La manipulación de todos estos datos se vuelve sencilla puesto que para acceder a un campo de la estructura "*Meca*" tan solo tendremos que seguir el método sencillo que plantea la codificación de *Matlab*. Siendo así, para acceder a la subvariable que guarda la el número de eslabones del lazo solo tendríamos que introducir "*Meca.Nbody*".

A continuación se muestra una gráfica a modo de aclaración de cómo se compone esté "árbol de variables" que es la estructura *Meca:*

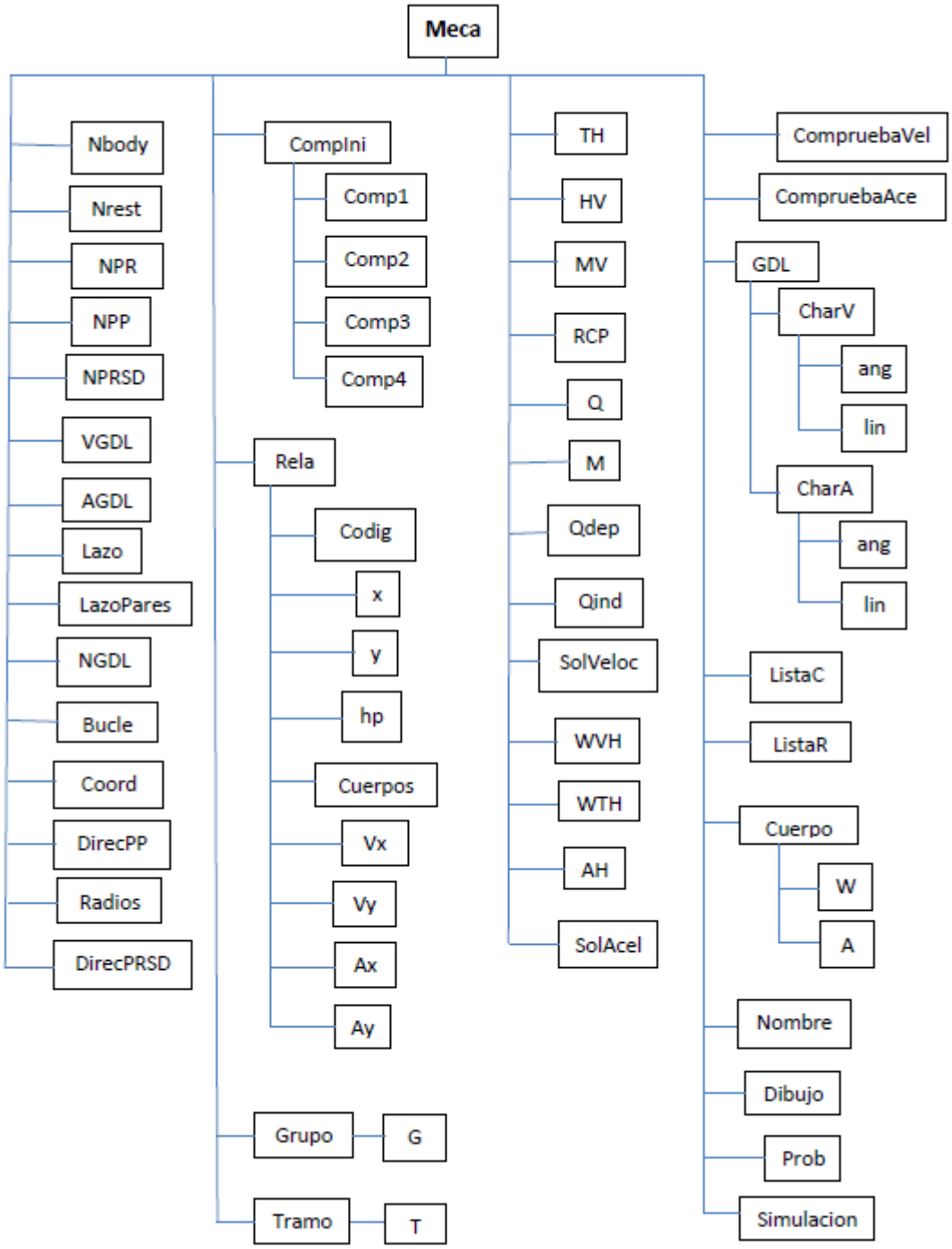


Figura 32

5.3 Requisitos de inicio

Toda la entrada y manipulación de los datos en el programa se realizará en forma numérica. Esto es así ya que es la forma más sencilla de manipulación de datos que presenta *Matlab* y para nuestro caso de estudio (Mecanismos planos) es necesario una descripción numérica del mecanismo (Coordenadas, velocidades iniciales, longitudes, etc.).

Esto hace necesario una codificación de las características del mecanismo en formato numérico sencillo y fácil de implementar, para que no altere la ya de por sí difícil programación de la resolución del problema. Esta codificación como todas las que sean necesarias realizar para la buena manipulación de los datos del mecanismo es fundamental que sea generalizable para cualquier mecanismo y dejando siempre al usuario la menor responsabilidad para su entendimiento. La introducción de los datos ha de ser sencilla para el usuario, realizando todas las codificaciones y descodificaciones que sean posibles en el código no observable ni editable por el usuario.

Se han realizado dos codificaciones sencillas para la introducción de los datos por parte del usuario. Una en la introducción del dato llamado "*Lazo del mecanismo"* y otra en la del dato "*Lazo de los pares del mecanismo"*. La primera se trata de la introducción de un vector de números enteros que debe iniciar en el 1 y acabar en el eslabón en el que acabe el lazo del mecanismo. El otro código se ha creado para facilitar la entrada de la secuencia de pares del lazo del mecanismo. Como se ha dicho ya en otra parte de este texto, la codificación utilizada es la siguiente:

1: Par de revolución

2: Par prismático

3: Par de rodadura sin deslizamiento

Estos números se emplearán, cuando procedan, en un vector numérico en el que se seguirá el mismo orden que el orden seguido en el vector "*Lazo del mecanismo*".

5.4 Funciones principales

Como se ha dicho anteriormente el programa consta de una serie de subprogramas capaces de calcular variables y resultados que en conjunto resuelven todo el problema cinemático plano del mecanismo. No sería eficiente explicar todos y cada uno de los programas de código de *Matlab* empleados para la resolución, guardado, análisis y de más funciones que tiene *CAC2D*. Pero es interesante plasmar en este documento algunos de los métodos y fórmulas que se emplean en los programas más destacados e importantes. Como resumen antes de hablar un poco de estos programas importantes si presenta el diagrama de flujo esencial del programa que nos da una idea de cómo seria los procedimientos que se siguen en el curso de desarrollo del programa.

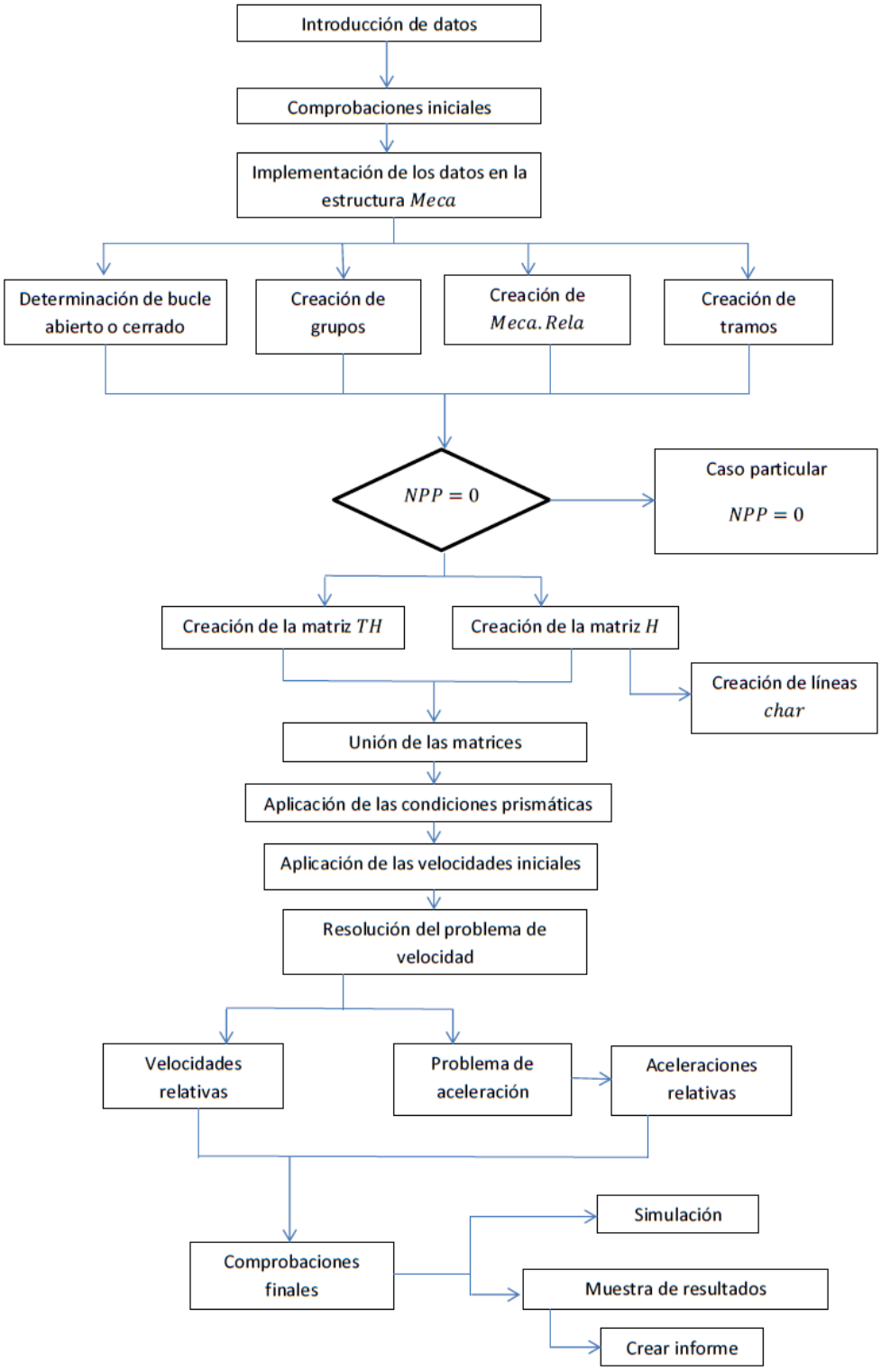


Figura 33

De los subprogramas más interesantes hay que destacar los que crean las matrices que hacen posible la resolución de los problemas de velocidad y aceleración. Estos subprogramas son capaces de crear las matrices a partir de los datos básicos de la estructura *Meca*, tales como las coordenadas, el lazo de los pares o el lazo del mecanismo, los vectores de grupo etc. Se trata de matrices muy importantes que apenas necesitan datos del mecanismo para ser definidas. Estamos hablando de matrices como *TH*, *H*, *WTH*, etc.

Otro subprograma muy importante es el encargado de crear los grupos del lazo del mecanismo. A través de la definición del lazo de los pares y del número de pares prismáticos, esta función es capaz de crear tantos grupos como pares prismáticos tenga el mecanismo. Como se vio en los casos particulares, si el lazo del mecanismo no posee ningún par prismático, es decir solo está compuesto por pares de revolución o pares de rodadura sin deslizamiento, este subprograma creará un solo grupo de acuerdo con la definición de los grupos del lazo de los pares. Cabe decir que la correcta definición de los grupos del lazo del mecanismo es fundamental para todas las posteriores definiciones de las matrices y las resoluciones de los problemas de velocidad y aceleración.

El método empleado para implementar el método de las velocidades y aceleraciones relativas es también un subprograma bastante destacado. El código de este subprograma se pensó para poder calcular las velocidades y aceleraciones lineales de todos los pares del mecanismo (partiendo de haber resuelto el problema de velocidad y aceleración, conociendo todas las velocidades y aceleraciones angulares de todos los eslabones del lazo), de una manera automática y generalizable para cualquier tipo de par o de lazo del mecanismo. Todo se basa en el método de las velocidades y aceleraciones relativas:

$$\vec{V}_{i1}^{Q} = \vec{V}_{i1}^{P} + \vec{\omega}_{i1} \wedge \overrightarrow{T_i}$$

$$\vec{a}_{i1}^{Q} = \vec{a}_{i1}^{P} + \vec{\alpha}_{i1} \wedge \overrightarrow{T_i} - \vec{\omega}_{i1}^{2} \overrightarrow{T_i}$$

Este método se ha generalizado para que pueda ser resuelto por *Matlab* independientemente del tipo de lazo. Partiendo del lazo y del lazo de los pares del mecanismo y habiendo resuelto los problemas de velocidad y aceleración, se disponen de ciertas velocidades y aceleraciones lineales conocidas a priori.

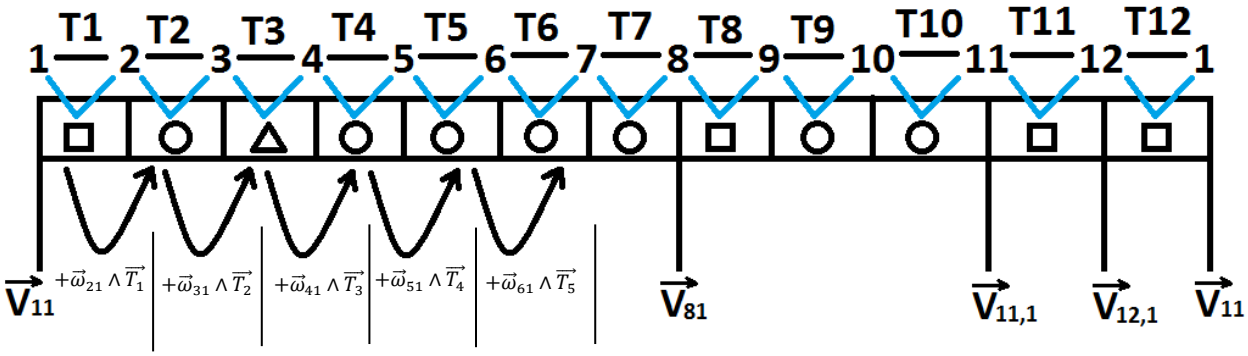


Figura 34

Como se intenta explicar en la figura anterior, para calcular la velocidad lineal de un par del mecanismo solo hay que conocer la velocidad lineal del par anterior y sumarle el término $\vec{\omega}_{i1} \wedge \overrightarrow{T_j}$ para averiguar la velocidad lineal del par en estudio. Los índices $i\ y\ j$ son respectivamente los encargados de representar los cuerpos y los tramos específicos de cada iteración.

Para el caso de las aceleraciones lineales es muy similar. En este caso sería necesario añadir a las aceleraciones del par anterior el término $\left(+\vec{\alpha}_{i1} \wedge \overrightarrow{T_j} - \vec{\omega}_{i1}\overrightarrow{T_j}\right)$ para obtener las aceleraciones lineales del par siguiente.

Esto es en lo que se ha basado el método de programación para calcular las velocidades y aceleraciones de cada par. Una vez calculadas todas se implementan en la estructura *Meca.Rela* donde se clasificarán según su naturaleza ($V_x, V_y, A_x\ y\ A_y$).

5.5 Explicación de la interfaz

La interfaz de usuario se trata de una ventana de introducción de datos principal y un conjunto de ventanas emergentes de aviso, ayuda, representación de resultados, etc. Para la mejora de la introducción de los datos así como la estancia del usuario con el programa se ha creado un manual de usuario siempre a disposición donde se explica, de manera extensa, todo lo referente a la introducción de los datos y como debe interpretarse

los resultados.

Toda la interfaz cuenta con botones interactivos, casillas de texto editable así como un menú de inicio en la parte superior izquierda del programa e imágenes descriptivas de conceptos básicos para el usuario. El usurario deberá introducir todos los datos que conozca del mecanismo que quiere analizar, siguiendo siempre las recomendaciones de las ayudas que el programa plantea.

La interfaz es un código escrito y ejecutado gracias a la herramienta de *Matlab* llamada "*GUIDE*" que aporta ayuda a la hora de la creación de nuevas interfaces. La interfaz se usa casi en su totalidad solo para la introducción de los datos necesarios para el cálculo. Es el programa creado en código Matlab el que resuelve el mecanismo y calcula todos los resultados. La interfaz también presenta al usuario opciones para la mejora del análisis del mecanismo como son el guardado y cargado del mecanismo, la representación gráfica, o la creación de un informe totalmente automático del mecanismo para poder tener un documento ".pdf" que refleje los resultados obtenidos.

De manera automática la interfaz crea dos carpetas donde se guardaran los datos de los mecanismos que usuario así decida. Las dos carpetas se llamarán "Archivos guardados" y "Informes de archivos" respectivamente. La primera carpeta se usará para guardar los datos de los mecanismos que se quieran cargar posteriormente. Esta carpeta presenta una ventaja ya que los mecanismos guardados en un ordenador podrían pasarse a otro donde podrían ser leídos por *CAC 2D* y ser resueltos de nuevo. Los archivos guardados de cada mecanismo se localizarán en carpetas en las que el usuario deberá respetar su contenido para no alterar los datos del mismo. La carpeta "Informes de archivos" será el lugar donde se guarden de manera automática los documentos ".pdf" referentes a los mecanismos estudiados.

A continuación se muestra una imagen de la interfaz utilizada y las distintas ventanas que la componen.

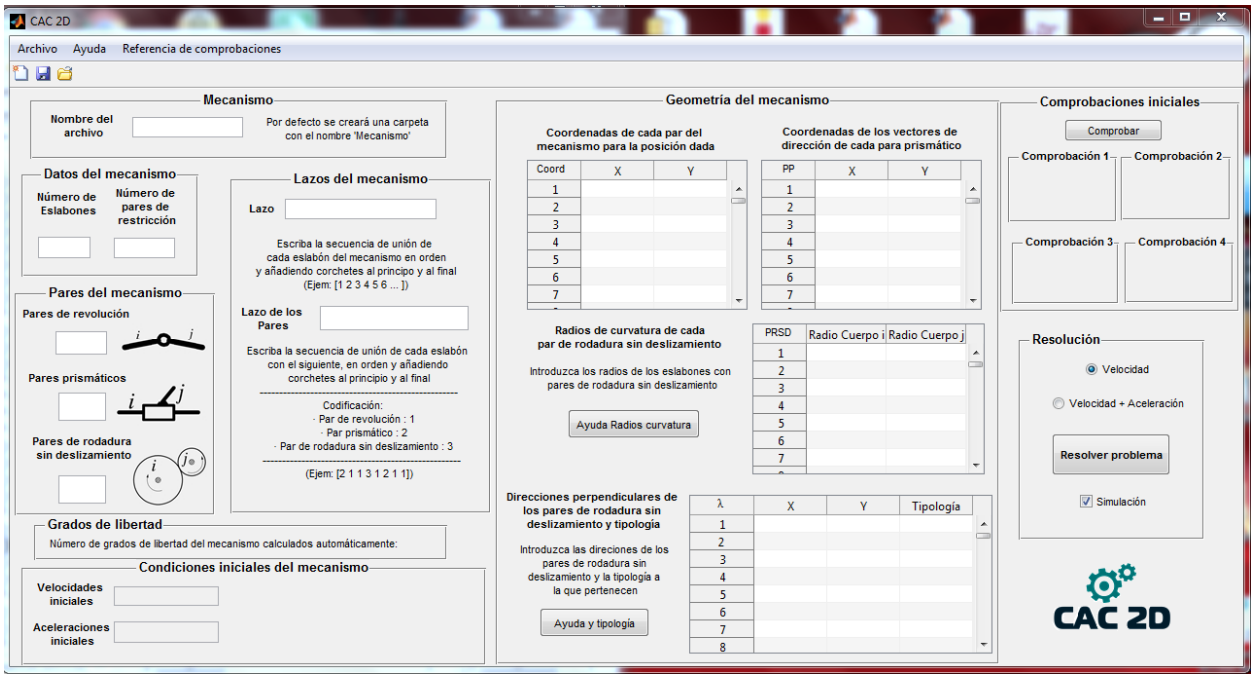


Figura 35

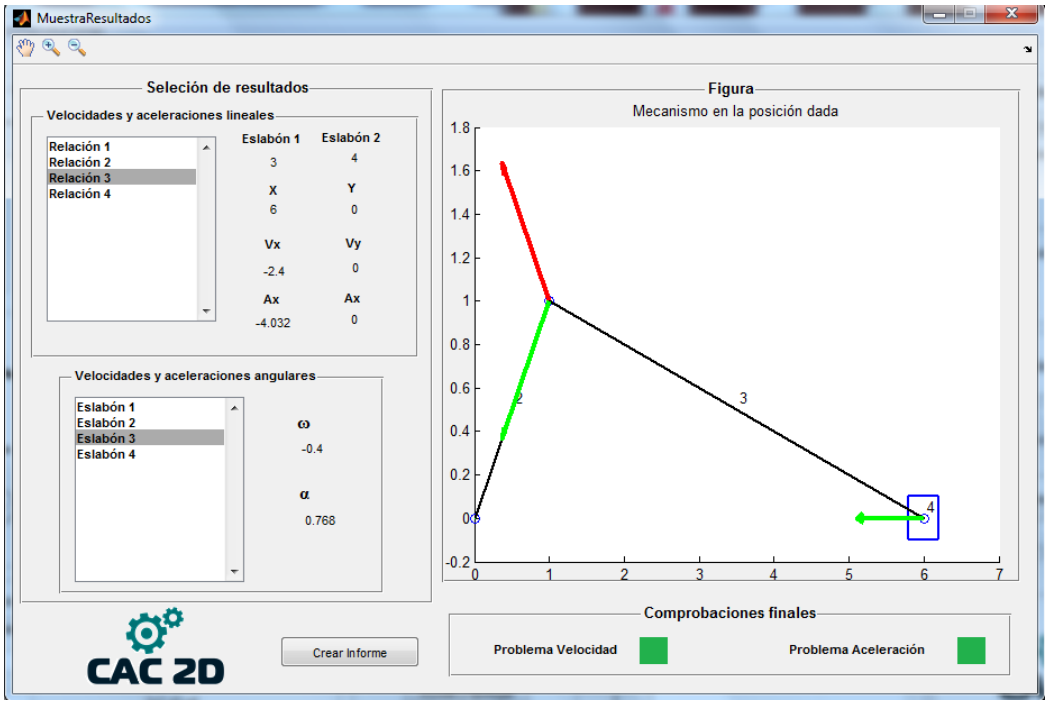


Figura 36

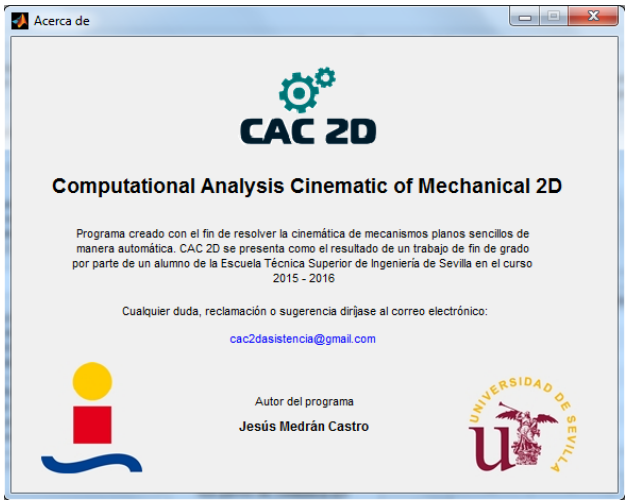


Figura 37

5.6 Limitaciones de programación

El programa *CAC 2D* cuenta con una serie de limitaciones referentes a la programación y a la interfaz del mismo. Así, como se ha explicado, el método de resolución que se emplea es un método matricial y como operación última se realiza una del tipo

$$Ax = B$$

El programa de cálculo *Matlab* es el que resuelve esta ecuación matricial con el operador '\' propio en su código. Este operador se encarga de resolver la ecuación por una serie de métodos propios y optimizados de los que *Matlab* se responsabiliza de su veracidad.

Otra limitación de programación son los posibles redondeos y errores contemplados en el cálculo de la

resolución del problema. A su vez se incluye que la interfaz sólo está preparada para registrar un número limitado de puntos de coordenadas, direcciones de pares prismáticos, radios de curvatura y direcciones de pares de rodadura sin deslizamiento. Este número máximo de puntos a incluir es 100 lo que permite una gran maniobra a la hora de analizar mecanismos sencillos.

5.7 Funciones auxiliares

Es necesario hablar de ciertas funciones que, a pesar de no poseer relevancia en cuanto al cálculo de las soluciones de los problemas de velocidad y aceleración, también realizan tareas que ayudan a la comprensión y al correcto manejo de los datos. Son funciones que no aportan nada al cálculo de la solución, sino más bien se presentan como subprogramas que partiendo de la solución del problema realizan sus tareas. Se trata de tareas como representar el mecanismo, guardarlo, cargarlo, generar una simulación, un informe, etc. En las próximas líneas analizaremos los procedimientos que siguen estos subprogramas ya que resultan interesantes y destacables.

Uno de los subprogramas a destacar son los encargados de guardar y cargar los datos del mecanismo. Esto permite a los usuarios salvar los datos de un mecanismo ya estudiado, volverlo a resolver, o transferir estos datos guardados a otro ordenador. Dentro de *Matlab* se plantean numerosos métodos para poder guardar variables y permanezcan a pesar de ser cerrado el programa. El método que se ha seguido es uno propio y programado para la mejora de la gestión del programa. Los datos del mecanismo, contenidos todos ellos en la estructura "*Meca*", se copian por separado en documentos tipo ".txt" con nombres los de la variable guardada ("*Nbody.txt*","*LazoPares.txt*","*NPRSD.txt*", etc.). Para un lazo estudiado tendremos 13 archivos "*.txt*" diferentes contenidos todos ellos en una carpeta denominada igual que el nombre que hayamos elegido para el mecanismo. Cada vez que guardemos un nuevo archivo se creará una nueva carpeta con estas condiciones, todas ellas estarán contenidas en una carpeta llamada "*Archivos guardados*" que se creará automáticamente en el mismo directorio en que el que este nuestro programa principal.

Cabe decir que el nombre que se le dé al mecanismo debe ser distinto a los mecanismos que ya han sido creados ya que si se guardará un mecanismo con el mismo nombre, el archivo de guardado anterior con ese nombre sería sustituido por el nuevo. Este método de guardado se presenta como una alternativa sencilla, adecuada y fácil de programar para un software de estas condiciones. Si se quieren ingresar nuevos mecanismos procedentes de otro ordenador simplemente bastaría con copiar la carpeta de ese mecanismo en la carpeta principal "*Archivos guardados*" para que pueda ser leída por nuestro programa.

El proceso de cargado está muy relacionado con lo escrito anteriormente. *CAC 2D* es capaz de leer el nombre que se lo ha dado y buscarlo en la carpeta "*Archivos guardados*". Una vez encontrado (si no existiera tal archivo este no podría cargarse y *CAC 2D* avisaría al usuario) se leerán y cargarán los datos al programa pudiendo verlos el usuario. Posteriormente se podrían modificar y resolver, si así se prefiere.

Otro subprograma destacable el encargado de crear un informe del mecanismo. El objetivo principal es crear un documento "*.pdf*" con los datos y los resultados del mecanismo. Esta tarea se consigue a través de un procedimiento en *Matlab* llamado "*Publish*". Esta función nativa de *Matlab* es capaz de crear diferentes tipos de documentos (pdf, html, etc.) del código del programa seleccionado así como plasmar en el mismo documento lo que se muestra por la pantalla principal de comandos de *Matlab.* Es esto último lo utilizado para el beneficio de crear un informe de los datos del mecanismo. Pacientemente se crea un programa de código en *Matlab* que se encarga de expresar los datos y resultados del mecanismo de manera clara y ordenada por la pantalla de comandos de *Matlab.* Posteriormente, y adecuando el proceso, se aplica la función "*Publish*" a dicho código. Esto genera un documento "*.pdf*" en el que sólo queda plasmado los datos y resultados clasificados por categorías de manera ordenada, quedando fuera de ese informe la publicación del código.

La misma estrategia se usa para añadir al informe una gráfica del mecanismo en la posición dada y otra que muestra la distribución de velocidades y aceleraciones. En este caso, al igual que pasaba con el guardado y el cargado, se necesitan conocimientos del proceso de guardado. Se define una carpeta llamada "*Informes de archivos*" que también se crea en el directorio raíz de *CAC 2D* donde se guardaran todas los informes de los mecanismos estudiados cada uno con su respectivo nombre.

Por último destacar dos subprogramas dedicados a la expresión gráfica de los mecanismos. Estos dos subprogramas se usan para generar una gráfica de la posición del mecanismo y, si el usuario lo decide, crear una simulación del movimiento natural del mecanismo. Las bases de la creación de estas gráficas están en los mecanismos de representación de *Matlab*. Para aplicarlos a nuestro objetivo se separa estratégicamente la representación gráfica de los eslabones, los pares del mecanismo y las representaciones de los vectores de velocidad y aceleración. Su estudio y representación por separado facilita el proceso y lo simplifica y generaliza. Posteriormente solo es necesario plasmar en una sola gráfica todos los resultados para poder obtener el dibujo completo del mecanismo. Es fundamental respetar las coordenadas implantadas por el usuario a la hora de la representación conjunta. El uso de colores y símbolos distintos también facilita su análisis.

Es necesario decir que la representación de los vectores de velocidad y aceleración no es la exacta que habría en el mecanismo. Si bien la representación de la posición y de los eslabones sí es la exacta para la posición dada, la de las velocidades y aceleraciones no. Esto es así para asegurar una correcta visualización del mecanismo por parte del usuario. Los vectores que en ella se representan coinciden en dirección y sentido con los reales pero su módulo es unidad. Esto se decidió así para que vectores de velocidades o aceleraciones muy grandes respecto a las dimensiones del mecanismo no afectaran a su visualización. También se distinguen dos colores, rojo para las velocidades y verde para las aceleraciones.

Como ya se ha mencionado anteriormente, *CAC 2D* es capaz de realizar una simulación del movimiento del mecanismo plano. Esto es así ya que, resulto el problema de velocidades y aceleraciones para la posición dada es posible estimar la siguiente posición a la que evolucionará el mecanismo. Las bases de esta estimación se exponen a continuación, junto con las hipótesis iniciales y las restricciones a las que está restringido el proceso de simulación.

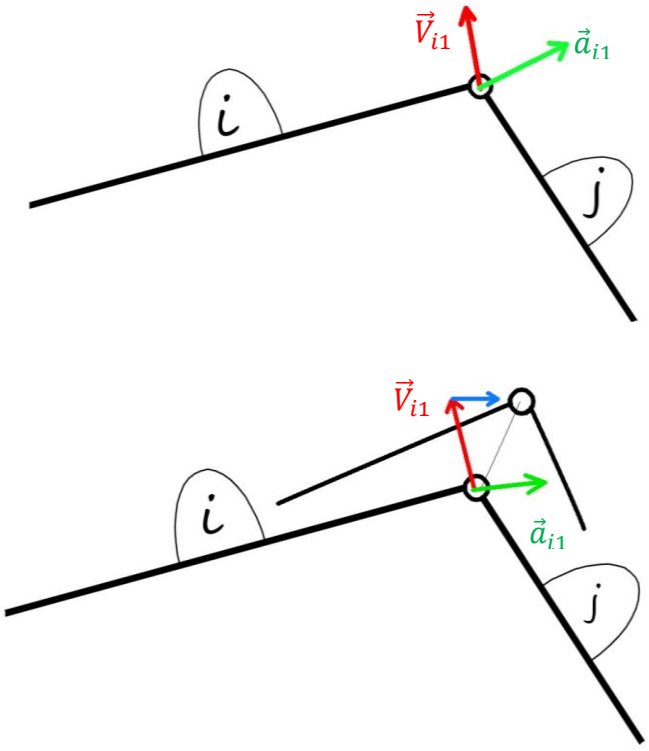


Figura 38

Partiendo de las velocidades y aceleraciones lineales de cada para se estima un incremento de tiempo muy pequeño (Δt) que servirá para la estimación de la posición siguiente. Así pues se asume la hipótesis de que la aceleración no variará en la próxima posición ya que es en un tiempo relativamente pequeño. Siendo así

se asume que el módulo del vector de aceleraciones marcará cuanto cambiará el módulo del vector velocidad para ese par. Así mismo con la dirección y sentido del vector aceleración también podremos obtener cuanto variará la dirección y sentido del vector velocidad.

Estos datos nos servirán para calcular un nuevo vector que denominaremos "*Vector de cambio de velocidad*" y que representaremos con $\overrightarrow{G_v}$. Para obtener la nueva velocidad del par que estamos estudiando simplemente sumaremos, al vector de velocidad antiguo de ese par el vector $\overrightarrow{G_v}$. Obtendremos así el nuevo vector de velocidad.

Solo faltaría hallar cuanto va a cambiar la posición para el par a estudio con esa nueva velocidad ya calculada. El módulo de la velocidad nueva ($\vec{V}'_{i1}$) multiplicado por el incremento de tiempo del principio (Δt) nos dará un valor escalar de cuánto se desplazará el par. Sin embargo si queremos averiguar la dirección y el sentido en el que se moverá deberemos multiplicar el vector de velocidad nuevo por ese pequeño incremento de tiempo. Con la dirección, el sentido y la magnitud que en ellos se desplaza, ya tenemos todos los datos para estimar cual será la próxima posición del par.

$$\begin{cases}\vec{a}_{i1}\\ \vec{V}_{i1}\end{cases} \rightarrow \begin{cases} |\vec{G}_v| = |\vec{a}_{i1}|\Delta t \\ \dfrac{\vec{G}_v}{|\vec{G}_v|} = Unit(\vec{G}_v) = Unit(\vec{a}_{i1}\Delta t) = \dfrac{\vec{a}_{i1}\Delta t}{|\vec{a}_{i1}\Delta t|}\end{cases}$$

$$\vec{V}'_{i1} = \vec{V}_{i1} + |\vec{G}_v| Unit(\vec{G}_v)$$

$$\begin{cases} |\vec{R}| = |\vec{V}'_{i1}|\Delta t \\ \dfrac{\vec{R}}{|\vec{R}|} = Unit(\vec{R}) = Unit(\vec{V}'_{i1}\Delta t) = \dfrac{\vec{V}'_{i1}\Delta t}{|\vec{V}'_{i1}\Delta t|}\end{cases}$$

$$Coordenadas\ actuales\ del\ par \rightarrow \vec{P}$$

$$Nuevas\ coordenadas\ del\ par \rightarrow \vec{P} + |\vec{R}| Unit(\vec{R})$$

Una vez calculadas las nuevas posiciones de todos los pares del mecanismo se debe resolver de nuevo toda la cinemática plana (problemas de velocidad y aceleración). Para ello deberemos estimar también cuanto variarán las velocidades y aceleraciones iniciales para esta nueva posición. Para este cálculo se realizan las mismas hipótesis anteriores. Las aceleraciones iniciales tanto angulares como lineales se mantendrán constantes en la próxima posición ya que se trata de un tiempo muy pequeño (Δt). Las nuevas velocidades iniciales angulares y lineales sí cambiarán. El proceso de cálculo será el mismo:

$$\begin{cases}\vec{\alpha}_{i1}\\ \vec{\omega}_{i1}\end{cases} \rightarrow G_\omega = |\vec{\alpha}_{i1}|\Delta t \rightarrow |\vec{\omega}'_{i1}| = |\vec{\omega}_{i1}| + G_\omega$$

El cálculo para los vectores de velocidades iniciales se realizaría del mismo modo que el calculado en los pares.

6 RESOLUCIÓN DE PROBLEMAS CON EL PROGRAMA CAC 2D

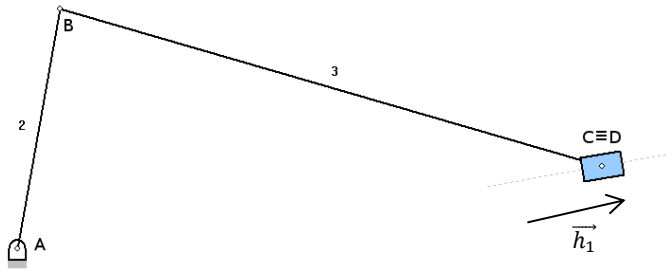
En la práctica, sólo es problema lo que la inteligencia puede resolver.

- Hermann Keyserling -

En este capítulo se pretende comprobar la funcionalidad del programa CAC 2D. Se resolverán usa serie de problemas de mecanismos planos usando el programa y mostrando el informe que este genera. Los cálculos pueden comprobarse de manera manual para cerciorar la veracidad de los resultados. Los mecanismos que aquí se resolverán son ejemplos típicos y comunes de máquinas que se diseñan y producen día a día.

6.1 Problema 1

Se resolverá un mecanismo sencillo de cuatro eslabones con un par prismático (biela-manivela típico).



$$\vec{h}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Coordenadas	X	Y
A	0	0
B	1	1
C	6	0
D	6	0

Condiciones iniciales	
$\vec{\omega}_{21}$	$2\ rad/s$
$\vec{\alpha}_{21}$	0

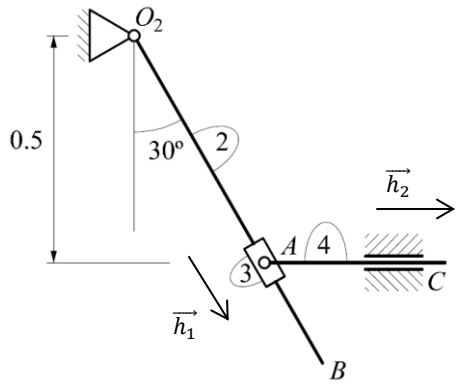
El informe de este mecanismo se adjunta en el anexo de este documento (Anexo B: Biela-Manivela). Los resultados obtenidos por el programa *CAC 2D* son los siguientes:

Velocidades y aceleraciones lineales		
	Velocidad (um/s)	Aceleración (um/s^2)
A	$(0,0)$	$(0,0)$
B	$(-2,2)$	$(-4,-4)$
C	$(-2.4,0)$	$(-4.032,0)$
D	$(-2.4,0)$	$(-4.032,0)$

Velocidades y aceleraciones angulares		
	Velocidad (rad/s)	Aceleración (rad/s^2)
Eslabón 1	0	0
Eslabón 2	2	0
Eslabón 3	-0.4	0.768
Eslabón 4	0	0

6.2 Problema 2

E mecanismo de la figura es un esquema de un mando de accionamiento del timón de un buque.



$$\vec{h}_1 = \begin{pmatrix} 0.5 \\ -0.8660 \end{pmatrix}$$

$$\vec{h}_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Condiciones iniciales
$\vec{V}_{41}^{C} = constante = \begin{pmatrix} 0.033 \\ 0 \end{pmatrix} um/s$

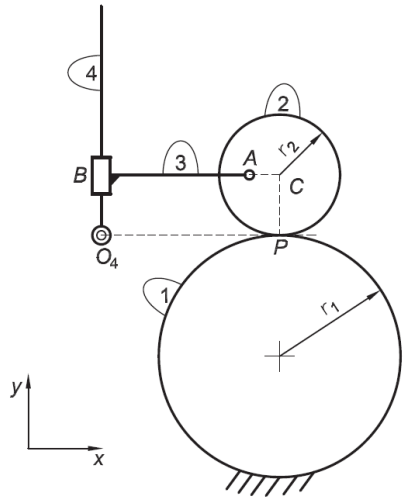
El informe de este mecanismo se adjunta en el anexo de este documento (Anexo C: Problema 2). Los resultados obtenidos por el programa *CAC 2D* son los siguientes:

Velocidades y aceleraciones lineales		
	Velocidad (um/s)	Aceleración (um/s^2)
A	$(0,0)$	$(0,0)$
B	$(0,0)$	$(-0.000707, 0.001225)$
C	$(0.033,0)$	$(0,0)$
D	$(0.033,0)$	$(0,0)$

Velocidades y aceleraciones angulares		
	Velocidad (rad/s)	Aceleración (rad/s^2)
Eslabón 1	0	0
Eslabón 2	0.0495	0
Eslabón 3	0.0495	0
Eslabón 4	0	0

6.3 Problema 3

En este problema el eslabón 2 rueda sin deslizar respecto al bastidor o cuerpo 1.



Datos	
$r_1 = 4\ um$	$\overline{CA} = 1\ um$
$r_2 = 2\ um$	$\overline{AB} = 3\ um$

Condiciones iniciales	
$\vec{\omega}_{21}$	$3\ rad/s$
$\vec{\alpha}_{21}$	0

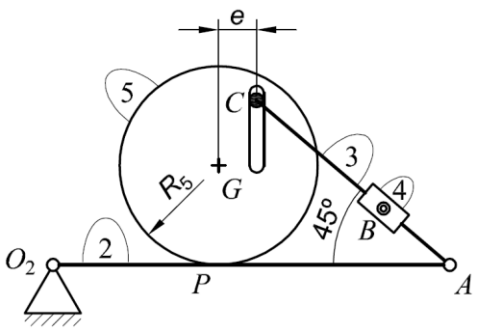
El informe de este mecanismo se adjunta en el anexo de este documento (Anexo D: Problema 3). Los resultados obtenidos por el programa *CAC 2D* son los siguientes:

Velocidades y aceleraciones lineales		
	Velocidad (um/s)	Aceleración (um/s^2)
A	$(0, 0)$	$(0, 0)$
B	$(-6, -3)$	$(-27, 0)$
C	$(-6, -12)$	$(0, 0)$
D	$(0, 0)$	$(0, 0)$

Velocidades y aceleraciones angulares		
	Velocidad (rad/s)	Aceleración (rad/s^2)
Eslabón 1	0	0
Eslabón 2	3	18
Eslabón 3	3	0
Eslabón 4	3	0

6.4 Problema 4

El siguiente problema se analizará por el interés de estudiar dos lazos que componen un mecanismo.



Datos	
$R_5 = 50\ um$	$\overline{O_2P} = 94.38\ um$
$e = 25\ um$	$\overline{O_2A} = 200\ um$
$\overline{CA} = 114\ um$	$\overline{AB} = 68\ um$

Condiciones iniciales			
$\vec{\omega}_{21}$	$1\ rad/s$	$\vec{\alpha}_{21}$	$1\ rad/s^2$

El mecanismo se puede dividir en dos lazos que son respectivamente:

- Lazo 1: eslabones → 1, 2, 3, 4, 1.
- Lazo 2: eslabones → 1, 3, 4, 5, 2, 1.

El informe de este mecanismo se adjunta en el anexo de este documento (Anexo E: Problema 4). Los resultados obtenidos por el programa *CAC 2D* son los siguientes:

Velocidades y aceleraciones lineales		
	Velocidad (um/s)	Aceleración (um/s^2)
O_2	$(0, 0)$	$(0, 0)$
A	$(0, 200)$	$(-200, 200)$
B	$(-100, 100)$	$(7.97258, -7.97258)$
C	$(0, 0)$	$(0, 0)$
P	$(48.0833, -57.5367)$	$(150.897, 201.535)$

Velocidades y aceleraciones angulares		
	Velocidad (rad/s)	Aceleración (rad/s^2)
Eslabón 1	0	0
Eslabón 2	1	1
Eslabón 3	2.0773	0
Eslabón 4	2.0773	0
Eslabón 5	1.67642	-0.871936

7 MEJORAS DE FUTURO Y CONCLUSIONES

En este capítulo se presentarán una serie de posibles mejoras de futuro a las que podría adaptarse el programa *CAC 2D*. Basándonos en el método utilizado y respetando toda la metodología impuesta, podemos sugerir ciertas mejoras del programa de cara al futuro. Si bien estas mejoras van destinadas a suplir ciertas carencias o limitaciones del programa que, sin suponer una disminución de la calidad del mismo, pueden ser fácilmente mejorables.

En cuanto a posibles mejoras podremos distinguir entre mejoras visuales, mejoras del método, mejoras del guardado, mejoras de la simulación, etc. Se intentará dar una explicación de cómo implementarlas y su uso.

Uno de los aspectos fundamentales del método es la restricción del cálculo a un solo lazo del mecanismo a estudio. Si bien esto no tiene que significar una desventaja ya que simplemente impondría al usuario estudiar por separado los lazos del mecanismo dado. Sí es recomendable que el método incluya una funcionalidad para estudiar todos los lazos a la vez dentro de un mismo mecanismo. El análisis de cada lazo individual sería el mismo que se realiza ahora pero se ahorraría al usuario estudiar los lazos por separado. Simplemente se podrían analizar todas las ecuaciones en conjunto de todos los lazos del mecanismo.

Los pares posibles que pueden contener los mecanismos que soporta *CAC 2D* son solo tres (par de revolución, par prismático y par de rodadura sin deslizamiento). Las ecuaciones de estos pares son conocidas y se han implementado al método para poder analizarlas de manera genérica. Los pares prismáticos deberán ser rectos para poder dar un resultado exitoso. Los pares prismáticos curvos no podrán ser analizados con este programa esto podría mejorarse en versiones futuras. Sería posible incluir las ecuaciones de distintos pares que actualmente no se contemplan tales como pares de rodadura con deslizamiento o de leva en los que el usuario defina completamente la geometría de la superficie. Estos pares son los que permitirían un deslizamiento entre los dos cuerpos en el punto de contacto.

Existen muchos otros pares de restricción que podrían incluirse pero su efecto en dos dimensiones no podría estudiarse. Podría ser otra mejora de futuro la adaptación del método para la cinemática 3D. Este paso tan importante (*CAC 3D*) implicaría la adaptación de las ecuaciones de los pares a 3D así como la adición de nuevos pares exclusivos de esta dimensión.

El proceso de simulación podría mejorarse en versiones futuras. Como ya se explicó en el capítulo 5 la simulación de esta versión está limitada solo a mecanismos con pares de rotación y pares prismáticos exclusivamente. Esta limitación se explicó en su momento pero sí que podrían mejorarse los subprogramas encargados de las representaciones del mecanismo así como las encargadas del cálculo de la siguiente posición del mismo. También se dijo que el método que se usa precisamente para el cálculo de la nueva posición tiene sus limitaciones y proporciona un error aceptable, todo esto podría mejorarse con técnicas mejores (la usada aquí considera cambios lineales y aceleraciones constantes) así como disminuciones en el grado de error

cometido por las mismas.

Temas referentes al proceso de guardado, cargado, interfaz gráfica o informe del archivo podrían mejorarse de cara a nuevas versiones. El proceso de guardado es sencillo y adecuado pero podría mejorarse en rendimiento con programas de encriptación de los datos o programas que compriman los datos guardados en un solo archivo. La interfaz utilizada es sencilla y plantea al usuario un entorno auto explicativo donde introducir los datos. Sin embargo se podría crear una interfaz quizás mejor, con funciones de ayuda e introducción de los datos mejores y más explicativas. El informe se genera en "*.pdf*" y con los datos del mecanismo debidamente ordenados. Esto podría mejorarse creando documentos quizás más elaborados.

- **Conclusiones**

Una vez finalizado el proyecto se puede afirmar que los objetivos se han cumplido exitosamente. Se ha estudiado la cinemática de mecanismos planos dándole un enfoque computacional con el objetivo de crear un programa capaz de resolver los problemas de velocidad y aceleración. Basándonos en las ecuaciones y la física ya existente (y altamente estudiada) se ha conseguido generalizar el procedimiento de resolución. Esto ha servido para ver cómo funcionan y qué papel desempeñan cada par cinemático en un mecanismo plano. Se ha estudiado también un proceso capaz de transformar las ecuaciones a un formato matricial lo cual ayudaba a la digitalización del método.

Se ha cumplido el objetivo de la creación de un programa en *Matlab* capaz de recoger, analizar y resolver la cinemática de mecanismos planos. Si bien el programa puede mejorarse y ampliarse como se recoge en los primeros párrafos de este capítulo, se trata de un programa funcional que cumple con su trabajo y que presenta funciones de apoyo y ayuda para mejorar la calidad de uso del mismo.

Puede concluirse añadiendo que se trata de un trabajo cerrado con bases teóricas ricas y extensas en las que se apoya el método pero que de cara a la computarización puede mejorarse con versiones más complejas y con más funcionalidades que añadir a esta ya útil herramienta de cálculo.

BIBLIOGRAFÍA

[1] RAFAEL CASTRO TRIGUERO, MANUEL HIDALGO MARTINEZ y JOSÉ ANTONIO SÁNCHEZ CASTILLEJO. *Analisis de mecanismos planos: (Ejercicios y resolución mediante Matlab)*. Córdoba: Servico de publicaciones, Universidad de Córdoba, 2006.

[2] SALVADOR CARDONA FOIX y DANIEL CLOS COSTA. *Teoria de màquines.* Ediciones UPC, 2001.

[3] MUNIR KHAMASHTA, LORENZO ALVAREZ y RAMÓN CAPDEVILA. *Problemas resueltos de cinemática de mecanismos planos.* Barcelona: Universitat Politécnica de Catalunya, 1986.

[4] FRANCISCO FERNÁNDEZ ZACARÍAS y ANTONIO ILLANA MARTOS. *Cinemática plana de mecanismos: Mecanismos, ejercicios resueltos.* Cádiz: Universidad de Cádiz, 2012.

[5] CARMELO VILLA PRADO. *Análisis dinámico de mecanismos planos mediante el método de transformaciones de velocidad.* Sevilla: Trabajo fin de grado, 2004.

[6] Apuntes de la asignatura de Teoría de Máquinas y Mecanismos. GITI, Escuela Técnica Superior de Ingeniería, Sevilla.

ANEXOS

ANEXO A: MANUAL DE USUARIO

Se presenta a continuación el manual de usuario. Este se ha diseñado penando en ayudar al usuario a la hora de introducir los datos en el programa *CAC 2D*. También se expresa en él cómo interpretar los datos, las comprobaciones iniciales y expone al usuario unas nociones básicas para el correcto uso del programa y sus funciones.

Escuela Técnica Superior de Ingeniería Sevilla

Manual de usuario

Computational Analysis Cinematic of Mechanical 2D

Manual de usuario

Índice

Introducción

Bienvenido al manual de usuario del programa CAC2D. En este pequeño documento le enseñaremos como utilizar fácilmente el programa y cómo interpretar sus resultados. Se expone a modo de ejemplo la resolución de un mecanismo sencillo con todos los tipos de pares que puede abarcar este programa (Pares de revolución, prismáticos y de rodadura sin deslizamiento).

Se indicarán una serie de pasos genéricos para poder implementar cualquier mecanismo. Cabe decir que el uso e interpretación de los datos de este programa está enfocado para personas con un nivel básico en la cinemática de mecanismos planos.

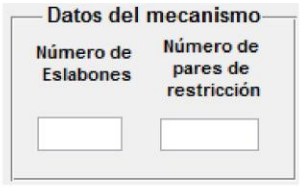
1. **Instalación del programa**
   Para facilitar la instalación de CAC2D al usuario se ha adjuntado el documento "Procedimiento de instalción.txt" donde se dice paso a paso cómo se debe instalar este programa.
   CAC2D está diseñado para ejecutarse en *Matlab 2012a* o versiones superiores como una script de código. Es por ello que se incluye el archivo "CAC_2D_Matlab" el cual podrá ejecutarse directamente en *Matlab.*
   No obstante CAC_2D también incluye una versión ejecutable para usuarios que no posean *Matlab*. Para ello será necesario instalar previamente el programa MCR (*MATLAB Compiler Runtime*) el cual hará posible el cálculo y las operaciones que se realizan dentro de CAC 2D sin disponer de *Matlab.* Esta versión, sin embargo, no posee licencias de creación de archivos ".pdf" o ".html" por lo que quedan suspendida la posibilidad de crear un informe del mecanismo. Las demás funcionalidades se mantienen.

2. **Inicio del programa**
   Para iniciar *CAC2D* simplemente ejecutaremos el archivo principal y seguiremos las instrucciones hasta llegar a la pantalla de introducción de datos.

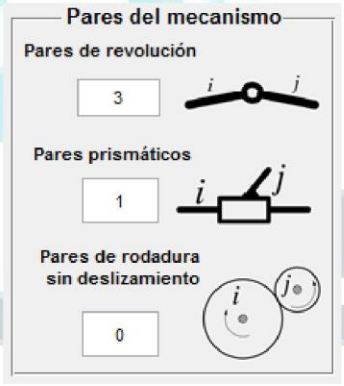
   Se trata de una interfaz interactiva con mensajes de ayuda y recomendaciones para facilitar al usuario la entrada de datos. Todos los datos excepto el campo "*Nombre del archivo*" deberá introducirse en formato numérico. Siempre deberá seguir las instrucciones de las posibles ventanas emergentes que podrán aparecer.
   Todo el programa usa variables adimensionales por lo que el usuario deberá introducir todos los datos en unidades métricas (*um*) y los datos referentes a dimensiones angulares en radianes (*rad*).

3. **Introducción de los datos básicos**
   Primeramente es recomendable darle un nombre al archivo donde guardaremos los datos del mecanismo. En el cuadro "*Nombre de archivo*" podremos introducir un nombre. Si no introducimos nada el archivo se guardará con el nombre "*Mecanismo*" por defecto.

Posteriormente, deberemos introducir los datos básicos del mecanismo. En cada cuadro de introducción de texto deberemos introducir el valor numérico que nos piden. Si el mecanismo no posee el ningún par prismático por ejemplo o cualquier otro tipo de par deberemos rellenar ese cuadro con un cero.



En el cuadro "*Pares del mecanismo*" deberemos introducir el número de pares de cada tipo que se encuentren en el mecanismo. Se ha incluido una imagen esquemática a la derecha de todos los cuadros de introducción de datos para poder identificar bien los pares.

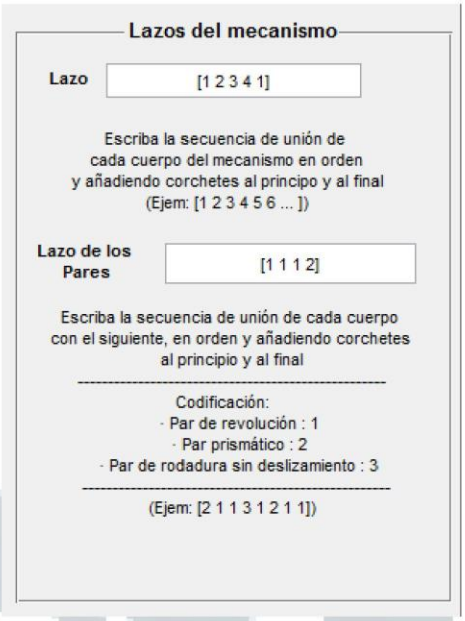


A la hora de introducir los vectores *Lazo* y *Lazo de los pares* deberemos seguir las instrucciones que aparecen junto a la ventana de introducción.
El vector *Lazo* se deberá introducir como una cadena de números entre corchetes y con un espacio entre cada número (sin espacios entre número y corchete). El lazo deberá empezar por el número 1 y continuar incrementando un número por cada eslabón que tenga el lazo a estudio. Si el lazo se trata de un lazo cerrado, este vector deberá empezar y acabar en 1, mientras que si es de lazo abierto podrá terminar en cualquier otro número distinto de 1.
El eslabón representado por el número 1 siempre deberá de ser el bastidor, es decir, este eslabón será el que no se mueve y sobre el que se referirán todas las velocidades y aceleraciones relativas. El programa impondrá esta condición de manera automática.
La introducción del valor de *Lazo de los pares* se hará igual, una cadena de números

entre corchetes y separados por un solo espacio (sin espacios entre número y corchete). En este caso se una codificación para cada tipo de par a estudio, siendo:

· Par de revolución → 1
· Par prismático → 2
· Par de rodadura sin deslizamiento → 3

No se podrá introducir ningún otro número que no sea cualquiera de estos tres. El orden de introducción es el natural, el mismo seguido en la introducción del vector *lazo.*



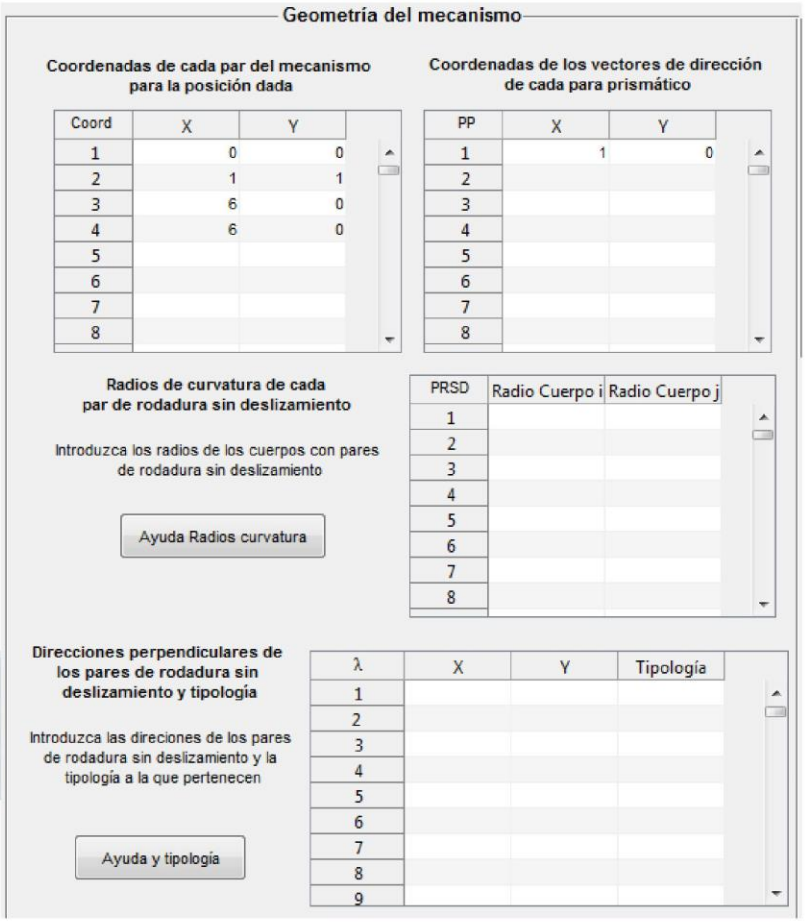
4. Introducción de la geometría

La geometría del mecanismo presenta un pilar fundamental a la hora del estudio del problema de velocidad y aceleración. La totalidad de la geometría del mecanismo debe ser conocida a priori.

De manera impositiva se usarán coordenadas cartesianas planas, siendo la posición del centro de referencia (Punto [0,0]) la que más interese al usuario. Se deberán introducir de manera obligatoria tantos puntos de coordenadas ([X,Y]) como números de pares tenga el lazo a estudio. El máximo de pares que se pueden introducir es 100, ya que este es el máximo de casillas expuestas en la interfaz del programa.

Si el lazo del mecanismo dispone de pares prismáticos se deberá introducir la dirección natural de este. Esta dirección se introducirá en términos [X,Y] y no será necesario la introducción normalizada de la misma (El mismo programa se encargará de normalizarla).

Si el lazo posee pares de rodadura sin deslizamiento entonces se deberán rellenar los cuadros de *Radios* y *Direcciones perpendiculares de los pares de rodadura sin deslizamiento*. En el cuadro *Radios* deberá introducir los dos radios de curvatura para la posición dada de cada eslabón que componen el par de rodadura sin deslizamiento. Si se trata de un eslabón plano (Radio de curvatura infinito) no es necesario que



introduzca es término. El botón "*Ayuda radios curvatura*" proporciona información visual de cómo debe rellenarse esta tabla.

Por último el cuadro de *Direcciones perpendiculares de los pares de rodadura sin deslizamiento y tipología* es donde se debe introducir la dirección perpendicular a cada par de este tipo y su tipología. Este último término, la tipología sigue una codificación del 1 al 4 dependiendo de qué tipo de cuerpos intervienen en ese par. Dicha codificación se muestra a continuación:

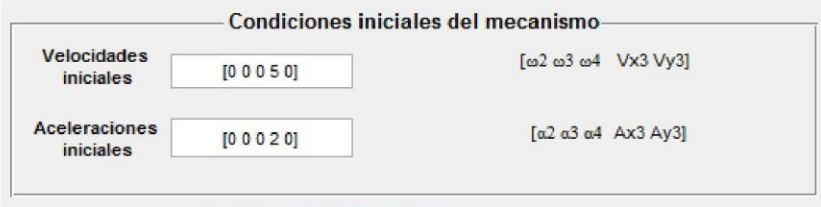
Tipo	Cóncavo - Convexo	Convexo - Cóncavo	Rueda - Plano	Plano - Rueda
Código	1	2	3	4

No se podrá introducir, en la columna *Tipología* ningún otro número que no esté dentro de esta codificación.

5. Introducción de las velocidades y aceleraciones iniciales

Una vez introducidos los datos básicos, los cuadros de introducción de las velocidades y las aceleraciones iniciales se desbloquearán. Al lado derecho de cada cuadro de texto editable aparecerán unas recomendaciones para facilitar al usuario la introducción de los datos.

Los datos a introducir se efectuaran del mismo modo al que se hacía en el cuadro *lazo* o *lazo de los pares*, se introducirá una cadena de números entre corchetes separados por espacios (sin espacios entre número y corchete).



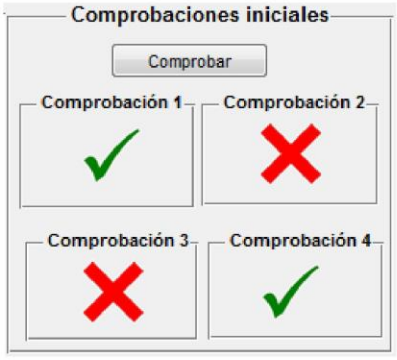
IMPORTANTE

Estos vectores de velocidades y aceleraciones iniciales deberán contener tantos términos distintos de cero como número de grados de libertad tenga el lazo del mecanismo. De este modo los vectores a introducir tendrán un tamaño igual al recomendado en la ayuda de la parte derecha y se compondrán de los términos de las velocidades iniciales conocidas. Las velocidades iniciales no conocidas y que aparezcan en el vector recomendación se deberán introducir como un valor igual a cero.

Se expone a modo de ejemplo una introducción de términos en estos cuadros para facilitar la comprensión.

6. Comprobaciones previas

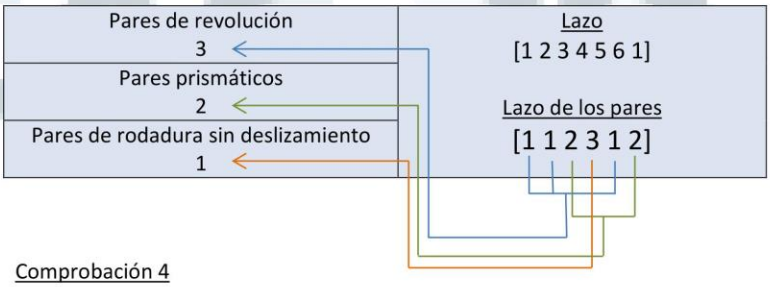
Estos cuatro cuadros facilitan al usuario una serie de comprobaciones previas a la resolución del lazo del mecanismo para evitar fallos en la introducción de los datos. Se trata de comprobaciones sencillas del sentido lógico de la introducción de los datos que harán que los resultados obtenidos posean sentido y el cálculo de los mismos no se vea afectado por errores triviales. De este modo el usuario puede ver de manera sencilla (con un tick verde o una cruz roja) si ha introducido de manera lógica los datos. Se realizan 4 comprobaciones básicas que se pasan a describir a continuación:



- Comprobación 1
  Esta comprobación repasa si se ha introducido de manera correcta el *lazo*, *lazo de los pares* y el número de restricciones del mecanismo.

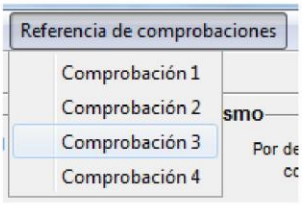
- Comprobación 2
  Aquí se comprobará si el número total de pares introducidos (pares de revolución + pares prismáticos + pares de rodadura sin deslizamiento) coincide con el número de restricciones impuesto.

- Comprobación 3
  El número de cada par introducido de manera separada debe coincidir con el número de pares de cada tipo introducidos en el *Lazo de los pares* conforme a la codificación establecida (1: Pares de revolución ; 2: Pares prismáticos ; 3: Pares de rodadura sin deslizamiento). De esta forma y a modo de ejemplo:



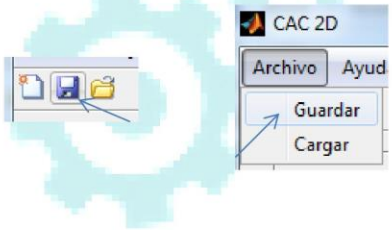
- Comprobación 4
  Esta última comprobación estudia si se introdujeron bien los vectores de las velocidades y aceleraciones iniciales en cuanto a tamaño. Para ello comprueba los datos básicos del mecanismo y comprueba el tamaño de estos vectores.

Todas las descripciones de cada comprobación también se podrá observar en la pestaña "*Referencia de comprobaciones*".

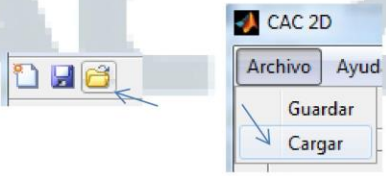


7. Guardado y cargado de un mecanismo

La geometría y datos básicos de los mecanismos estudiados se podrán guardar para, posteriormente volverlos a recuperar. En este caso *CAC2D* incluye un método de guardado provisto de una carpeta llamada "Archivos guardados" donde se guardarán los mecanismos que se hayan escrito en el programa si así lo desea el usuario.
El método de guardado es sencillo. Se introducen los datos del mecanismo y se le dota de un nombre (si no se introduce ningún nombre el archivo se guardará por defecto con el nombre "*Mecanismo*"). Posteriormente se pulsa el botón de guardar y el archivo quedará guardado en la carpeta anteriormente nombrada. Si se guarda un archivo con un nombre igual a otro archivo existente en esta carpeta de guardado el antigua archivo será reemplazado por el nuevo.



Si se desea cargar un archivo de mecanismo guardado previamente se deberá introducir el nombre completo que se le dio en el cuadro *Nombre de archivo* y pulsar el botón de cargar.



Si no existe ningún archivo guardado con ese nombre, o este es erróneo el programa avisará con una ventana emergente. El archivo cargado no podrá ser modificado salvo en los datos ya introducidos cambiando valores de la geometría o de las características básicas.
Para volver a obtener los resultados de un mecanismo ya guardado se deberá cargar de nuevo y resolver dicho mecanismo.
Se recomienda que para la introducción de un nuevo mecanismo se pulse previamente el botón "Nuevo" de esta forma el programa se reseteara y dejara todos los cuadros vacíos a la espera de la introducción de nuevos datos.

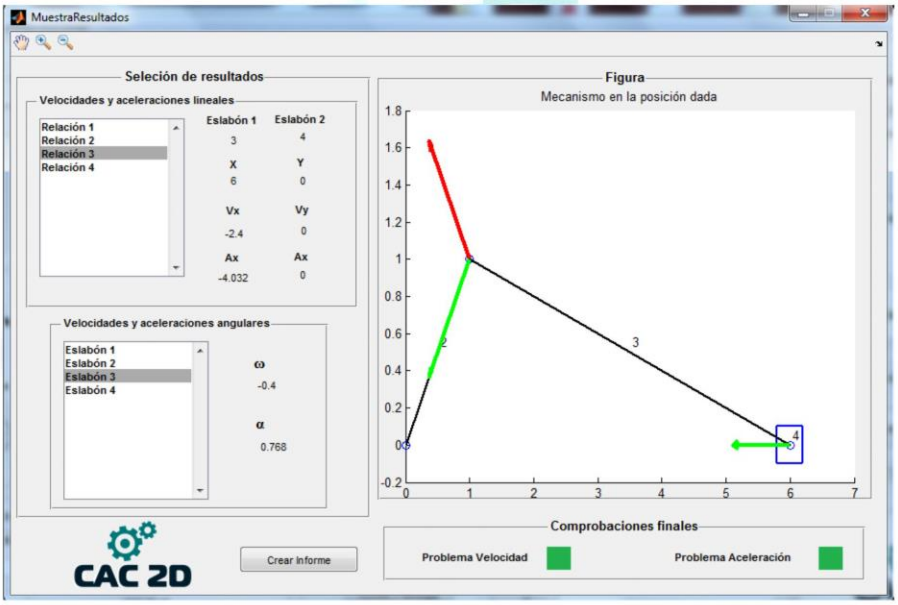
8. Resolución, simulación e interpretación de los datos

Para la resolución del mecanismo se deberá pulsar el botón "*Resolver*" pero previamente deberemos seleccionar entre dos opciones, resolver solo el problema de velocidades o resolver el problema de velocidades y el de aceleraciones. Por defecto, al inicio del programa, vendrá impuesta la resolución del problema de velocidades pero el usuario podrá cambiarlo a gusto en el momento que quiera.

Una vez pulsado el botón "*Resolver*" aparecerán ventanas emergentes si existe cualquier fallo o falta de información en la introducción de los datos. Si todo esta correcto se abrirá una nueva venta que mostrará el resultado de la resolución y un esquema del mecanismo.

Haciendo clip en las distintas relaciones se nos mostrara qué cuerpos unen, cuáles son sus coordenadas y sus velocidades y aceleraciones lineales correspondientes (obviamente si solo se ha seleccionado la resolución del método de velocidades las aceleraciones, ni lineales ni angulares aparecerán).

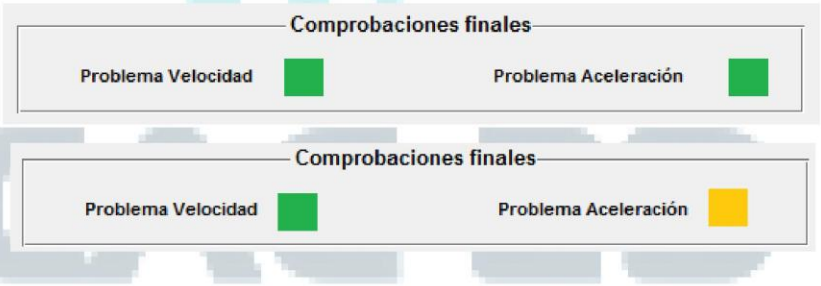
Más abajo se nos mostrará una lista completa de los eslabones que forman el lazo del mecanismo y sus velocidades y aceleraciones angulares.



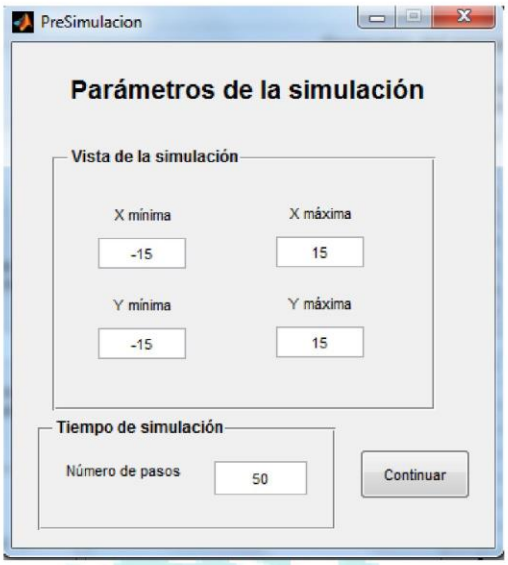
A la derecha se mostrará un esquema del mecanismo en un plano cartesiano con las coordenadas impuestas por el usuario. *La imagen que aparecería está realizada en unas condiciones gráficas automáticas para el programa que hacen que se pueda ver con claridad todo el mecanismo. Esto puede provocar el efecto de distorsión de los ángulos del mismo.*

Los pares se representaran en un tono azul, diferenciándose en su forma. Los pares de revolución se representarán como pequeños círculos, los pares prismáticos como cuadrados orientados en la dirección natural del par y los pares de rodadura sin

deslizamiento se representaran como ruedas en contacto cuyo radio será el radio de curvatura de cada eslabón que compone el par.
Las velocidades y aceleraciones de cada par, aparte de representarse numéricamente en la parte izquierda anteriormente mencionada se harán una representación esquemática de su dirección y sentido en el dibujo del mecanismo. Las velocidades de representan como vectores rojos y las aceleraciones como vectores verdes. El módulo representado en el dibujo no es el real, solo un esquema de cómo sería (se hace así para hacer más visual la representación). Los vectores aceleración y velocidad podrían llegar a pisarse en la representación (Por ello se recomienda un estudio separado de ambos problemas).
Los eslabones se representarán con una barra negra entre cada par en el que se establece dicho eslabón. Junto a cada eslabón aparece su número de correspondiente.
En la parte inferior derecha se ha incluido unas comprobaciones finales de los datos calculados. Se realizan dos comprobaciones, una del problema de velocidad y otra del problema de aceleración. Esquemáticamente si las comprobaciones están correctas aparecerá un cuadro color verde junto a estas. Si por el contrario, el sistema de comprobación detecta algún fallo, automáticamente aparecerá un cuadro color amarillo. Esto indica que a pesar de encontrar una solución al problema puede que los métodos numéricos empleados no proporcionen una exactitud aceptable de las soluciones. En estos casos el mismo programa dictamina si existen posibles errores en las soluciones de cara a la veracidad del mismo.



El programa también cuenta con un software capaz de simular el movimiento del mecanismo. De este modo solo tendremos que introducir los datos de nuestro mecanismo y marcar la casilla "*Simulación*" junto al botón *Resolver* y pulsar este último. Automáticamente se nos abrirá una ventana para introducir los valores máximo y mínimos de los ejes de accisas en la representación de la simulación. También podremos introducir el número de pasos en los que se desarrollará la simulación de movimiento.



Debido a las características del método de resolución el programa solo podrá representar simulaciones de mecanismos que no posean pares de rodadura sin deslizamiento. La simulación se presentará como una herramienta útil para ver cómo es, de manera básica, el movimiento del mecanismo. Cabe decir que los valores obtenidos solo serán representativos si las velocidades iniciales introducidas se mantienen constantes a lo largo del tiempo.

9. Creación del informe

En la ventana "*Muestra resultados*" podremos pulsar el botón *Crear informe* y automáticamente CAC2D generará un archivo "*.pdf*" con todos los datos de nuestro mecanismo. También se incluirán los resultados de resolver los problemas de velocidad y aceleración (el que se halla elegido previamente), un esquema gráfico del mecanismo en la posición dada y un esquema de la distribución de velocidades y aceleraciones.

Nota importante: Existen dos formas de ejecutar el programa CAC2D. La primera y más sencilla va dirigida a los usuarios que no posean el software *Matlab 2012a* o versiones superiores. Para que CAC2D pueda ejecutarse se basa en la instalación previa de *MATLAB Compiler Runtime* (MCR), una aplicación de *Matlab* que permite ejecutar los cálculos y las operaciones necesarias para el correcto desarrollo del programa. Esta forma de operar presenta limitaciones que no permiten la creación de archivos de extensión ".pdf" o ".html". Si el usuario no posee *Matlab* y usa MCR no podrá crearse un informe del mecanismo estudiado.
Otra forma de ejecutar CAC2D es directamente corriendo el script en *Matlab.* Si el usuario posee *Matlab* en versión 2012a o superior podrá ejecutar el código del

programa directamente sin tener que usar MCR. Esto permitirá tener todas las licencias para el uso de todas las opciones que ofrece CAC2D. La ejecución del código en *Matlab* se ha facilitado al usuario con la creación del archivo "CAC_2D_Matlab.m". El usuario tan solo tendrá que abrir *Matlab* y ejecutar en el este archivo y podrá disfrutar de todas las opciones de CAC2D incluida la creación de informes.

10. Dudas, reclamaciones y sugerencias

Para resolver cualquier duda, reclamación sobre el programa o solicitar algunas sugerencias para futuras mejoras diríjase a la dirección de correo electrónico: cac2dasistencia@gmail.com

ANEXO B: BIELA-MANIVELA

Table of Contents

Informe del mecanismo

Informe CAC 2D

Nombre del mecanismo

biela-manivela *02-Aug-2016*

Datos del mecanismo

Datos básicos

Número de eslabones: 4 ; Número de pares de restricción: 4

Datos de los pares

Pares de revolución: 3 ; Pares prismáticos: 1

Lazo del mecanismo

{ 1 2 3 4 1 }

Lazo de los pares del mecanismo

{ 1 1 1 2 }

---------- Codificación de los pares ----------

Par de revolución: 1 ; Par prismático: 2
Par de rodadura sin deslizamiento: 3

```
--------------------------------------------------
```

Geometría del mecanismo

Coordenadas de los pares [X,Y]

```
Par 1 : [ 0 ; 0 ]
Par 2 : [ 1 ; 1 ]
Par 3 : [ 6 ; 0 ]
Par 4 : [ 6 ; 0 ]
```

Direcciones de los pares prismáticos

```
Par prismático 1 :
   Eslabón 4  y  Eslabón 1
Dirección del par: [ 1 ; 0 ]
```

Direciones de los pares de rodadura sin deslizamiento

```
<<El mecanismo no posee pares de rodadura sin deslizamiento>>
```

Radios de curvatura de los pares de rodadura sin deslizamiento

```
<<El mecanismo no posee pares de rodadura sin deslizamiento>>
```

Tipología de los pares de rodadura sin deslizamiento

```
<<El mecanismo no posee pares de rodadura sin deslizamiento>>
```

Velocidades y acelecaciones iniciales

Velocidades iniciales

```
[w2 w3 w4] : [ 2 0 0 ] um/s
```

Acelecaciones iniciales

```
[a2 a3 a4] : [ 0 0 0 ] um/s^2
```

Resultados

Velocidades y aceleraciones lineales

```
Relación 1 :
    Eslabón 1  y Eslabón 2
    Coordenadas : [ 0 ; 0 ]
    Velocidad lineal : [ 0 ; 0 ] um/s
    Acelecarión lineal : [ 0 ; 0 ] um/s^2

-----------------
Relación 2 :
    Eslabón 2  y Eslabón 3
    Coordenadas : [ 1 ; 1 ]
    Velocidad lineal : [ -2 ; 2 ] um/s
    Acelecarión lineal : [ -4 ; -4 ] um/s^2

-----------------
Relación 3 :
    Eslabón 3  y Eslabón 4
    Coordenadas : [ 6 ; 0 ]
    Velocidad lineal : [ -2.400000e+00 ; 0 ] um/s
    Acelecarión lineal : [ -4.032000e+00 ; 0 ] um/s^2

-----------------
Relación 4 :
    Eslabón 4  y Eslabón 1
    Coordenadas : [ 6 ; 0 ]
    Velocidad lineal : [ -2.400000e+00 ; 0 ] um/s
    Acelecarión lineal : [ -4.032000e+00 ; 0 ] um/s^2
```

Velocidades y aceleraciones angulares

```
Eslabón 1 :
    Velocidad angular : 0 rad/s
    Acelecarión angular : 0 rad/s^2

-----------------
Eslabón 2 :
    Velocidad angular : 2 rad/s
    Acelecarión angular : 0 rad/s^2

-----------------
Eslabón 3 :
    Velocidad angular : -4.000000e-01 rad/s
    Acelecarión angular : 7.680000e-01 rad/s^2

-----------------
Eslabón 4 :
    Velocidad angular : 0 rad/s
    Acelecarión angular : 0 rad/s^2
```

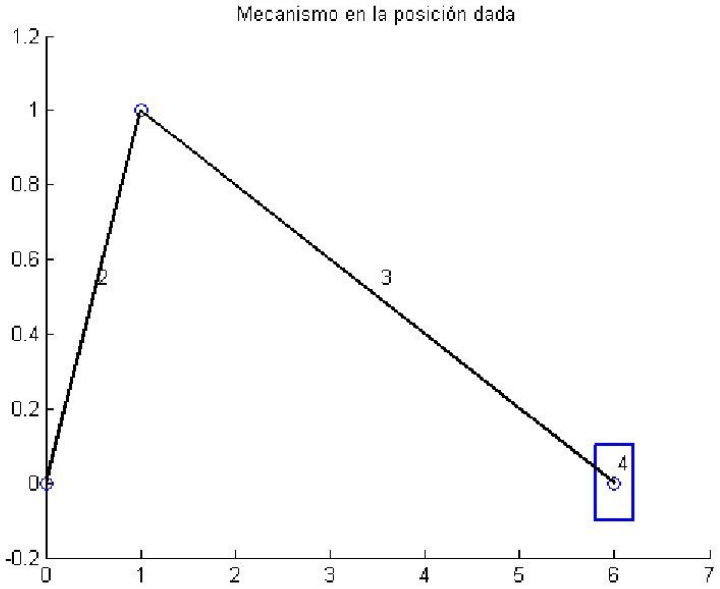
Comprobaciones de los resultados

```
CAC 2D comprueba los resultados obtenidos mediante
el método de cálculo con un error inferior a 1.000000e-05.
Si se detectase algún problema en los valores obtenidos
se comunicará un posible fallo.
```

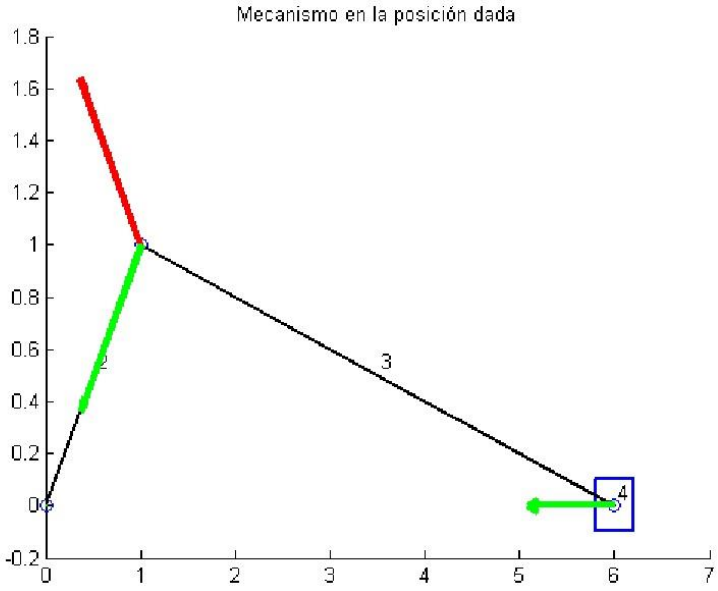
Comprobación del problema de velocidad

```
Problema de velocidad correcto
   con error 0

Problema de aceleración correcto
   con error 0
```

Esquema gráfico del mecanismo



Distribución de velocidades y aceleraciones



Published with MATLAB® 7.14

ANEXO C: PROBLEMA 2

Table of Contents

Informe del mecanismo

Informe CAC 2D

Nombre del mecanismo

Problema2 *02-Aug-2016*

Datos del mecanismo

Datos básicos

Número de eslabones: 4 ; Número de pares de restricción: 4

Datos de los pares

Pares de revolución: 2 ; Pares prismáticos: 2

Lazo del mecanismo

{ 1 2 3 4 1 }

Lazo de los pares del mecanismo

{ 1 2 1 2 }

---------- Codificación de los pares ----------

```
   Par de revolución: 1   ;  Par prismático: 2
      Par de rodadura sin deslizamiento: 3

-------------------------------------------------
```

Geometría del mecanismo

Coordenadas de los pares [X,Y]

```
Par 1 : [ 0 ; 0 ]
Par 2 : [ 2.886751e-01 ; -5.000000e-01 ]
Par 3 : [ 2.886751e-01 ; -5.000000e-01 ]
Par 4 : [ 9.000000e-01 ; -5.000000e-01 ]
```

Direcciones de los pares prismáticos

```
Par prismático 1 :
   Eslabón 2  y  Eslabón 3
Dirección del par: [ 5.000000e-01 ; -8.660254e-01 ]
-----------------
Par prismático 2 :
   Eslabón 4  y  Eslabón 1
Dirección del par: [ 1 ; 0 ]
```

Direciones de los pares de rodadura sin deslizamiento

```
<<El mecanismo no posee pares de rodadura sin deslizamiento>>
```

Radios de curvatura de los pares de rodadura sin deslizamiento

```
<<El mecanismo no posee pares de rodadura sin deslizamiento>>
```

Tipología de los pares de rodadura sin deslizamiento

```
<<El mecanismo no posee pares de rodadura sin deslizamiento>>
```

Velocidades y acelecaciones iniciales

Velocidades iniciales

```
[w2 w3 w4 Vx3 Vy3 Vx5 Vy5] : [ 0 0 0 3.300000e-02 0 ] um/s
```

Acelecaciones iniciales

```
[a2 a3 a4 Ax3 Ay3 Ax5 Ay5] : [ 0 0 0 0 0 0 ] um/s^2
```

Resultados

Velocidades y aceleraciones lineales

```
Relación 1 :
    Eslabón 1  y Eslabón 2
    Coordenadas : [ 0 ; 0 ]
    Velocidad lineal : [ 0 ; 0 ] um/s
    Acelecarión lineal : [ 0 ; 0 ] um/s^2

-----------------
Relación 2 :
    Eslabón 2  y Eslabón 3
    Coordenadas : [ 2.886751e-01 ; -5.000000e-01 ]
    Velocidad lineal : [ 0 ; 0 ] um/s
    Acelecarión lineal : [ -7.073262e-04 ; 1.225125e-03 ] um/s^2

-----------------
Relación 3 :
    Eslabón 3  y Eslabón 4
    Coordenadas : [ 2.886751e-01 ; -5.000000e-01 ]
    Velocidad lineal : [ 3.300000e-02 ; -1.551584e-18 ] um/s
    Acelecarión lineal : [ 8.785689e-11 ; -5.780100e-27 ] um/s^2

-----------------
Relación 4 :
    Eslabón 4  y Eslabón 1
    Coordenadas : [ 9.000000e-01 ; -5.000000e-01 ]
    Velocidad lineal : [ 3.300000e-02 ; -1.551584e-18 ] um/s
    Acelecarión lineal : [ 8.785689e-11 ; -5.780100e-27 ] um/s^2
```

Velocidades y aceleraciones angulares

```
Eslabón 1 :
    Velocidad angular : 0 rad/s
    Acelecarión angular : 0 rad/s^2

-----------------
Eslabón 2 :
    Velocidad angular : 4.950000e-02 rad/s
    Acelecarión angular : 0 rad/s^2

-----------------
Eslabón 3 :
    Velocidad angular : 4.950000e-02 rad/s
    Acelecarión angular : 0 rad/s^2

-----------------
```

```
Eslabón 4 :
    Velocidad angular : 0 rad/s
    Acelecarión angular : 0 rad/s^2
```

Comprobaciones de los resultados

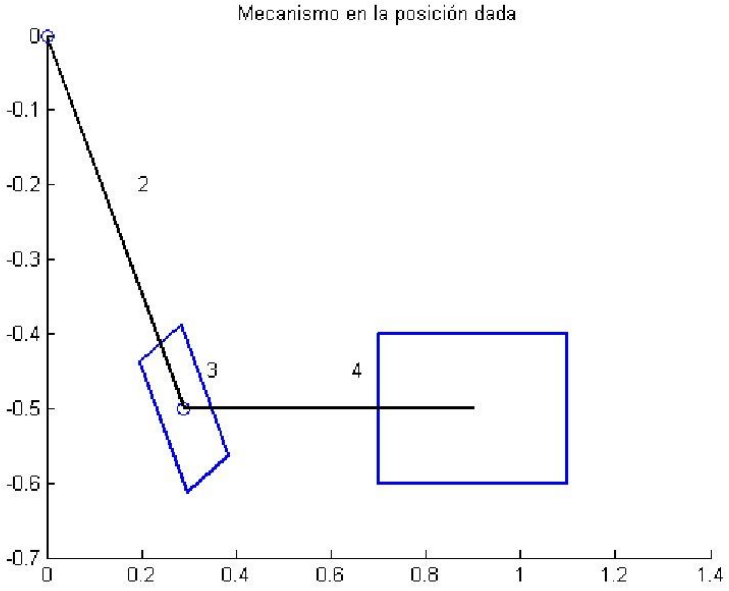
```
CAC 2D comprueba los resultados obtenidos mediante
el método de cálculo con un error inferior a 1.000000e-05.
Si se detectase algún problema en los valores obtenidos
se comunicará un posible fallo.
```

Comprobación del problema de velocidad

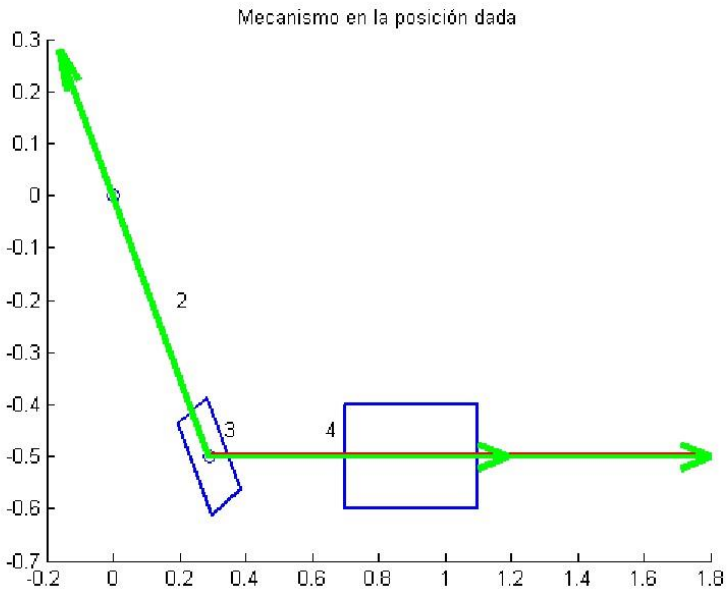
```
Problema de velociad correcto
   con un errror del 10^-1.729921e+01
```

```
Problema de aceleración correcto
   con un errror del 10^-1.011869e+01
```

Esquema gráfico del mecanismo



Distribución de velocidades y aceleraciones



Published with MATLAB® 7.14

ANEXO D: PROBLEMA 3

Table of Contents

Informe del mecanismo

Nombre del mecanismo

Problema3 *02-Aug-2016*

Datos del mecanismo

Datos básicos

Número de eslabones: 4 ; Número de pares de restricción: 4

Datos de los pares

Pares de revolución: 2 ; Pares prismáticos: 1

Lazo del mecanismo

{ 1 2 3 4 1 }

Lazo de los pares del mecanismo

{ 3 1 2 1 }

---------- Codificación de los pares ----------

Par de revolución: 1 ; Par prismático: 2

```
Par de rodadura sin deslizamiento: 3

--------------------------------------------------
```

Geometría del mecanismo

Coordenadas de los pares [X,Y]

```
Par 1 : [ 0 ; 4 ]
Par 2 : [ -1 ; 6 ]
Par 3 : [ -4 ; 6 ]
Par 4 : [ -4 ; 4 ]
```

Direcciones de los pares prismáticos

```
Par prismático 1 :
   Eslabón 3  y  Eslabón 4
Dirección del par: [ 0 ; 1 ]
```

Direciones de los pares de rodadura sin deslizamiento

```
Par de rodadura sin deslizamiento 1 :
   Eslabón 1  y  Eslabón 2
Dirección del par: [ 0 ; -1 ]
-----------------
```

Radios de curvatura de los pares de rodadura sin deslizamiento

```
Par de rodadura sin deslizamiento 1 :
    Radio de curvatura eslabón 1 : 4     ;     Radio curvatura eslabón 2 :
-----------------
```

Tipología de los pares de rodadura sin deslizamiento

```
Par de rodadura sin deslizamiento 1 :
   Eslabón 1  y  Eslabón 2
   Tipología 1 : Cóncavo - Convexo
-----------------
```

Velocidades y acelecaciones iniciales

Velocidades iniciales

```
[w2 w3 w4] : [ 3 0 0 ] um/s
```

Acelecaciones iniciales

```
[a2 a3 a4] : [ 0 0 0 ] um/s^2
```

Resultados

Velocidades y aceleraciones lineales

```
Relación 1 :
    Eslabón 1  y Eslabón 2
    Coordenadas : [ 0 ; 4 ]
    Velocidad lineal : [ 0 ; 0 ] um/s
    Acelecarión lineal : [ 0 ; 0 ] um/s^2

-----------------
Relación 2 :
    Eslabón 2  y Eslabón 3
    Coordenadas : [ -1 ; 6 ]
    Velocidad lineal : [ -6 ; -3 ] um/s
    Acelecarión lineal : [ -27 ; 0 ] um/s^2

-----------------
Relación 3 :
    Eslabón 3  y Eslabón 4
    Coordenadas : [ -4 ; 6 ]
    Velocidad lineal : [ -6 ; -12 ] um/s
    Acelecarión lineal : [ 0 ; 0 ] um/s^2

-----------------
Relación 4 :
    Eslabón 4  y Eslabón 1
    Coordenadas : [ -4 ; 4 ]
    Velocidad lineal : [ 0 ; 0 ] um/s
    Acelecarión lineal : [ 0 ; 0 ] um/s^2
```

Velocidades y aceleraciones angulares

```
Eslabón 1 :
    Velocidad angular : 0 rad/s
    Acelecarión angular : 0 rad/s^2

-----------------
Eslabón 2 :
    Velocidad angular : 3 rad/s
    Acelecarión angular : 18 rad/s^2

-----------------
Eslabón 3 :
    Velocidad angular : 3 rad/s
    Acelecarión angular : 0 rad/s^2
```

```
-----------------
Eslabón 4 :
    Velocidad angular : 3 rad/s
    Acelecarión angular : 0 rad/s^2
```

Comprobaciones de los resultados

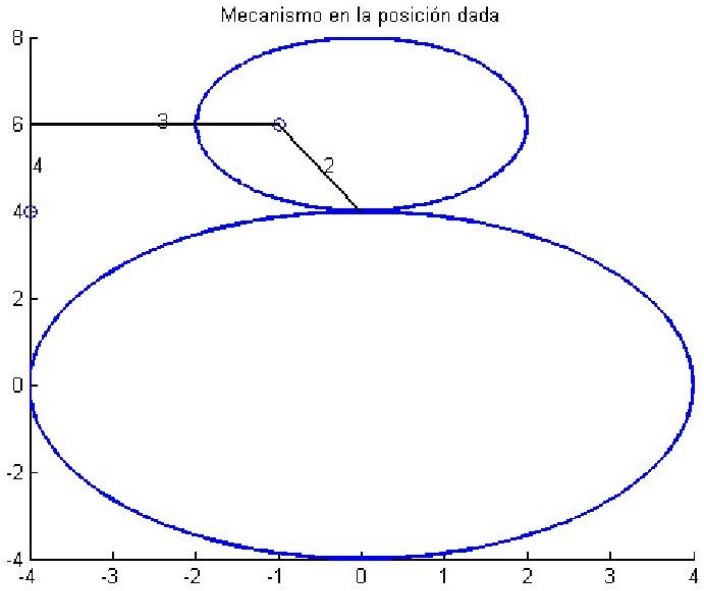
```
CAC 2D comprueba los resultados obtenidos mediante
el método de cálculo con un error inferior a 1.000000e-05.
Si se detectase algún problema en los valores obtenidos
se comunicará un posible fallo.
```

Comprobación del problema de velocidad

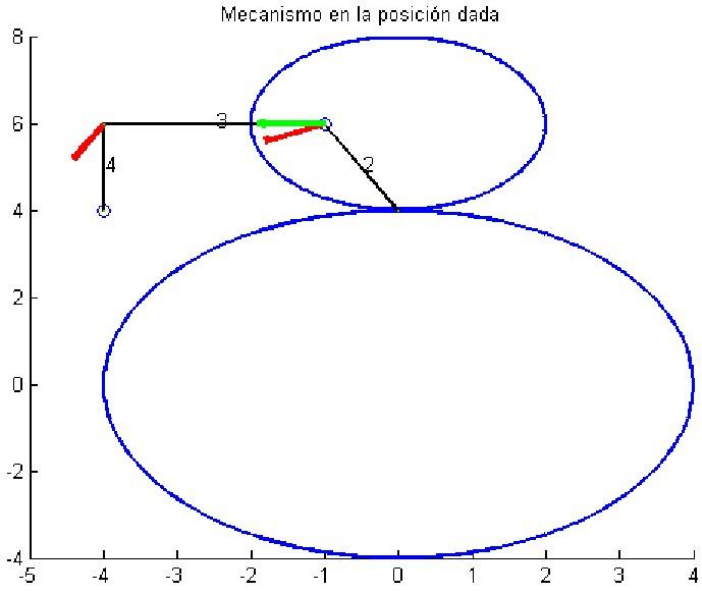
```
Problema de velocidad correcto
   con error 0
```

```
Problema de aceleración correcto
   con error 0
```

Esquema gráfico del mecanismo



Distribución de velocidades y aceleraciones



Published with MATLAB® 7.14

ANEXO E: PROBLEMA 4

Table of Contents

Informe del mecanismo

Informe CAC 2D

Nombre del mecanismo

Problema4a *02-Aug-2016*

Datos del mecanismo

Datos básicos

Número de eslabones: 4 ; Número de pares de restricción: 4

Datos de los pares

Pares de revolución: 3 ; Pares prismáticos: 1

Lazo del mecanismo

{ 1 2 3 4 1 }

Lazo de los pares del mecanismo

{ 1 1 2 1 }

---------- Codificación de los pares ----------

Par de revolución: 1 ; Par prismático: 2
Par de rodadura sin deslizamiento: 3

```
--------------------------------------------------
```

Geometría del mecanismo

Coordenadas de los pares [X,Y]

```
Par 1 : [ 0 ; 0 ]
Par 2 : [ 200 ; 0 ]
Par 3 : [ 1.519167e+02 ; 4.808326e+01 ]
Par 4 : [ 1.519167e+02 ; 4.808326e+01 ]
```

Direcciones de los pares prismáticos

```
Par prismático 1 :
   Eslabón 3  y  Eslabón 4
Dirección del par: [ -7.071068e-01 ; 7.071068e-01 ]
```

Direciones de los pares de rodadura sin deslizamiento

```
<<El mecanismo no posee pares de rodadura sin deslizamiento>>
```

Radios de curvatura de los pares de rodadura sin deslizamiento

```
<<El mecanismo no posee pares de rodadura sin deslizamiento>>
```

Tipología de los pares de rodadura sin deslizamiento

```
<<El mecanismo no posee pares de rodadura sin deslizamiento>>
```

Velocidades y acelecaciones iniciales

Velocidades iniciales

```
[w2 w3 w4] : [ 1 0 0 ] um/s
```

Acelecaciones iniciales

```
[a2 a3 a4] : [ 1 0 0 ] um/s^2
```

Resultados

Velocidades y aceleraciones lineales

```
Relación 1 :
    Eslabón 1  y Eslabón 2
    Coordenadas : [ 0 ; 0 ]
    Velocidad lineal : [ 0 ; 0 ] um/s
    Acelecarión lineal : [ 0 ; 0 ] um/s^2

-----------------
Relación 2 :
    Eslabón 2  y Eslabón 3
    Coordenadas : [ 200 ; 0 ]
    Velocidad lineal : [ 0 ; 200 ] um/s
    Acelecarión lineal : [ -200 ; 200 ] um/s^2

-----------------
Relación 3 :
    Eslabón 3  y Eslabón 4
    Coordenadas : [ 1.519167e+02 ; 4.808326e+01 ]
    Velocidad lineal : [ -1.000000e+02 ; 1.000000e+02 ] um/s
    Acelecarión lineal : [ 7.972583e+00 ; -7.972583e+00 ] um/s^2

-----------------
Relación 4 :
    Eslabón 4  y Eslabón 1
    Coordenadas : [ 1.519167e+02 ; 4.808326e+01 ]
    Velocidad lineal : [ 0 ; 0 ] um/s
    Acelecarión lineal : [ 0 ; 0 ] um/s^2
```

Velocidades y aceleraciones angulares

```
Eslabón 1 :
    Velocidad angular : 0 rad/s
    Acelecarión angular : 0 rad/s^2

-----------------
Eslabón 2 :
    Velocidad angular : 1 rad/s
    Acelecarión angular : 1 rad/s^2

-----------------
Eslabón 3 :
    Velocidad angular : 2.079726e+00 rad/s
    Acelecarión angular : -8.995358e-10 rad/s^2

-----------------
Eslabón 4 :
    Velocidad angular : 2.079726e+00 rad/s
    Acelecarión angular : -8.995358e-10 rad/s^2
```

Comprobaciones de los resultados

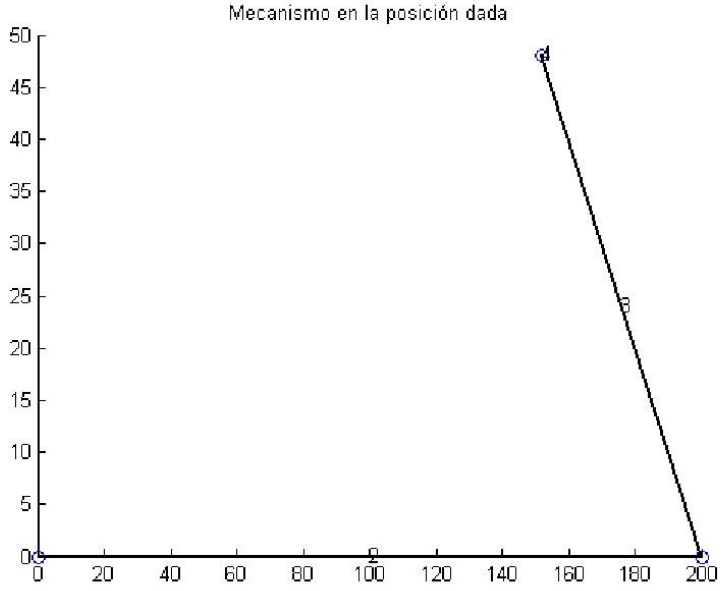
```
CAC 2D comprueba los resultados obtenidos mediante
el método de cálculo con un error inferior a 1.000000e-05.
Si se detectase algún problema en los valores obtenidos
se comunicará un posible fallo.
```

Comprobación del problema de velocidad

```
Problema de velocidad correcto
   con error 0
```

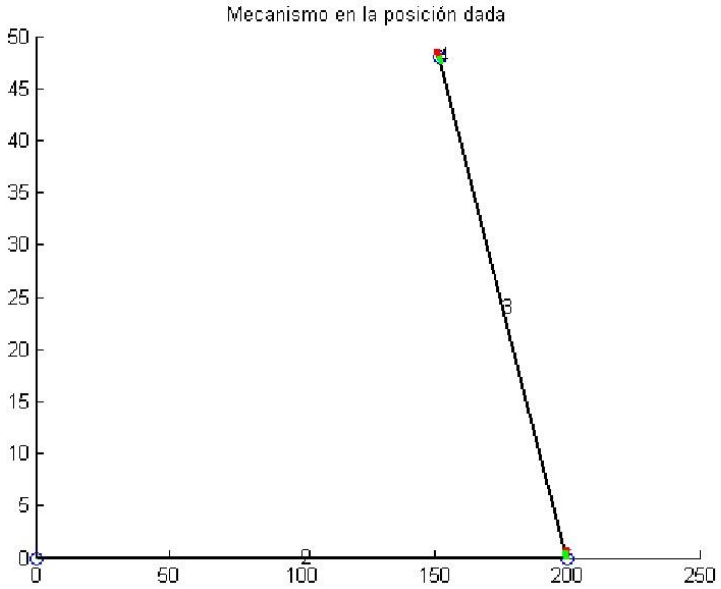
```
Problema de aceleración correcto
   con error 0
```

Esquema gráfico del mecanismo



4

Distribución de velocidades y aceleraciones



Published with MATLAB® 7.14

Table of Contents

Informe del mecanismo

Informe CAC 2D

Nombre del mecanismo

Problema4b *02-Aug-2016*

Datos del mecanismo

Datos básicos

Número de eslabones: 5 ; Número de pares de restricción: 5

Datos de los pares

Pares de revolución: 2 ; Pares prismáticos: 2

Lazo del mecanismo

{ 1 2 3 4 5 1 }

Lazo de los pares del mecanismo

{ 1 2 1 2 3 }

---------- Codificación de los pares ----------

```
   Par de revolución: 1   ;  Par prismático: 2
      Par de rodadura sin deslizamiento: 3

-------------------------------------------------
```

Geometría del mecanismo

Coordenadas de los pares [X,Y]

```
Par 1 : [ 1.519167e+02 ; 4.808326e+01 ]
Par 2 : [ 1.519167e+02 ; 4.808326e+01 ]
Par 3 : [ 1.193898e+02 ; 8.061017e+01 ]
Par 4 : [ 1.193898e+02 ; 8.061017e+01 ]
Par 5 : [ 9.438000e+01 ; 0 ]
```

Direcciones de los pares prismáticos

```
Par prismático 1 :
   Eslabón 2  y  Eslabón 3
Dirección del par: [ -7.071068e-01 ; 7.071068e-01 ]
-----------------
Par prismático 2 :
   Eslabón 4  y  Eslabón 5
Dirección del par: [ 0 ; 1 ]
```

Direciones de los pares de rodadura sin deslizamiento

```
Par de rodadura sin deslizamiento 1 :
   Eslabón 5  y  Eslabón 1
Dirección del par: [ 0 ; 1 ]
-----------------
```

Radios de curvatura de los pares de rodadura sin deslizamiento

```
Par de rodadura sin deslizamiento 1 :
    Radio de curvatura eslabón 5 : 50     ;     Radio curvatura eslabón 1
-----------------
```

Tipología de los pares de rodadura sin deslizamiento

```
Par de rodadura sin deslizamiento 1 :
   Eslabón 5  y  Eslabón 1
   Tipología 3 : Rueda - Plano
-----------------
```

Velocidades y acelecaciones iniciales

Velocidades iniciales

```
[w2 w3 w4 w5 Vx3 Vy3 Vx5 Vy5] : [ 2.077300e+00 2.077300e+00 0 0 0 0 ] um/
```

Acelecaciones iniciales

```
[a2 a3 a4 a5 Ax3 Ay3 Ax5 Ay5] : [ 0 0 0 0 0 0 ] um/s^2
```

Resultados

Velocidades y aceleraciones lineales

```
Relación 1 :
    Eslabón 1  y Eslabón 2
    Coordenadas : [ 1.519167e+02 ; 4.808326e+01 ]
    Velocidad lineal : [ 0 ; 0 ] um/s
    Acelecarión lineal : [ 0 ; 0 ] um/s^2

-----------------
Relación 2 :
    Eslabón 2  y Eslabón 3
    Coordenadas : [ 1.519167e+02 ; 4.808326e+01 ]
    Velocidad lineal : [ 6.756815e+01 ; 6.756813e+01 ] um/s
    Acelecarión lineal : [ 0 ; 0 ] um/s^2

-----------------
Relación 3 :
    Eslabón 3  y Eslabón 4
    Coordenadas : [ 1.193898e+02 ; 8.061017e+01 ]
    Velocidad lineal : [ -8.705302e+01 ; -5.753670e+01 ] um/s
    Acelecarión lineal : [ 3.196177e+02 ; 8.953705e+01 ] um/s^2

-----------------
Relación 4 :
    Eslabón 4  y Eslabón 5
    Coordenadas : [ 1.193898e+02 ; 8.061017e+01 ]
    Velocidad lineal : [ -8.705302e+01 ; -5.753670e+01 ] um/s
    Acelecarión lineal : [ 1.508971e+02 ; 2.015350e+02 ] um/s^2

-----------------
Relación 5 :
    Eslabón 5  y Eslabón 1
    Coordenadas : [ 9.438000e+01 ; 0 ]
    Velocidad lineal : [ 4.808326e+01 ; -5.753670e+01 ] um/s
    Acelecarión lineal : [ 1.508971e+02 ; 2.015350e+02 ] um/s^2
```

Velocidades y aceleraciones angulares

```
Eslabón 1 :
    Velocidad angular : 0 rad/s
    Acelecarión angular : 0 rad/s^2

-----------------
Eslabón 2 :
    Velocidad angular : 2.077300e+00 rad/s
    Acelecarión angular : -6.633240e-07 rad/s^2

-----------------
Eslabón 3 :
    Velocidad angular : 2.077300e+00 rad/s
    Acelecarión angular : -6.633240e-07 rad/s^2

-----------------
Eslabón 4 :
    Velocidad angular : 1.676417e+00 rad/s
    Acelecarión angular : -8.719360e-01 rad/s^2

-----------------
Eslabón 5 :
    Velocidad angular : 1.676417e+00 rad/s
    Acelecarión angular : -8.719360e-01 rad/s^2
```

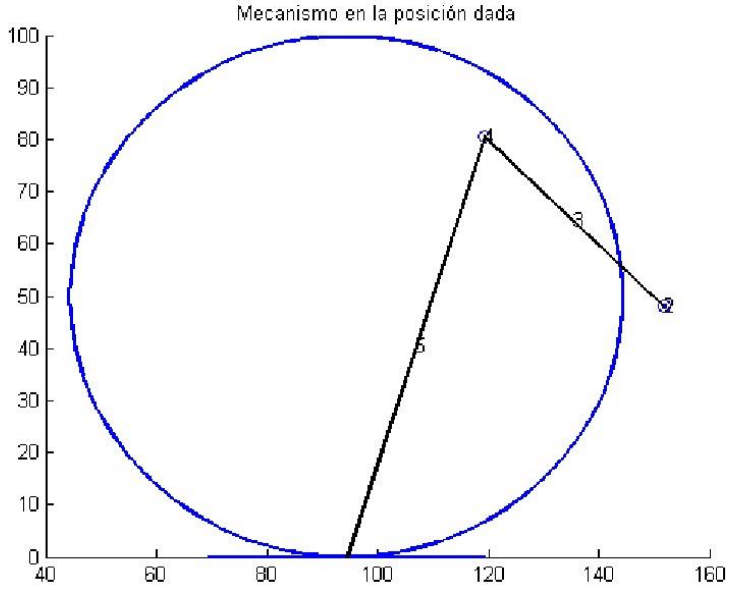
Comprobaciones de los resultados

```
CAC 2D comprueba los resultados obtenidos mediante
el método de cálculo con un error inferior a 1.000000e-05.
Si se detectase algún problema en los valores obtenidos
se comunicará un posible fallo.
```

Comprobación del problema de velocidad

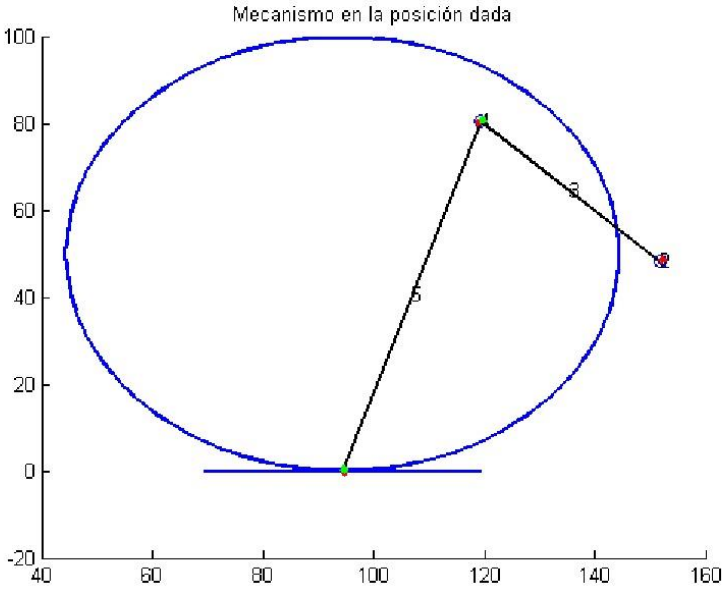
```
 Problema de velocidad
    recuelto con algun tipo de error


 Problema de aceleración
    recuelto con algun tipo de error
```

Esquema gráfico del mecanismo



5

Distribución de velocidades y aceleraciones



Published with MATLAB® 7.14

Trabajo Fin de Grado
Grado en Ingenieria en Tecnologías Industriales

Análisis Computacional Cinemático de Mecanismos Planos: Código Fuente

Autor: Jesús Medrán Castro
Tutor: Carmen Madrigal Sánchez

Dep. Ingeniería Mecánica y Fabricación
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016

Trabajo Fin de Grado
Grado en Ingeniería en Tecnologías Industriales

Análisis Computacional Cinemático de Mecanismos Planos: Código Fuente

Autor:
Jesús Medrán Castro

Tutor:
Carmen Madrigal Sánchez
Profesor titular

Dep. de Ingeniería Mecánica y Fabricación
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2016

Proyecto Fin de Carrera: Análisis Computacional Cinemático de Mecanismos Planos: Código Fuente

Autor: Jesús Medrán Castro

Tutor: Carmen Madrigal Sánchez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2016

El Secretario del Tribunal

A mi familia

A mis maestros

Resumen

Este documento se presenta como un texto adicional al Trabajo de Fin de Grado: *Análisis Computacional Cinemático de Mecanismos Planos* escrito por el alumno Jesús Medrán Castro. En las siguientes páginas se recoge el código fuente del programa *CAC 2D* escrito integramente en *Matlab*. Se presentan todas las funciones utilizadas ordenadas alfabeticamente.

Índice

ACELERELA.M

```
function [Meca] = AceleRela(Meca)
%% Función que implementa el método de las aceleraciones relativas %%

   % Creación de Meca.Cuerpo(i).A
     Meca.Cuerpo(1).A=0;
      for i=[1:1:Meca.Nbody-1]
          Meca.Cuerpo(i+1).A=Meca.SolAcel(i);
      end
   %

   % Implementación de Meca.SolVeloc=A41;A81..... en Meca.Rela().Ax Ay
      APP=[];
      for i=[1:1:length(Meca.Grupo)]
          fin=Meca.Grupo(i).G(end);
          fin=Meca.Lazo(fin+1);
          if fin==1
          else
              APP=[APP,fin];
          end
      end
    if length(APP)==0
        APP=0;
    end
    APP;
%     % Restrinción en caso de bucle abierto [Pruebas]
%        if Meca.Bucle==1
%            APP(end)=[];
%        end
%     %
    Elim=1;
    k=1;
  if APP==0
  else
    for i=[Meca.Nbody:1:length(Meca.SolAcel)]
        if k>length(APP)
        else
            R=APP(k);
        if Elim==1
            Meca.Rela(R).Ax=Meca.SolAcel(i);
            Elim=Elim+1;
        else
            Meca.Rela(R).Ay=Meca.SolAcel(i);
            Elim=1;
            k=k+1;
        end
        end
    end
  end
   %

%%% Código nuevo inicio %%%
MAR=zeros(2,Meca.Nbody);
   % Caso de no existir pares prismáticos NPP=0
      if Meca.NPP==0
          % Calcular matriz MT
            MT=[];
```

```
MW=[];
MV=[];
for i=[1:1:length(Meca.Rela)]
   Rf=Meca.Lazo(i);
   Ri=Meca.Lazo(i+1);
   Posfin=[Meca.Rela(Rf).x;Meca.Rela(Rf).y];
   Posini=[Meca.Rela(Ri).x;Meca.Rela(Ri).y];
   T=Posfin-Posini;
   T=[-T(2);T(1)];
   MT=[MT,T];
end
for i=[1:1:length(MT(1,:))-1]
    W=Meca.Cuerpo(i+1).W;
    T=MT(:,i);
    T=[T(2);-T(1)];
    MW(:,i)=T.*(W.^2);
end
MW=[[0;0] MW];
for i=[1:1:length(MT(1,:))-1]
   A=Meca.Cuerpo(i+1).A;
   MT(:,i)=MT(:,i).*A;
end
MT(:,end)=[];
MT=[[0;0] MT];
k=1;
for i=[1:1:length(Meca.Rela)]
    if Meca.Rela(i).Codig==3
      Ri=Meca.Radios(k,1);
      Rj=Meca.Radios(k,2);
      landa=Meca.DirecPRSD(k,:);
      landa=(landa)./norm(landa)';
      wi=Meca.Cuerpo(Meca.Rela(i).Cuerpos(1)).W;
      wj=Meca.Cuerpo(Meca.Rela(i).Cuerpos(2)).W;
        if Meca.Radios(k,3)==1
           aij=((wi-wj).^2).*((Ri.*Rj)./(Ri-Rj)).*landa';
        end
        if Meca.Radios(k,3)==2
            aij=((wi-wj).^2).*((Ri.*Rj)./(Ri+Rj)).*landa';
        end
        if Meca.Radios(k,3)==3
            aij=((wi-wj).^2).*Ri.*landa';
        end
        if Meca.Radios(k,3)==4
            aij=((wi-wj).^2).*Rj.*landa';
        end
        V=[Meca.Rela(i).Vx;Meca.Rela(i).Vy];
        V=[-V(2);V(1)];
        MV(:,i)=aij+2*wi.*V;
        k=k+1;
    else
        MV(:,i)=[0;0];
    end
end
MAR=-MT-MW-MV;
%
% Igualación de los extremos a cero si es par de revolución
 if Meca.LazoPares(1)==1
     MAR(:,1)=0;
 end
 if Meca.LazoPares(end)==1
     MAR(:,end)=0;
 end
%
```

```
        % Proceso de suma
           for i=[2:1:length(MAR(1,:))]
               MAR(:,i)=MAR(:,i)+MAR(:,i-1);
           end
        %
        % Colocación MAR en Meca.Rela(i).Ax y Meca.Rela(i).Ay
               MAR(:,1)=0;
               MAR(:,end)=0;
           for i=[1:1:length(MAR(1,:))]
               Meca.Rela(i).Ax=MAR(1,i);
               Meca.Rela(i).Ay=MAR(2,i);
           end
        %
    end
%
% Caso de solo un par prismático NPP=1
    if Meca.NPP==1
        % Calcular matriz MT
          MT=[];
          MV=[];
          MW=[];
          for i=[1:1:length(Meca.Rela)]
             Ri=Meca.Lazo(i);
             Rf=Meca.Lazo(i+1);
             Posfin=[Meca.Rela(Rf).x;Meca.Rela(Rf).y];
             Posini=[Meca.Rela(Ri).x;Meca.Rela(Ri).y];
             T=Posfin-Posini;
             T=[-T(2);T(1)];
             MT=[MT,T];
          end
          for i=[1:1:length(MT(1,:))-1]
              W=Meca.Cuerpo(i+1).W;
              T=MT(:,i);
              T=[T(2);-T(1)];
              MW(:,i)=T.*(W.^2);
          end
          MW=[[0;0] MW];
          for i=[1:1:length(MT(1,:))-1]
             A=Meca.Cuerpo(i+1).A;
             MT(:,i)=MT(:,i).*A;
          end
          MT(:,end)=[];
          MT=[[0;0] MT];
          k=1;
          for i=[1:1:length(Meca.Rela)]
              if Meca.Rela(i).Codig==3
                Ri=Meca.Radios(k,1);
                Rj=Meca.Radios(k,2);
                landa=Meca.DirecPRSD(k,:);
                landa=(landa)./norm(landa)';
                wi=Meca.Cuerpo(Meca.Rela(i).Cuerpos(1)).W;
                wj=Meca.Cuerpo(Meca.Rela(i).Cuerpos(2)).W;
                  if Meca.Radios(k,3)==1
                     aij=((wi-wj).^2).*((Ri.*Rj)./(Ri-Rj)).*landa';
                  end
                  if Meca.Radios(k,3)==2
                      aij=((wi-wj).^2).*((Ri.*Rj)./(Ri+Rj)).*landa';
                  end
                  if Meca.Radios(k,3)==3
                      aij=((wi-wj).^2).*Ri.*landa';
                  end
                  if Meca.Radios(k,3)==4
```

```
                    aij=((wi-wj).^2).*Rj.*landa';
                end
                V=[Meca.Rela(i).Vx;Meca.Rela(i).Vy];
                V=[-V(2);V(1)];
                MV(:,i)=aij+2*wi.*V;
                k=k+1;
            else
                MV(:,i)=[0;0];
            end
        end
        MAR=MT-MW-MV;
    %
    % Igualación de los extremos a cero si es par de revolución
        if Meca.LazoPares(1)==1
            MAR(:,1)=0;
        end
        if Meca.LazoPares(end)==1
            MAR(:,end)=0;
        end
    %
    % Proceso de suma
        NVPP=[];
        for i=[1:1:length(Meca.LazoPares)]
            if Meca.LazoPares(i)==2
                NVPP=[NVPP,i];
            end
        end
        NVPP;
        % Suma-> Caso de NPP en primera posición
        if NVPP==1
            MAR=-MAR;
            for i=[length(MAR(1,:))-1:-1:1]
                MAR(:,i)=MAR(:,i)+MAR(:,i+1);
            end
        end
        %
        % Suma-> Caso de NPP en última posición
        if NVPP==Meca.Nrest
            for i=[2:1:length(MAR(1,:))]
                MAR(:,i)=MAR(:,i)+MAR(:,i-1);
            end
        end
        %
        % Suma-> Caso de NPP en cualquier posición
        if NVPP~=1 && NVPP~=Meca.Nrest
            for i=[2:1:NVPP]
                MAR(:,i)=MAR(:,i)+MAR(:,i-1);
            end
            for i=[length(MAR(1,:))-1:-1:NVPP+1]
                MAR(:,i)=-(MAR(:,i)+MAR(:,i+1));
            end
        end
        %
    %
    % Colocación MAR en Meca.Rela(i).Ax y Meca.Rela(i).Ay
            MAR(:,1)=0;
        for i=[1:1:length(MAR(1,:))]
            Meca.Rela(i).Ax=MAR(1,i);
            Meca.Rela(i).Ay=MAR(2,i);
        end
    %
end
%
```

```
% Caso de más de un par prismático NPP>=2
   if Meca.NPP>=2
       % Calcular matriz MT
         MT=[];
         for i=[1:1:length(Meca.Rela)]
            Ri=Meca.Lazo(i);
            Rf=Meca.Lazo(i+1);
            Posfin=[Meca.Rela(Rf).x;Meca.Rela(Rf).y];
            Posini=[Meca.Rela(Ri).x;Meca.Rela(Ri).y];
            T=Posfin-Posini;
            T=[-T(2);T(1)];
            MT=[MT,T];
         end
         MT(:,end)=[];
         MT=[[0;0] MT];
         for i=[1:1:length(MT(1,:))]
             W=Meca.Cuerpo(i).W;
             T=MT(:,i);
             T=[T(2);-T(1)];
             MW(:,i)=T.*(W.^2);
         end
         for i=[1:1:length(MT(1,:))-1]
            A=Meca.Cuerpo(i+1).A;
            MT(:,i)=MT(:,i).*A;
         end
         APP;
         for i=[1:1:length(APP)]
            MT(:,APP(i))=0;
         end
       %
       % Introducir las A calculadas de Meca.SolAce
          for i=[1:1:length(APP)]
             MAR(1,APP(i))=Meca.Rela(APP(i)).Ax;
             MAR(2,APP(i))=Meca.Rela(APP(i)).Ay;
          end
       %
         k=1;
         for i=[1:1:length(Meca.Rela)]
             if Meca.Rela(i).Codig==3
               Ri=Meca.Radios(k,1);
               Rj=Meca.Radios(k,2);
               landa=Meca.DirecPRSD(k,:);
               landa=(landa)./norm(landa)';
               wi=Meca.Cuerpo(Meca.Rela(i).Cuerpos(1)).W;
               wj=Meca.Cuerpo(Meca.Rela(i).Cuerpos(2)).W;
                 if Meca.Radios(k,3)==1
                    aij=((wi-wj).^2).*((Ri.*Rj)./(Ri-Rj)).*landa';
                 end
                 if Meca.Radios(k,3)==2
                     aij=((wi-wj).^2).*((Ri.*Rj)./(Ri+Rj)).*landa';
                 end
                 if Meca.Radios(k,3)==3
                     aij=((wi-wj).^2).*Ri.*landa';
                 end
                 if Meca.Radios(k,3)==4
                     aij=((wi-wj).^2).*Rj.*landa';
                 end
                 V=[Meca.Rela(i).Vx;Meca.Rela(i).Vy];
                 V=[-V(2);V(1)];
                 MV(:,i)=aij+2*wi.*V;
                 k=k+1;
             else
```

```
                    MV(:,i)=[0;0];
                end
            end
%             MT
%             MW
%             MV
%             MAR
            MV(:,end)=[];
            MV=[[0;0] MV];
          MAR=MT-MV-MW+MAR;
          % Igualación de los extremos a cero si es par de revolución
             if Meca.LazoPares(1)==1
                 MAR(:,1)=0;
             end
             if Meca.LazoPares(end)==1
                 MAR(:,end)=0;
             end
          %
          % Proceso de suma hacia la izquierda
             NVPP=[];
             for i=[1:1:length(Meca.LazoPares)]
                  if Meca.LazoPares(i)==2
                     NVPP=[NVPP,i];
                  end
             end
             NVPP(end)=[];
             k=0;
             for i=[length(MAR(1,:))-1:-1:1]
                 if length(NVPP)-k<=0
                     MAR(:,i)=MAR(:,i)+MAR(:,i+1);
                 else
                     if i==NVPP(end-k)
                         k=k+1;
                     else
                         MAR(:,i)=MAR(:,i)+MAR(:,i+1);
                     end
                 end
                 MAR;
             end
          %
          % Colocación MAR en Meca.Rela(i).Ax y Meca.Rela(i).Ay
             if Meca.LazoPares(1)==1
                 MAR(:,1)=0;
             end
             for i=[1:1:length(MAR(1,:))]
                 Meca.Rela(i).Ax=MAR(1,i);
                 Meca.Rela(i).Ay=MAR(2,i);
             end
          %
      end
   %
%%% Código nuevo final %%%

%% Código nuevo %%


%%%%%%%%%%%%%%%%%%


end
```

ACERCADE.M

```
function varargout = AcercaDe(varargin)
% ACERCADE MATLAB code for AcercaDe.fig
%      ACERCADE, by itself, creates a new ACERCADE or raises the existing
%      singleton*.
%
%      H = ACERCADE returns the handle to a new ACERCADE or the handle to
%      the existing singleton*.
%
%      ACERCADE('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ACERCADE.M with the given input arguments.
%
%      ACERCADE('Property','Value',...) creates a new ACERCADE or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before AcercaDe_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to AcercaDe_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help AcercaDe

% Last Modified by GUIDE v2.5 05-Jul-2016 18:28:56

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @AcercaDe_OpeningFcn, ...
                   'gui_OutputFcn',  @AcercaDe_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before AcercaDe is made visible.
function AcercaDe_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to AcercaDe (see VARARGIN)

% Choose default command line output for AcercaDe
```

```
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

Logo=imread('Logo1.png');
axes(handles.Logo);
imshow(Logo)
axis off

ETSI=imread('logoESI.jpg');
axes(handles.ETSI);
imshow(ETSI)
axis off

US=imread('logoUS.jpg');
axes(handles.US);
imshow(US)
axis off

% UIWAIT makes AcercaDe wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = AcercaDe_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

AYUDADIRECPRSD.M

```
function varargout = AyudaDirecPRSD(varargin)
% AYUDADIRECPRSD MATLAB code for AyudaDirecPRSD.fig
%      AYUDADIRECPRSD, by itself, creates a new AYUDADIRECPRSD or raises the
existing
%      singleton*.
%
%      H = AYUDADIRECPRSD returns the handle to a new AYUDADIRECPRSD or the
handle to
%      the existing singleton*.
%
%      AYUDADIRECPRSD('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in AYUDADIRECPRSD.M with the given input
arguments.
%
%      AYUDADIRECPRSD('Property','Value',...) creates a new AYUDADIRECPRSD or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before AyudaDirecPRSD_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to AyudaDirecPRSD_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help AyudaDirecPRSD

% Last Modified by GUIDE v2.5 05-Jul-2016 10:59:00

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @AyudaDirecPRSD_OpeningFcn, ...
                   'gui_OutputFcn',  @AyudaDirecPRSD_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before AyudaDirecPRSD is made visible.
function AyudaDirecPRSD_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to AyudaDirecPRSD (see VARARGIN)

% Choose default command line output for AyudaDirecPRSD
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes AyudaDirecPRSD wait for user response (see UIRESUME)
% uiwait(handles.figure1);
ConCov=imread('ConCov1.jpg');
axes(handles.ConcavoConvexo);
imshow(ConCov)
axis off

CovCon=imread('CovCon.jpg');
axes(handles.ConvexoConcavo);
imshow(CovCon)
axis off

RuedaPlano=imread('RuedaPlano.jpg');
axes(handles.RuedaPlano);
imshow(RuedaPlano)
axis off

PlanoRueda=imread('PlanoRueda.jpg');
axes(handles.PlanoRueda);
imshow(PlanoRueda)
axis off




% --- Outputs from this function are returned to the command line.
function varargout = AyudaDirecPRSD_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

AYUDARADIOS.M

```
function varargout = AyudaRadios(varargin)
% AYUDARADIOS MATLAB code for AyudaRadios.fig
%      AYUDARADIOS, by itself, creates a new AYUDARADIOS or raises the
existing
%      singleton*.
%
%      H = AYUDARADIOS returns the handle to a new AYUDARADIOS or the handle
to
%      the existing singleton*.
%
%      AYUDARADIOS('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in AYUDARADIOS.M with the given input
arguments.
%
%      AYUDARADIOS('Property','Value',...) creates a new AYUDARADIOS or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before AyudaRadios_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to AyudaRadios_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help AyudaRadios

% Last Modified by GUIDE v2.5 19-May-2016 20:46:07

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @AyudaRadios_OpeningFcn, ...
                   'gui_OutputFcn',  @AyudaRadios_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before AyudaRadios is made visible.
function AyudaRadios_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% varargin   command line arguments to AyudaRadios (see VARARGIN)

% Choose default command line output for AyudaRadios
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

Ayuda=imread('AyudaRadios1.jpg');
axes(handles.Grafica);
image(Ayuda)
axis off

% UIWAIT makes AyudaRadios wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = AyudaRadios_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes when figure1 is resized.
function figure1_ResizeFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on scroll wheel click while the figure is in focus.
function figure1_WindowScrollWheelFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  structure with the following fields (see FIGURE)
%   VerticalScrollCount: signed integer indicating direction and number of
clicks
%   VerticalScrollAmount: number of lines scrolled for each click
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

BUCLEAC.M

```
function [Meca] = BucleAC(Meca)
%% Función que modifica los datos de Meca en caso de Bucle abierto %%

   % Caso de bucle cerrado
     if Meca.Lazo(end)==1
         Meca.Bucle=0;
     end
   %

   % Caso de bucle abierto
      if Meca.Lazo(end)~=1
          Meca.Nrest=Meca.Nrest+1;
          Meca.NPP=Meca.NPP+1;
          Meca.Lazo=[Meca.Lazo,1];
          Meca.LazoPares=[Meca.LazoPares,2];
          Meca.Bucle=1;
      end
   %

end
```

CALCULADOR.M

```
function [Meca] = Calculador(Meca)
%% Función que resuelve de nuevo el problema de velocidad y aceleración para
su posterior simulación %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Intento de implementación de resolución del problema de velocidad V5 %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Cálculos %%
   % Determinación de bucle abierto o cerrado
      Meca=BucleAC(Meca);
   %
   % Creación de la subestructura Meca.Rela
      Meca=Relaciones(Meca);
   %
   % Creación de los grupos para la matriz TH - Meca.Grupo().G
      Meca=CreaGrupos(Meca);
   %
   % Creación de los tramos Meca.Tramo().T
      Meca=CreaTramos(Meca);
   %

   % Diferenciación del caso de NPP=0 y NPP~=0
      if Meca.NPP==0
          % Caso particular de NPP=0
             Meca=NOPP(Meca);
          %
      else
          % Creación de la matriz TH
             Meca=MatrizTH(Meca);
          %
          % Creación de la matriz HV
             Meca=MatrizHV(Meca);
          %
          % Unión de las matrices TH y HV. Creación de la Matriz Meca.Q
             Q=[Meca.TH,-Meca.HV];
             Meca.Q=Q;
          %
          % Aplicación de la condición de par prismático (Wi=Wj) a Meca.Q
             Meca=CondPrismatico(Meca);
          %
      end
          % Aplicación de los GDL y creación de las matrices Meca.Qind y
Meca.Qdep
             Meca=CondVGDL(Meca);
          %
          % Resolución del problema de velocidad
            Sol=Meca.Qdep\-Meca.Qind;
            Sol=Meca.VGDL+Sol;
            Meca.SolVeloc=Sol;
            if Meca.NPP~=0
               Meca=CondPrisSolV(Meca);
            end
          %
   %

   % Aplicación del Método de las velocidades relativas
      Meca=VelocRela(Meca);
   %
```

```
   % Resolución del problema de aceleraciones
      Meca=ProbAcel(Meca);
   %

   % Aplicación del Método de las aceleraciones relativas
      Meca=AceleRela(Meca);
   %



%%%%%%%%%%%%%%

%% Comprobaciones finales %%
   % Comprobador del resultado del problema de velocidad
         Meca=CompruebaVel(Meca);
         aux=sum(abs(Meca.CompruebaVel));
         ErrorV=log10(aux);
         if ErrorV==-Inf
             fprintf('\n--> Problema de velocidad correcto\n    con error
0\n')
         else
              if aux<10^-5
                 fprintf('\n--> Problema de velociad correcto\n    con un
errror del 10^%d\n',ErrorV)
              else
                 fprintf('\n--> Problema de velocidad\n    recuelto con algun
tipo de error\n')
              end
         end
   %

   % Comprobador del resultado del problema de aceleración
       Meca=CompruebaAce(Meca);
         aux=sum(abs(Meca.CompruebaAce));
         ErrorA=log10(aux);
         if ErrorA==-Inf
             fprintf('\n--> Problema de aceleración correcto\n    con error
0\n')
         else
              if aux<10^-5
                 fprintf('\n--> Problema de aceleración correcto\n    con un
errror del 10^%d\n',ErrorV)
              else
                 fprintf('\n--> Problema de aceleración\n    recuelto con
algun tipo de error\n')
              end
         end
   %


%%%%%%%%%%%%%%%%%%%%%%%%%%%

end
```

CARGA_ARCHIVOS.M

```
function [Meca] = Carga_Archivos(Directorio)
%% Función que carga un archivo guardado %%

   Directorio=char(Directorio);
   cd(Directorio);

   Nbody=fopen('Nbody.txt','r');
     Nbody=fscanf(Nbody,'%c',inf);
     Meca.Nbody=Nbody;

   Nrest=fopen('Nrest.txt','r');
     Nrest=fscanf(Nrest,'%c',inf);
     Meca.Nrest=Nrest;

   NPR=fopen('NPR.txt','r');
     NPR=fscanf(NPR,'%c',inf);
     Meca.NPR=NPR;

   NPP=fopen('NPP.txt','r');
     NPP=fscanf(NPP,'%c',inf);
     Meca.NPP=NPP;

   NPRSD=fopen('NPRSD.txt','r');
     NPRSD=fscanf(NPRSD,'%c',inf);
     Meca.NPRSD=NPRSD;

   Lazo=fopen('Lazo.txt','r');
      Lazo=fscanf(Lazo,'%c',inf);
      Meca.Lazo=Lazo;

   LazoPares=fopen('LazoPares.txt','r');
      LazoPares=fscanf(LazoPares,'%c',inf);
      Meca.LazoPares=LazoPares;

   VGDL=fopen('VGDL.txt','r');
      VGDL=fscanf(VGDL,'%c',inf);
      Meca.VGDL=VGDL;

   AGDL=fopen('AGDL.txt','r');
      AGDL=fscanf(AGDL,'%c',inf);
      Meca.AGDL=AGDL;

   Coord=fopen('Coord.txt','rt');
   M=[];
   while (~feof(Coord))
      x=fscanf(Coord,'%f %f',[1 2]);
      M=[M;x];
   end
   Coord=M;
      Meca.Coord=double(Coord);

   DirecPRSD=fopen('DirecPRSD.txt','r');
   M=[];
   while (~feof(DirecPRSD))
      x=fscanf(DirecPRSD,'%f %f %f',[1 3]);
      M=[M;x];
```

```
    end
    DirecPRSD=M;
       Meca.DirecPRSD=double(DirecPRSD);

    DirecPP=fopen('DirecPP.txt','r');
    M=[];
    while (~feof(DirecPP))
       x=fscanf(DirecPP,'%f %f',[1 2]);
       M=[M;x];
    end
    DirecPP=M;
       Meca.DirecPP=double(DirecPP);

    Radios=fopen('Radios.txt','r');
    M=[];
    while (~feof(Radios))
       x=fscanf(Radios,'%f %f',[1 2]);
       M=[M;x];
    end
    Radios=M;
       Meca.Radios=double(Radios);

end
```

COMPINI.M

```
function [Meca] = CompIni(Nbody,Nrest,NPR,NPP,NPRSD,Lazo,LazoPares,VGDL,AGDL)
%% Función realiza las comprobaciones iniciales %%

%% Comprobaciones iniciales %%
   % Comprobación 1
      if length(LazoPares)+1~=length(Lazo) || length(Lazo)~=Nrest+1
          fprintf('\n --> >>ERROR<<  Comprobación 1\n')
          Comp1=1;
      else
          Comp1=0;
      end
   %
   % Comprobación 2
      if Nrest~=NPR+NPP+NPRSD
         fprintf('\n --> >>ERROR<<  Comprobación 2\n')
         Comp2=1;
      else
          Comp2=0;
      end
   %
   % Comprobación 3
   npr=0;npp=0;nprsd=0;
      for i=[1:1:length(LazoPares)]
         if LazoPares(i)==1
             npr=npr+1;
         end
         if LazoPares(i)==2
             npp=npp+1;
         end
         if LazoPares(i)==3
             nprsd=nprsd+1;
         end
      end
     if npr~=NPR || npp~=NPP || nprsd~=NPRSD
         fprintf('\n --> >>ERROR<<  Comprobación 3\n')
         Comp3=1;
     else
         Comp3=0;
     end
   %
   % Comprobación 4
      Cont=0;
      for i=[1:1:length(LazoPares)]
          if LazoPares(i)==2
              if i==1
              else
                  Cont=Cont+1;
              end
          end
      end
      if NPP==1
          if length(VGDL)~=Nbody-1;
              fprintf('\n --> <<ERROR>>  Comprobación 4\n')
              Comp4=1;
          else
              if length(VGDL)~=length(AGDL)
                 fprintf('\n --> <<ERROR>>  Comprobación 4\n')
                 Comp4=1;
```

```
                else
                    Comp4=0;
                end
            end
        else
            if length(VGDL)~=Nbody-1+Cont*2
               fprintf('\n --> <<ERROR>>  Comprobación 4\n')
               Comp4=1;
            else
                if length(VGDL)~=length(AGDL)
                    fprintf('\n --> <<ERROR>>  Comprobación 4\n')
                    Comp4=1;
                else
                    Comp4=0;
                end
            end
        end
    %

        Meca.CompIni.Comp1=Comp1;
        Meca.CompIni.Comp2=Comp2;
        Meca.CompIni.Comp3=Comp3;
        Meca.CompIni.Comp4=Comp4;

end
```

COMPRUEBAACE.M

```
function [Meca] = CompruebaAce(Meca)
%% Función Comprobador para el problema de aceleración

% Cálculo del término de la izquierda

if Meca.NPP==0
  P=0;
    for i=[1:1:length(Meca.Grupo(1).G)]
        u=Meca.Grupo(1).G(i);
           if u==Meca.Nbody
               w=0;
               A=0;
           else
               w=Meca.Cuerpo(u+1).W;
               A=Meca.Cuerpo(u+1).A;
           end
         T=Meca.Tramo(u).T;
         Tp=[-T(2);T(1)];
         P=P+(Tp.*A)-(T.*((w).^2));
    end
   Izq=P;
else
      Izq=[];
      k=1;
      for i=[1:1:length(Meca.Grupo)]
          AT=0;
          WT=0;
          AWV=0;
          for j=[1:1:length(Meca.Grupo(i).G)]
              u=Meca.Grupo(i).G(j);
               if u==Meca.Nbody
                  w=0;
                  A=0;
               else
                   w=Meca.Cuerpo(u+1).W;
                  A=Meca.Cuerpo(u+1).A;
               end
              T=Meca.Tramo(u).T;
              T=[-T(2);T(1)];
              hp=Meca.DirecPP(i,:)';
              hp=[-hp(2);hp(1)];

              AT=AT+A*sum(T.*hp);
              WT=WT+(w^2)*sum([T(2);-T(1)].*hp);
              if Meca.Rela(u).Codig==3
                     Ri=Meca.Radios(k,1);
                     Rj=Meca.Radios(k,2);
                     landa=Meca.DirecPRSD(k,:);
                     landa=(landa)./norm(landa)';
                     wi=Meca.Cuerpo(Meca.Rela(u).Cuerpos(1)).W;
                     wj=Meca.Cuerpo(Meca.Rela(u).Cuerpos(2)).W;
                       if Meca.Radios(k,3)==1
                          aij=((wi-wj).^2).*((Ri.*Rj)./(Ri-Rj)).*landa';
                       end
                       if Meca.Radios(k,3)==2
                           aij=((wi-wj).^2).*((Ri.*Rj)./(Ri+Rj)).*landa';
                       end
                       if Meca.Radios(k,3)==3
```

```
                            aij=((wi-wj).^2).*Ri.*landa';
                        end
                        if Meca.Radios(k,3)==4
                            aij=((wi-wj).^2).*Rj.*landa';
                        end
                        V=[Meca.Rela(u).Vx;Meca.Rela(u).Vy];
                        V=[-V(2);V(1)];
                        aux=aij+2*wj.*V;
                        AWV=AWV+sum(aux.*hp);
                        k=k+1;
                end
            end
            Izq=[Izq;AT-WT-AWV];
        end
end
%

% Cálculo del término de la derecha
if Meca.NPP>=2
    Der=[];
    RCP=Meca.RCP;
    if Meca.Bucle==1
        RCP(end,:)=[];
    end

    Linea=[RCP(:,1)' 1];
    RCP;
    for i=[1:1:length(RCP(:,1))]
        hp=Meca.DirecPP(i,:)';
        hp=[-hp(2);hp(1)];
        fin=Linea(i+1);
        ini=Linea(i);
        Afin=[Meca.Rela(fin).Ax;Meca.Rela(fin).Ay];
        Aini=[Meca.Rela(ini).Ax;Meca.Rela(ini).Ay];
        P=sum((Afin-Aini).*hp);
        Der=[Der;P];
    end
else
    Der=0;
end

if Meca.NPP==0
     P=[0;0];
     cont=1;
      for i=[1:1:length(Meca.Grupo(1).G)]
          u=Meca.Grupo(1).G(i);
            if Meca.Rela(u).Codig==3
                wi=Meca.Cuerpo(u).W;
                  if u==Meca.Nbody
                      wj=0;
                  else
                      wj=Meca.Cuerpo(u+1).W;
                  end
                 Ri=Meca.Radios(cont,1);
                 Rj=Meca.Radios(cont,2);
                 landa=Meca.DirecPRSD(cont,:);
                 landa=(landa)./norm(landa);
                    if Meca.Radios(cont,3)==1
                         aij=((wi-wj).^2).*((Ri.*Rj)./(Ri-Rj)).*landa;
                    end
                    if Meca.Radios(cont,3)==2
```

```
                    aij=((wi-wj).^2).*((Ri.*Rj)./(Ri+Rj)).*landa;
                end
                if Meca.Radios(cont,3)==3
                    aij=((wi-wj).^2).*(Ri).*landa;
                end
                if Meca.Radios(cont,3)==4
                    aij=((wi-wj).^2).*(Rj).*landa;
                end
            P=P+aij';
            vi=[Meca.Rela(u).Vx;Meca.Rela(u).Vy];
            wv=2.*[-vi(2);vi(1)].*wj;
            P=P+(wv);
            cont=cont+1;
        else
            P=[0;0];
        end
    end
    Der=P;
end

%
    if Meca.Bucle==1
        Der=[Der;0];
    end

Meca.CompruebaAce=Izq-Der;
end
```

COMPRUEBAVEL.M

```
function [Meca] = CompruebaVel(Meca)
%% Función Comprobador para el problema de velocidad

% Cálculo del término de la izquierda
   Izq=[];
  if Meca.NPP==0
    P=0;
      for i=[1:1:length(Meca.Grupo(1).G)]
          u=Meca.Grupo(1).G(i);
             if u==Meca.Nbody
                 w=0;
             else
                 w=Meca.Cuerpo(u+1).W;
             end
          T=Meca.Tramo(u).T;
          T=[-T(2);T(1)];
          P=P+T.*w;
      end
      Izq=P;
  else
      for i=[1:1:length(Meca.Grupo)]
          P=0;
          for j=[1:1:length(Meca.Grupo(i).G)]
              u=Meca.Grupo(i).G(j);
              if u==Meca.Nbody
                  w=0;
              else
                  w=Meca.Cuerpo(u+1).W;
              end
              T=Meca.Tramo(u).T;
              T=[-T(2);T(1)];
              hp=Meca.DirecPP(i,:)';
              hp=[-hp(2);hp(1)];
              P=P+w*sum(T.*hp);
          end
          Izq=[Izq;P];
      end
  end
%

% Cálculo del término de la derecha
   Der=[];
   if Meca.NPP>=2
      RCP=Meca.RCP;
      if Meca.Bucle==1
          RCP(end,:)=[];
      end

      Linea=[RCP(:,1)' 1];

      for i=[1:1:length(RCP(:,1))]
          hp=Meca.DirecPP(i,:)';
          hp=[-hp(2);hp(1)];
          fin=Linea(i+1);
          ini=Linea(i);
          Vfin=[Meca.Rela(fin).Vx;Meca.Rela(fin).Vy];
```

```
            Vini=[Meca.Rela(ini).Vx;Meca.Rela(ini).Vy];
            P=sum((Vfin-Vini).*hp);
            Der=[Der;P];
        end
    else
        Der=0;
    end

 if Meca.NPP==0
     Der=[0;0];
 end

%
    if Meca.Bucle==1
        Der=[Der;0];
    end
Meca.CompruebaVel=Izq-Der;
end
```

CONDAGDL.M

```
function [Meca] = CondAGDL(Meca)
%% Función que aplica los GDL y crea las matrices Meca.Qdep y Meca.Qdep %%
   % Aceleraciones %
   Qdep=Meca.M;
   for i=[1:1:length(Meca.AGDL)]
       if Meca.AGDL(i)==0
       else
           Qdep(:,i)=0;
       end
   end
   Meca.Qdep=Qdep;

   Qind=Meca.M*Meca.AGDL;
   Meca.Qind=Qind;

end
```

CONDICIONES_DE_USO.M

```
function varargout = Condiciones_de_uso(varargin)
% CONDICIONES_DE_USO MATLAB code for Condiciones_de_uso.fig
%      CONDICIONES_DE_USO, by itself, creates a new CONDICIONES_DE_USO or
raises the existing
%      singleton*.
%
%      H = CONDICIONES_DE_USO returns the handle to a new CONDICIONES_DE_USO
or the handle to
%      the existing singleton*.
%
%      CONDICIONES_DE_USO('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in CONDICIONES_DE_USO.M with the given input
arguments.
%
%      CONDICIONES_DE_USO('Property','Value',...) creates a new
CONDICIONES_DE_USO or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Condiciones_de_uso_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Condiciones_de_uso_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Condiciones_de_uso

% Last Modified by GUIDE v2.5 20-Jul-2016 13:53:35

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Condiciones_de_uso_OpeningFcn, ...
                   'gui_OutputFcn',  @Condiciones_de_uso_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Condiciones_de_uso is made visible.
function Condiciones_de_uso_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Condiciones_de_uso (see VARARGIN)

% Choose default command line output for Condiciones_de_uso
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

Logo=imread('Logo1.png');
axes(handles.Logo);
imshow(Logo)
axis off

% UIWAIT makes Condiciones_de_uso wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Condiciones_de_uso_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on button press in Continuar.
function Continuar_Callback(hObject, eventdata, handles)
% hObject    handle to Continuar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

Prueba11()
close(Condiciones_de_uso)




% --- Executes on button press in NVM.
function checkbox2_Callback(hObject, eventdata, handles)
% hObject    handle to NVM (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of NVM
```

CONDPRISMATICO.M

```
function [Meca] = CondPrismatico(Meca)
%% Función que aplica la condición de par prismático (Wi=Wj) en Meca.Q %%

   RCP=[];
   for i=[1:1:length(Meca.Rela)]
       if Meca.Rela(i).Codig==2
           C1=Meca.Rela(i).Cuerpos(1);
           C2=Meca.Rela(i).Cuerpos(2);
           RCP=[RCP;C1,C2];
       end
   end
   Meca.RCP=RCP;
   % Paso la Matriz RCP a versión número de las columnas
      RCP=RCP-1;
   %
   % Procedimiento de suma de las columnas
      M=Meca.Q;

      % Restricción en caso de bucle abierto [Pruebas]
%         if Meca.Bucle==1
%             RCP(end,:)=[];
%         end
      %

      for Repeticiones=[1:1:Meca.NPP*5]
       % Casos en los que Wi=W1=0
         for i=[1:1:length(RCP(:,1))]
             a=RCP(i,1);
             b=RCP(i,2);
             if a==0
                 M(:,b)=0;
             end
             if b==0
                 M(:,a)=0;
             end
         end
       %
       for i=[1:1:length(RCP(:,1))]
           a=RCP(i,1);
           b=RCP(i,2);
           if a~=0 && b~=0
               M(:,a)=M(:,a)+M(:,b);
               M(:,b)=0;
           end
       end
   %
      end
   % Eliminación de columnas que sean Wi=Wj=Wk=W1=0
   if Meca.NPP>1
     LineaRCP=[];
      for i=[1:1:length(RCP(:,1))]
          LineaRCP=[LineaRCP,RCP(i,:)];
      end
      fin=0;
      i=1;
      LZ=[];
      RCP;
      while fin==0
```

```
        if RCP(i,2)==RCP(i+1,1)
            LZ=[LZ,RCP(i+1,1)];
            i=i+1;
               if i==length(RCP(:,1))
                   fin=1;
               end
        else
            fin=1;
        end

      end
      i=0;
      fin=0;
      while fin==0
        if RCP(end-i,1)==RCP(end-i-1,2)
            LZ=[LZ,RCP(end-i-1,:)];
            i=i+1;
               if i-1==0
                   fin=1;
               end
        else
            fin=1;
        end
      end
      LZ;
      for i=[1:1:length(LZ)]
          M(:,LZ(i))=0;
      end
   end
   %

   Meca.M=M;

end
```

CONDPRISSOLA.M

```
function [Meca] = CondPrisSolA(Meca)
%% Función que implementa las condiciones prismáticas Wi=Wj en la solución%%

   RCP=Meca.RCP-1;

   % Restrinción en caso de bucle abierto [Pruebas]
      if Meca.Bucle==1
          RCP(end,:)=[];
      end
   %

   Sol=Meca.SolAcel;
   for i=[1:1:length(RCP(:,1))]
        a=RCP(i,1);
        b=RCP(i,2);
          if a==0
              Sol(b)=0;
          end
          if b==0
             Sol(a)=0;
          end
   end
   RCP;
   for i=[1:1:length(RCP(:,1))]
       a=RCP(i,1);
       b=RCP(i,2);
          if a~=0 && b~=0
              S1=Sol(a);
              S2=Sol(b);
              if S1==0 && S2==0
              end
              if S1==0 && S2~=0
                  Sol(a)=S2;
              end
              if S1~=0 && S2==0
                 Sol(b)=S1;
              end
          end
   end

   Meca.SolAcel=Sol;

end
```

CONDPRISSOLV.M

```
function [Meca] = CondPrisSolV(Meca)
%% Función que implementa las condiciones prismáticas Wi=Wj en la solución%%

   RCP=Meca.RCP-1;

   % Restrinción en caso de bucle abierto [Pruebas]
      if Meca.Bucle==1
          RCP(end,:)=[];
      end
   %

   Sol=Meca.SolVeloc;
   for i=[1:1:length(RCP(:,1))]
        a=RCP(i,1);
        b=RCP(i,2);
          if a==0
              Sol(b)=0;
          end
          if b==0
             Sol(a)=0;
          end
   end
   RCP;
   for i=[1:1:length(RCP(:,1))]
       a=RCP(i,1);
       b=RCP(i,2);
          if a~=0 && b~=0
              S1=Sol(a);
              S2=Sol(b);
              if S1==0 && S2==0
              end
              if S1==0 && S2~=0
                  Sol(a)=S2;
              end
              if S1~=0 && S2==0
                 Sol(b)=S1;
              end
          end
   end

   Meca.SolVeloc=Sol;

end
```

CONDVGDL.M

```
function [Meca] = CondVGDL(Meca)
%% Función que aplica los GDL y crea las matrices Meca.Qdep y Meca.Qdep %%
   % Velocidades %
   Qdep=Meca.M;
   for i=[1:1:length(Meca.VGDL)]
       if Meca.VGDL(i)==0
       else
           Qdep(:,i)=0;
       end
   end
   Meca.Qdep=Qdep;

   Qind=Meca.M*Meca.VGDL;
   Meca.Qind=Qind;

end
```

CREACHAR.M

```
function [Meca] = CreaChar(Meca)
%% Función que crea dos caenas char con las velocidades iniciales a
introducir %%
% Meca.GDL.CharV para las velocidades
% Meca.GDL.CharA para las aceleraciones

V='[w2';
A='[a2';
 for i=[3:1:Meca.Nbody]
     f=num2str(i);
     V=[V ' w'];
     V=[V f];

     A=[A ' a'];
     A=[A f];
 end
  Meca.GDL.CharV.ang=V;
  Meca.GDL.CharA.ang=A;

  V=[];
  A=[];
 if Meca.NPP>=2
     V=[];
     A=[];
     Linea=Meca.RCP(:,1);
     for j=[1:1:length(Linea)]
         if Linea(j)==1
         else
             f=Linea(j)+1;
             f=num2str(f);
             V=[V ' Vx'];
             V=[V f];
             V=[V ' Vy'];
             V=[V f];

             A=[A ' Ax'];
             A=[A f];
             A=[A ' Ay'];
             A=[A f];
         end

     end
     V=[V ']'];
     A=[A ']'];

 else
    V=[V ']'];
    A=[A ']'];
 end

 Meca.GDL.CharV.lin=V;
 Meca.GDL.CharA.lin=A;
end
```

CREAGRUPOS.M

```
function [Meca] = CreaGrupos(Meca)
%% Función que genera los grupos para la matriz TH %%

   % Caso en el que no existan pares prismáticos o exista solo uno
     if Meca.NPP<=1
         Meca.Grupo(1).G=[1:1:Meca.Nrest];
     end
   %

%    % Caso en el que exisitan más de un par prismatico
%       if Meca.NPP>=2
%           G=[];
%           k=1;
%           for i=[1:1:length(Meca.LazoPares)]
%               G=[G,i];
%               if Meca.LazoPares(i)==2
%                   Meca.Grupo(k).G=G;
%                   k=k+1;
%                   G=[];
%               end
%           end
%       end
%    %

   % Caso en el que el mecanismo empiece o acabe con un par prismático
%        LazoModif=Meca.LazoPares;
%        if Meca.LazoPares(1)==2
%            LazoModif(1)=1;
%        end
%        LazoModif=[LazoModif,LazoModif(1)];
%        G=[];
%        k=1;
%           for i=[1:1:length(LazoModif)-1]
%              G=[G,i];
%               if LazoModif(i+1)==2
%                   Meca.Grupo(k).G=G;
%                   k=k+1;
%                   G=[];
%               end
%           end
   %

   if Meca.NPP>=2
       G=[];
       k=1;
       cont=0;

         for i=[1:1:length(Meca.LazoPares)-1]
            G=[G,i];
              if Meca.LazoPares(i)==2
                cont=cont+1;
              end
              if Meca.LazoPares(i+1)==2 && cont==1
                 Meca.Grupo(k).G=G;
                cont=0;
                k=k+1;
                G=[];
```

```
                end
                if cont==2
                    Meca.Grupo(k).G=G;
                    cont=0;
                    k=k+1;
                    G=[];
                end
                if i+1==length(Meca.LazoPares)
                    G=[G,i+1];
                    Meca.Grupo(k).G=G;
                    cont=0;
                    k=k+1;
                    G=[];
                end
            end

%   if Meca.Lazo(end)~=1
%       Meca.Grupo(end).G=[Meca.Grupo(end).G Meca.Lazo(end)];
%   end


    end

end
```

CREAINFORME.M

```
%% Informe del mecanismo
function [] = CreaInforme()

figure
Logo=imread('LogoInforme.jpg');
imshow(Logo)

 Fecha=date;


 Meca=evalin('base','Meca');
     if isempty(Meca.Nombre)
       Meca.Nombre='Mecanismo';
     end

 ListaPP=[];
 ListaPRSD=[];
  for i=[1:1:length(Meca.LazoPares)]
      if Meca.LazoPares(i)==2
          ListaPP=[ListaPP;i];
      end
      if Meca.LazoPares(i)==3
          ListaPRSD=[ListaPRSD;i];
      end
  end


%% Nombre del mecanismo
   fprintf('%s                          %s \n\n',Meca.Nombre,Fecha);
%% Datos del mecanismo
% Datos básicos
   fprintf('Número de eslabones: %d  \t;   Número de pares de restricción:
%d\n',Meca.Nbody,Meca.Nrest);
%%
% Datos de los pares
   fprintf('Pares de revolución: %d   \t;   Pares prismáticos:
%d',Meca.NPR,Meca.NPP);
%%
% Lazo del mecanismo
 fprintf('                {');
   for i=[1:1:length(Meca.Lazo)]
       fprintf(' %d',Meca.Lazo(i));
   end
 fprintf(' }');
%%
% Lazo de los pares del mecanismo
 fprintf('                {');
   for i=[1:1:length(Meca.LazoPares)]
       fprintf(' %d',Meca.LazoPares(i));
   end
 fprintf(' }');
fprintf('\n\n------------ Codificación de los pares ------------');

  fprintf('\n\n   Par de revolución: 1  \t;  Par prismático: 2');
  fprintf('\n      Par de rodadura sin deslizamiento: 3\n');

fprintf('\n--------------------------------------------------');
```

```
%% Geometría del mecanismo
% Coordenadas de los pares [X,Y]
 fprintf('\n')
   for i=[1:1:length(Meca.Coord(:,1))]
      fprintf('Par %d : [ %d ; %d ]\n',i,Meca.Coord(i,1),Meca.Coord(i,2));
   end
 fprintf('\n');
%%
% Direcciones de los pares prismáticos
if Meca.NPP==0
    fprintf('\n <<El mecanismo no posee pares prismáticos>>');
else
    fprintf('\n')
      for i=[1:1:length(Meca.DirecPP(:,1))]
         fprintf('Par prismático %d : \n',i);
         u=ListaPP(i);
         C1=Meca.Rela(u).Cuerpos(1);
         C2=Meca.Rela(u).Cuerpos(2);
         fprintf('   Eslabón %d  y  Eslabón %d\n',C1,C2);
         fprintf('Dirección del par: [ %d ; %d
]',Meca.DirecPP(i,1),Meca.DirecPP(i,2));
         if i==length(Meca.DirecPP(:,1))
         else
            fprintf('\n-----------------\n');
         end
      end
end
    fprintf('\n');
%%
% Direciones de los pares de rodadura sin deslizamiento
if Meca.NPRSD==0
    fprintf('\n <<El mecanismo no posee pares de rodadura sin
deslizamiento>>');
else
    fprintf('\n')
      for i=[1:1:length(Meca.DirecPRSD(:,1))]
         fprintf('Par de rodadura sin deslizamiento %d :\n',i);
         u=ListaPRSD(i);
         C1=Meca.Rela(u).Cuerpos(1);
         C2=Meca.Rela(u).Cuerpos(2);
         fprintf('   Eslabón %d  y  Eslabón %d\n',C1,C2);
         fprintf('Dirección del par: [ %d ; %d
]',Meca.DirecPRSD(i,1),Meca.DirecPRSD(i,2));
         if i==length(Meca.DirecPRSD)
         else
            fprintf('\n-----------------\n');
         end
      end
end
    fprintf('\n');
%%
% Radios de curvatura de los pares de rodadura sin deslizamiento
if Meca.NPRSD==0
    fprintf('\n <<El mecanismo no posee pares de rodadura sin
deslizamiento>>');
else
    fprintf('\n')
      for i=[1:1:length(Meca.Radios(:,1))]
         fprintf('Par de rodadura sin deslizamiento %d :\n',i);
         u=ListaPRSD(i);
```

```
         C1=Meca.Rela(u).Cuerpos(1);
         C2=Meca.Rela(u).Cuerpos(2);
         fprintf('    Radio de curvatura eslabón %d : %d     ;     Radio
curvatura eslabón %d : %d',C1,Meca.Radios(i,1),C2,Meca.Radios(i,2))
         if i==length(Meca.DirecPRSD)
         else
            fprintf('\n-----------------\n');
         end
      end
end
    fprintf('\n');

%%
% Tipología de los pares de rodadura sin deslizamiento
if Meca.NPRSD==0
    fprintf('\n <<El mecanismo no posee pares de rodadura sin
deslizamiento>>');
else
    fprintf('\n')
      for i=[1:1:length(Meca.Radios(:,1))]
         fprintf('Par de rodadura sin deslizamiento %d :\n',i);
         u=ListaPRSD(i);
         C1=Meca.Rela(u).Cuerpos(1);
         C2=Meca.Rela(u).Cuerpos(2);
         fprintf('   Eslabón %d  y  Eslabón %d\n',C1,C2);
         fprintf('   Tipología %d : ',Meca.Radios(i,3));
            if Meca.Radios(i,3)==1
                fprintf('Cóncavo - Convexo');
            end
            if Meca.Radios(i,3)==2
                fprintf('Convexo - Cóncavo');
            end
            if Meca.Radios(i,3)==3
                fprintf('Rueda - Plano');
            end
            if Meca.Radios(i,3)==4
                fprintf('Plano - Rueda');
            end
         if i==length(Meca.DirecPRSD)
         else
            fprintf('\n-----------------\n');
         end
      end
end
    fprintf('\n');

%% Velocidades y acelecaciones iniciales
% Velocidades iniciales
FV=[Meca.GDL.CharV.ang Meca.GDL.CharV.lin];
fprintf('\n %s : [',FV);
   for i=[1:1:length(Meca.VGDL)]
       fprintf(' %d',Meca.VGDL(i));
   end
 fprintf(' ] um/s \n');

%%
% Acelecaciones iniciales
FA=[Meca.GDL.CharA.ang Meca.GDL.CharA.lin];
fprintf('\n %s : [',FA);
   for i=[1:1:length(Meca.AGDL)]
       fprintf(' %d',Meca.AGDL(i));
   end
 fprintf(' ] um/s^2 \n');
```

```
%% Resultados
% Velocidades y aceleraciones lineales
if Meca.Prob==2
fprintf('\n');
   for i=[1:1:length(Meca.Rela)]
       fprintf('Relación %d : \n',i);
       fprintf('    Eslabón %d  y Eslabón
%d\n',Meca.Rela(i).Cuerpos(1),Meca.Rela(i).Cuerpos(2));
       fprintf('    Coordenadas : [ %d ; %d
]\n',Meca.Rela(i).x,Meca.Rela(i).y);
       fprintf('    Velocidad lineal : [ %d ; %d ]
um/s\n',Meca.Rela(i).Vx,Meca.Rela(i).Vy);
       fprintf('    Acelecarión lineal : [ %d ; %d ]
um/s^2\n',Meca.Rela(i).Ax,Meca.Rela(i).Ay);
          if i==length(Meca.Rela)
          else
              fprintf('\n-----------------\n');
          end
   end
fprintf('\n');
end
if Meca.Prob==1
fprintf('\n');
   for i=[1:1:length(Meca.Rela)]
       fprintf('Relación %d : \n',i);
       fprintf('    Eslabón %d  y Eslabón
%d\n',Meca.Rela(i).Cuerpos(1),Meca.Rela(i).Cuerpos(2));
       fprintf('    Coordenadas : [ %d ; %d
]\n',Meca.Rela(i).x,Meca.Rela(i).y);
       fprintf('    Velocidad lineal : [ %d ; %d ]
um/s\n',Meca.Rela(i).Vx,Meca.Rela(i).Vy);
          if i==length(Meca.Rela)
          else
              fprintf('\n-----------------\n');
          end
   end
fprintf('\n');
end

%%
% Velocidades y aceleraciones angulares
if Meca.Prob==2
fprintf('\n');
   for i=[1:1:length(Meca.Cuerpo)]
       fprintf('Eslabón %d : \n',i);
       fprintf('    Velocidad angular : %d rad/s\n',Meca.Cuerpo(i).W);
       fprintf('    Acelecarión angular : %d rad/s^2\n',Meca.Cuerpo(i).A);
          if i==length(Meca.Rela)
          else
              fprintf('\n-----------------\n');
          end
   end
fprintf('\n');
end
if Meca.Prob==1
fprintf('\n');
   for i=[1:1:length(Meca.Cuerpo)]
       fprintf('Eslabón %d : \n',i);
       fprintf('    Velocidad angular : %d rad/s\n',Meca.Cuerpo(i).W);
          if i==length(Meca.Rela)
```

```
          else
              fprintf('\n-----------------\n');
          end
   end
fprintf('\n');
end


%% Comprobaciones de los resultados
fprintf('\nCAC 2D comprueba los resultados obtenidos mediante \nel método de
cálculo con un error inferior a %d. \nSi se detectase algún problema en los
valores obtenidos \nse comunicará un posible fallo.',10^-5);
%%
% Comprobación del problema de velocidad
   aux=sum(abs(Meca.CompruebaVel));
   ErrorV=log10(aux);
      if ErrorV==-Inf
          fprintf('\n Problema de velocidad correcto\n    con error 0\n')
      else
           if aux<10^-5
              fprintf('\n Problema de velociad correcto\n    con un errror
del 10^%d\n',ErrorV)
           else
              fprintf('\n Problema de velocidad\n    recuelto con algun tipo
de error\n')
           end
      end

%%
if Meca.Prob==2
% Comprobación del problema de aceleración
   aux=sum(abs(Meca.CompruebaAce));
   ErrorA=log10(aux);
      if ErrorA==-Inf
          fprintf('\n Problema de aceleración correcto\n    con error 0\n')
      else
           if aux<10^-5
              fprintf('\n Problema de aceleración correcto\n    con un errror
del 10^%d\n',ErrorA)
           else
              fprintf('\n Problema de aceleración\n    recuelto con algun
tipo de error\n')
           end
      end
end
%% Esquema gráfico del mecanismo
figure
    axis auto
    title('Mecanismo en la posición dada')
    % Muestreo de los pares del mecanismo
       MP=MuestraPares(Meca);
       if MP==1
       else
           fprintf('\n --> >>ERROR<< Muestreo de los pares\n')
       end
    %
    % Muestreo de los cuerpos del mecanismo
       MC=MuestraCuerpos(Meca);
       if MC==1
       else
           fprintf('\n --> >>ERROR<< Muestreo de los cuerpos\n')
       end
    %
```

```
%% Distribución de velocidades y aceleraciones
    axis auto
    title('Mecanismo en la posición dada')
    % Muestreo de los pares del mecanismo
       MP=MuestraPares(Meca);
       if MP==1
       else
           fprintf('\n --> >>ERROR<< Muestreo de los pares\n')
       end
    %
    % Muestreo de los cuerpos del mecanismo
       MC=MuestraCuerpos(Meca);
       if MC==1
       else
           fprintf('\n --> >>ERROR<< Muestreo de los cuerpos\n')
       end
    %
    % Muestreo de las velocidades de cada Par
       MV=MuestraVelocidades(Meca);
       if MV==1
       else
           fprintf('\n --> >>ERROR<<  Muestreo de las velocidades\n')
       end
    %
    % Muestreo de las velocidades de cada Par
    if Meca.Prob==2
       MA=MuestraAceleraciones(Meca);
       if MA==1
       else
           fprintf('\n --> >>ERROR<<  Muestreo de las Aceleraciones\n')
       end
    end
    %

%%
fprintf('\n\n\n\n\n');

close
close
close
end
```

CREATRAMOS.M

```
function [Meca] = CreaTramos(Meca)
%% Función que genera los tramos a partir de las coordenadas - Meca.Tramo().T
%%

   for i=[1:1:Meca.Nrest-1]
       fin=[Meca.Rela(i+1).x;Meca.Rela(i+1).y];
       ini=[Meca.Rela(i).x;Meca.Rela(i).y];
       V=fin-ini;
       Meca.Tramo(i).T=V;
   end

   % Último tramo
          fin=[Meca.Rela(1).x;Meca.Rela(1).y];
          ini=[Meca.Rela(end).x;Meca.Rela(end).y];
          V=fin-ini;
          Meca.Tramo(Meca.Nrest).T=V;
   %
   % Si es un bucle abierto
%        if Meca.Lazo(end)~=1
%            u=Meca.Lazo(end);
%            Meca.Tramo(u).T=[0;0];
%        end
   %
end
```

ENTRADA.M

```
function varargout = Entrada(varargin)
% ENTRADA MATLAB code for Entrada.fig
%      ENTRADA, by itself, creates a new ENTRADA or raises the existing
%      singleton*.
%
%      H = ENTRADA returns the handle to a new ENTRADA or the handle to
%      the existing singleton*.
%
%      ENTRADA('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ENTRADA.M with the given input arguments.
%
%      ENTRADA('Property','Value',...) creates a new ENTRADA or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Entrada_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Entrada_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Entrada

% Last Modified by GUIDE v2.5 19-Jun-2016 19:29:15

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Entrada_OpeningFcn, ...
                   'gui_OutputFcn',  @Entrada_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Entrada is made visible.
function Entrada_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Entrada (see VARARGIN)

% Choose default command line output for Entrada
```

```
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

Logo=imread('Logo1.png');
axes(handles.Logo);
imshow(Logo)
axis off

LogoETSI=imread('LogoESI.jpg');
axes(handles.LogoETSI);
imshow(LogoETSI)
axis off

LogoUS=imread('LogoUS.jpg');
axes(handles.LogoUS);
imshow(LogoUS)
axis off

% UIWAIT makes Entrada wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Entrada_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in Entrar.
function Entrar_Callback(hObject, eventdata, handles)
% hObject    handle to Entrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over Salir.
Frase='Iniciando ...';
set(handles.Entrar,'String',Frase);
Condiciones_de_uso()
% pause(5)
close(Entrada)




% --- Executes on button press in Salir.
function Salir_Callback(hObject, eventdata, handles)
% hObject    handle to Salir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close
```

GUARDA_ARCHIVOSM

```
function [s] = Guarda_Archivos(Save)
%% Función que guarda los datos en la carpeta Archivos guardados %%

s=0;

W=what;
DirectorioA=W.path;
DirectorioN=[DirectorioA '\Archivos guardados'];
DirectorioN=[DirectorioN '\'];

cd(DirectorioN);
if isempty(Save.Nombre)
   mkdir([DirectorioN,'Mecanismo']);
   DirectorioN=[DirectorioN '\'];
   DirectorioN=[DirectorioN 'Mecanismo'];
else
   mkdir([DirectorioN,Save.Nombre]);
   DirectorioN=[DirectorioN '\'];
   DirectorioN=[DirectorioN Save.Nombre];
end

   cd(DirectorioN)

     ANbody=fopen('Nbody.txt','w');
     fprintf(ANbody,'%d',Save.Nbody);

     ANrest=fopen('Nrest.txt','w');
     fprintf(ANrest,'%d',Save.Nrest);

     ANPR=fopen('NPR.txt','w');
     fprintf(ANPR,'%d',Save.NPR);

     ANPP=fopen('NPP.txt','w');
     fprintf(ANPP,'%d',Save.NPP);

     ANPRSD=fopen('NPRSD.txt','w');
     fprintf(ANPRSD,'%d',Save.NPRSD);

     AVGDL=fopen('VGDL.txt','w');
       fprintf(AVGDL,'[');
       for i=[1:1:length(Save.VGDL)]
           if i==length(Save.VGDL)
              fprintf(AVGDL,'%d',Save.VGDL(i));
              fprintf(AVGDL,']');
           else
              fprintf(AVGDL,'%d',Save.VGDL(i));
              fprintf(AVGDL,' ');
           end
       end

     AAGDL=fopen('AGDL.txt','w');
       fprintf(AAGDL,'[');
       for i=[1:1:length(Save.AGDL)]
           if i==length(Save.AGDL)
              fprintf(AAGDL,'%d',Save.AGDL(i));
```

```
            fprintf(AAGDL,']');
        else
            fprintf(AAGDL,'%d',Save.AGDL(i));
            fprintf(AAGDL,' ');
        end
    end

    ALazo=fopen('Lazo.txt','w');
    fprintf(ALazo,'[');
    for i=[1:1:length(Save.Lazo)]
        if i==length(Save.Lazo)
            fprintf(ALazo,'%d',Save.Lazo(i));
            fprintf(ALazo,']');
        else
            fprintf(ALazo,'%d',Save.Lazo(i));
            fprintf(ALazo,' ');
        end
    end

    ALazoPares=fopen('LazoPares.txt','w');
    fprintf(ALazoPares,'[');
    for i=[1:1:length(Save.LazoPares)]
        if i==length(Save.LazoPares)
            fprintf(ALazoPares,'%d',Save.LazoPares(i));
            fprintf(ALazoPares,']');
        else
            fprintf(ALazoPares,'%d',Save.LazoPares(i));
            fprintf(ALazoPares,' ');
        end
    end

    ACoord=fopen('Coord.txt','wt');
    J=[];
    k=1;
%     for i=[1:1:100]
%         if isnan(Save.Coord(i,1)) || isnan(Save.Coord(i,2))
%             k=k+1;
%         end
%         M(i,1)=Save.Coord(i,1);
%         M(i,2)=Save.Coord(i,2);
%     end
%     if k==101
%         fprintf(ACoord,'');
%     else
%        for i=[1:1:100-k+1]
%            J(i,1)=double(M(i,1));
%            J(i,2)=double(M(i,2));
%        end
J=Save.Coord;
    if isempty(J)
        fprintf(ACoord,'');
    else
        for i=[1:1:length(J(:,1))]
            p=double(J(i,1));
            s=double(J(i,2));
            fprintf(ACoord,'%d ',p);
            fprintf(ACoord,'%d',s);
            fprintf(ACoord,'\n');
        end
    end

    ADirecPRSD=fopen('DirecPRSD.txt','wt');
```

```
      J=[];
       k=1;
%       for i=[1:1:100]
%           if isnan(Save.DirecPRSD(i,3))
%               k=k+1;
%           end
%           M(i,1)=Save.DirecPRSD(i,1);
%           M(i,2)=Save.DirecPRSD(i,2);
%           M(i,3)=Save.DirecPRSD(i,3);
%       end
%       if k==101
%           fprintf(ADirecPRSD,'');
%       else
%          for i=[1:1:100-k+1]
%              J(i,1)=double(M(i,1));
%              J(i,2)=double(M(i,2));
%              J(i,3)=double(M(i,3));
%          end
J=Save.DirecPRSD;
    if isempty(J)
        fprintf(ADirecPRSD,'');
    else
         for i=[1:1:length(J(:,1))]
             p=double(J(i,1));
             s=double(J(i,2));
             t=double(J(i,3));
             fprintf(ADirecPRSD,'%d ',p);
             fprintf(ADirecPRSD,'%d ',s);
             fprintf(ADirecPRSD,'%d',t);
             fprintf(ADirecPRSD,'\n');
         end
    end

      ARadios=fopen('Radios.txt','wt');
      J=[];
      k=1;
%       for i=[1:1:100]
%           if isnan(Save.Radios(i,1)) || isnan(Save.Radios(i,2))
%               k=k+1;
%           end
%           M(i,1)=Save.Radios(i,1);
%           M(i,2)=Save.Radios(i,2);
%       end
%       if k==101
%           fprintf(ARadios,'');
%       else
%          for i=[1:1:100-k+1]
%              J(i,1)=double(M(i,1));
%              J(i,2)=double(M(i,2));
%          end
J=Save.Radios;
    if isempty(J)
        fprintf(ARadios,'');
    else
         for i=[1:1:length(J(:,1))]
             p=double(J(i,1));
             s=double(J(i,2));
             fprintf(ARadios,'%d ',p);
             fprintf(ARadios,'%d',s);
             fprintf(ARadios,'\n');
         end
```

```
   end


     ADirecPP=fopen('DirecPP.txt','wt');
     J=[];
     k=1;
%      for i=[1:1:100]
%          if isnan(Save.DirecPP(i,1)) || isnan(Save.DirecPP(i,2))
%              k=k+1;
%          end
%          M(i,1)=Save.DirecPP(i,1);
%          M(i,2)=Save.DirecPP(i,2);
%      end
%      if k==101
%          fprintf(ADirecPP,'');
%      else
%         for i=[1:1:100-k+1]
%             J(i,1)=double(M(i,1));
%             J(i,2)=double(M(i,2));
%         end
J=Save.DirecPP;
   if isempty(J)
       fprintf(ADirecPP,'');
   else
        for i=[1:1:length(J(:,1))]
            p=double(J(i,1));
            s=double(J(i,2));
            fprintf(ADirecPP,'%d ',p);
            fprintf(ADirecPP,'%d',s);
            fprintf(ADirecPP,'\n');
        end
   end


 fclose all;


cd(DirectorioA);
 s=1;

end
```

HAZDIBUJO.M

```
function [] = HazDibujo(Meca)
%% Función que representa el dibujo del mecanismo %%

%     axis([-15 15 -15 15])
    axis auto
    title('Mecanismo en la posición dada')
    % Muestreo de los pares del mecanismo
       MP=MuestraPares(Meca);
       if MP==1
       else
           fprintf('\n --> >>ERROR<< Muestreo de los pares\n')
       end
    %
    % Muestreo de los cuerpos del mecanismo
       MC=MuestraCuerpos(Meca);
       if MC==1
       else
           fprintf('\n --> >>ERROR<< Muestreo de los cuerpos\n')
       end
    %
    % Muestreo de las velocidades de cada Par
       MV=MuestraVelocidades(Meca);
       if MV==1
       else
           fprintf('\n --> >>ERROR<<  Muestreo de las velocidades\n')
       end
    %
    % Muestreo de las velocidades de cada Par
    if Meca.Prob==2
       MA=MuestraAceleraciones(Meca);
       if MA==1
       else
           fprintf('\n --> >>ERROR<<  Muestreo de las Aceleraciones\n')
       end
    end
    %

end
```

HAZSIMULACION.M

```
function [] = HazSimulacion(Meca,Vista)
%% Función que representa la simulación del mecanismo %%

    axis([Vista.Xmin Vista.Xmax Vista.Ymin Vista.Xmax])

%     axis auto
  title('Simulación en movimiento del mecanismo')
    Meca.ResTiempo=[];
    Meca.Error=0;
    for i=[1:1:Vista.Npasos]

        Meca.Iteracion=i;
       pause(0.05);
       cla
          % Muestreo de los pares del mecanismo
            MP=MuestraPares(Meca);
            if MP==1
            else
                fprintf('\n --> >>ERROR<< Muestreo de los pares\n')
            end
          %
          % Muestreo de los cuerpos del mecanismo
             MC=MuestraCuerpos(Meca);
             if MC==1
             else
                fprintf('\n --> >>ERROR<< Muestreo de los cuerpos\n')
             end
         %
         % Muestreo de las velocidades de cada Par
            MV=MuestraVelocidades(Meca);
            if MV==1
            else
              fprintf('\n --> >>ERROR<<  Muestreo de las velocidades\n')
            end
         %
         % Muestreo de las Aceleraciones de cada Par
            MA=MuestraAceleraciones(Meca);
              if MA==1
              else
                fprintf('\n --> >>ERROR<<  Muestreo de las Aceleraciones\n')
              end
         %
            Coord=[];
            Meca=NuevasCoord(Meca);
            for j=[1:1:Meca.Nrest]
               Coord=[Coord;Meca.Rela(j).x Meca.Rela(j).y];
            end

            Meca.ResTiempo=[Meca.ResTiempo;i
[Meca.Rela(3).Ax+Meca.Rela(3).Ay]];
            Meca=NuevasGDL(Meca);
            Meca.Coord=Coord;
            Meca=Calculador(Meca);

    end
end
```

LISTADOCUERPOS.M

```
function [ListadoC] = ListadoCuerpos(Meca)
%% Función que crea el listado lso cuerpos existentes en Meca

ListadoC='';
for i=[1:1:Meca.Nbody]
    Frase='Eslabón ';
    n=num2str(i);
    Frase=[Frase n];
    ListadoC=[ListadoC;Frase];
end

end
```

LISTADORELACIONES.M

```
function [ListadoR] = ListadoRelaciones(Meca)
%% Función que crea el listado de las relaciones existentes en Meca

ListadoR='';
for i=[1:1:length(Meca.Rela)]
    Frase='Relación ';
    n=num2str(i);
    Frase=[Frase n];
    ListadoR=[ListadoR;Frase];
end

end
```

MATRIZAH.M

```
function [Meca] = MatrizAH(Meca)
%% Programa que calcula la matriz AH relacionada con los términos de
% aceleraciones relativa de los pares de rodadura sin deslizamiento aij

AH=zeros(length(Meca.Grupo),1);
if Meca.NPP==0
  cont=1;
  P=[0;0];
    for i=[1:1:length(Meca.Grupo(1).G)]
        u=Meca.Grupo(1).G(i);
           if Meca.Rela(u).Codig==3
               wi=Meca.Cuerpo(u).W;
                  if u==Meca.Nbody
                      wj=0;
                  else
                      wj=Meca.Cuerpo(u+1).W;
                  end
                Ri=Meca.Radios(cont,1);
                Rj=Meca.Radios(cont,2);
                landa=Meca.DirecPRSD(cont,:);
                landa=(landa)./norm(landa);
                   if Meca.Radios(cont,3)==1
                        aij=((wi-wj).^2).*((Ri.*Rj)./(Ri-Rj)).*landa;
                   end
                   if Meca.Radios(cont,3)==2
                        aij=((wi-wj).^2).*((Ri.*Rj)./(Ri+Rj)).*landa;
                   end
                   if Meca.Radios(cont,3)==3
                        aij=((wi-wj).^2).*(Ri).*landa;
                   end
                   if Meca.Radios(cont,3)==4
                        aij=((wi-wj).^2).*(Rj).*landa;
                   end

                P=P+aij';
                cont=cont+1;
           else
               P=[0;0];
           end
    end
  AH=P;

else
   cont=1;
      for i=[1:1:length(Meca.Grupo)]
          for j=[1:1:length(Meca.Grupo(i).G)]
              u=Meca.Grupo(i).G(j);
              if Meca.LazoPares(u)==3
                  wi=Meca.Cuerpo(u).W;
                  if u==Meca.Nbody
                      wj=0;
                  else
                      wj=Meca.Cuerpo(u+1).W;
                  end
                  Ri=Meca.Radios(cont,1);
                  Rj=Meca.Radios(cont,2);
```

```
                landa=Meca.DirecPRSD(cont,:);
                landa=(landa)./norm(landa)';
                if Meca.Radios(cont,3)==1
                    aij=((wi-wj).^2).*((Ri.*Rj)./(Ri-Rj)).*landa;
                end
                if Meca.Radios(cont,3)==2
                    aij=((wi-wj).^2).*((Ri.*Rj)./(Ri+Rj)).*landa;
                end
                if Meca.Radios(cont,3)==3
                    aij=((wi-wj).^2).*(Ri).*landa;
                end
                if Meca.Radios(cont,3)==4
                    aij=((wi-wj).^2).*(Rj).*landa;
                end
                hp=Meca.DirecPP(i,:)';
                hp=[-hp(2);hp(1)];
                AH(i,1)=sum(aij'.*hp);
                cont=cont+1;
            end
        end
    end

end
  Meca.AH=AH;


end
```

MATRIZHV.M

```
function [Meca] = MatrizHV(Meca)
%% Función que genera la matriz HV %%

   % Determinación de las Vi de cada grupo
      MV=[];
      for i=[1:1:length(Meca.Grupo)]
          ini=Meca.Grupo(i).G(1);
          fin=Meca.Grupo(i).G(end);
%           if Meca.Lazo(end)~=1 && i==length(Meca.Grupo)
%              MV=[MV;ini,Meca.Lazo(fin)];
%           else
            MV=[MV;ini,Meca.Lazo(fin+1)];
%           end
      end
      Meca.MV=MV;
   %

   % Creación de la matriz HV
      linea=[1:1:Meca.NPP,1];
      n=length(Meca.MV(:,1));
      k=1;
      Maux=[];
      for i=[1:1:length(linea)-1]
          Laux=[linea(i),linea(i+1)];
          Maux=[Maux;Laux];
      end
      hv=zeros(Meca.NPP);
      for i=[1:1:length(Maux(:,1))]
         u=Maux(i,1);
         v=Maux(i,2);
         hv(i,u)=-1;
         hv(i,v)=1;
      end
      % Determinación de la submatriz Hpx
        Z=zeros(length(hv(:,1)),1);
        Hpx=[];
        VHpx=[];
         for i=[1:1:length(hv(1,:))]
             Hpx=[Hpx,hv(:,i)];
             Hpx=[Hpx,Z];
         end
         for i=[1:1:Meca.NPP]
             h=Meca.DirecPP(i,:);
             hp=[-h(2);h(1)];
             VHpx=[VHpx;hp(1)];
         end
         for i=[1:1:length(VHpx)]
             for j=[1:1:length(Hpx(1,:))]
                 Hpx(i,j)=Hpx(i,j).*VHpx(i);
             end
         end
      %
      % Determinación de la submatriz Hpy
        Hpy=Z;
        VHpy=[];
         for i=[1:1:length(hv(1,:))]
```

```
            Hpy=[Hpy,hv(:,i)];
            Hpy=[Hpy,Z];
        end
        Hpy(:,end)=[];
        for i=[1:1:Meca.NPP]
            h=Meca.DirecPP(i,:);
            hp=[-h(2);h(1)];
            VHpy=[VHpy;hp(2)];
        end
        for i=[1:1:length(VHpy)]
            for j=[1:1:length(Hpy(1,:))]
                Hpy(i,j)=Hpy(i,j).*VHpy(i);
            end
        end
    %
    HV=Hpx+Hpy;

    % Eliminación de las columnas relacionadas con V1
        HV(:,2)=[];
        HV(:,1)=[];
    %
    Meca.HV=HV;

  %

end
```

MATRIZTH.M

```
function [Meca] = MatrizTH(Meca)
%% Función que genera la Matriz TH %%

TH=zeros(length(Meca.Grupo),Meca.Nbody);

   for i=[1:1:length(Meca.Grupo)]
       for j=[1:1:length(Meca.Grupo(i).G)]
           u=Meca.Grupo(i).G(j);
           T=Meca.Tramo(u).T;
           T=[-T(2);T(1)];
           hp=Meca.DirecPP(i,:)';
           hp=[-hp(2);hp(1)];
           TH(i,u)=sum(T.*hp);
       end
   end
   TH(:,Meca.Nbody)=[];
   Meca.TH=TH;

end
```

MATRIZWTH.M

```
function [Meca] = MatrizWTH(Meca)
%% Programa que calcula la matriz WTH

WTH=zeros(length(Meca.Grupo),Meca.Nbody);
if Meca.NPP==0
  P=0;
    for i=[1:1:length(Meca.Grupo(1).G)]
        u=Meca.Grupo(1).G(i);
          if u==Meca.Nbody
              w=0;
          else
              w=Meca.Cuerpo(u+1).W;
          end
        T=Meca.Tramo(u).T;
        P=P+T.*((w).^2);
    end
   WTH=P;
   Meca.WTH=WTH;
else
     for i=[1:1:length(Meca.Grupo)]
         for j=[1:1:length(Meca.Grupo(i).G)]
             u=Meca.Grupo(i).G(j);
             T=Meca.Tramo(u).T;
             T=[T(1);T(2)];
             hp=Meca.DirecPP(i,:)';
             hp=[-hp(2);hp(1)];
             if u==Meca.Nbody
                 w=0;
             else
                 w=Meca.Cuerpo(u+1).W;
             end
             WTH(i,u)=sum(T.*hp).*(w.^2);
         end
     end
     WTH(:,Meca.Nbody)=[];
     Meca.WTH=sum(WTH')';
end

end
```

MATRIZWVH.M

```
function [Meca] = MatrizWVH(Meca)
%% Función que calcula la matriz WVH relacionada con el par de rodadura sin
deslizamiento

WVH=zeros(length(Meca.Grupo),1);

if Meca.NPP==0
  P=[0;0];
    for i=[1:1:length(Meca.Grupo(1).G)]
        u=Meca.Grupo(1).G(i);
           if Meca.Rela(u).Codig==3
                if u==Meca.Nbody
                    wj=0;
                else
                    wj=Meca.Cuerpo(u+1).W;
                end
              vi=[Meca.Rela(u).Vx;Meca.Rela(u).Vy];
              wv=2.*[-vi(2);vi(1)].*wj;
              P=P+(wv);
           else
               P=[0;0];
           end
    end
    WVH=P;

else
      for i=[1:1:length(Meca.Grupo)]
          for j=[1:1:length(Meca.Grupo(i).G)]
              u=Meca.Grupo(i).G(j);
              if Meca.LazoPares(u)==3
                  if u==Meca.Nbody
                      wj=0;
                  else
                      wj=Meca.Cuerpo(u+1).W;
                  end
                  vi=[Meca.Rela(u).Vx;Meca.Rela(u).Vy];
                  wv=[-vi(2);vi(1)].*wj;
                  hp=Meca.DirecPP(i,:)';
                  hp=[-hp(2);hp(1)];
                  WVH(i,1)=2.*sum(wv.*hp);
              end
          end
      end
end
   Meca.WVH=WVH;


end
```

MUESTRAACELERACIONES.M

```
function [MA] = MuestraAceleraciones(Meca)
%% Función que muestra en un plot las velocidades de los pares del mecanismo
%%

MA=0;
hold on
for i=[1:1:length(Meca.Rela)]
    x=Meca.Rela(i).x;
    y=Meca.Rela(i).y;
    Ax=Meca.Rela(i).Ax;
    Ay=Meca.Rela(i).Ay;
    P=[x;y];
    A=[Ax;Ay];
    A=A./norm(A);
    Pfin=A+P;
    quiver3(P(1),P(2),0,Pfin(1)-P(1),Pfin(2)-
P(2),0,'g','linewidth',3,'MaxHeadSize',0.3)
end
% x0=0;
% y0=0;
% z0=0;
%
% x1=1;
% y1=2;
% z1=0;
% % axis([0 2 0 5 0 0 1 5]);
% hold on
% quiver3(x0,y0,z0,x1-x0,y1-y0,z1-z0,'r','linewidth',3,'MaxHeadSize',0.3)

MA=1;
end
```

MUESTRACUERPOS.M

```
function [MC] = MuestraCuerpos(Meca)
%% Función que muestra en un plot los cuerpos del mecanismo %%
   MC=0;
  for i=[1:1:length(Meca.Rela)-1]
      Pini=[Meca.Rela(i).x;Meca.Rela(i).y];
      Pfin=[Meca.Rela(i+1).x;Meca.Rela(i+1).y];
      line([Pini(1),Pfin(1)],[Pini(2),Pfin(2)],'Color','k','LineWidth',2)
      Cuerpo=Meca.Rela(i).Cuerpos(2);
      P=(Pini+Pfin)./2;
      P=P+[0.05;0.05];
      C=num2str(Cuerpo);
      text(P(1),P(2),C)
  end

  MC=1;
end
```

MUESTRAPARES.M

```
function [MP] = MuestraPares(Meca)
%% Función que muestra en un plot los pares del mecanismo %%

   MP=0;
   hold on
   g=[0.2;0];
   e=[0;0.1];
   k=1;
   for i=[1:1:length(Meca.Rela)]
       if Meca.Rela(i).Codig==1
           plot(Meca.Rela(i).x,Meca.Rela(i).y,'ob')
       end
       if Meca.Rela(i).Codig==2
           if i==length(Meca.Rela) && Meca.Bucle==1
           else
           x=Meca.Rela(i).x;
           y=Meca.Rela(i).y;
           P=[x;y];
           hp=Meca.Rela(i).hp;
           h=[-hp(2);hp(1)];
           hu=h./(norm(h));
           hpu=hp./norm(hp);
           P1=P-sum(g.*h).*hu+sum(e.*hp).*hpu;
           P2=P+sum(g.*h).*hu+sum(e.*hp).*hpu;
           P3=P+sum(g.*h).*hu-sum(e.*hp).*hpu;
           P4=P-sum(g.*h).*hu-sum(e.*hp).*hpu;
           line([P1(1),P2(1)],[P1(2),P2(2)],'LineWidth',2)
           line([P2(1),P3(1)],[P2(2),P3(2)],'LineWidth',2)
           line([P3(1),P4(1)],[P3(2),P4(2)],'LineWidth',2)
           line([P4(1),P1(1)],[P4(2),P1(2)],'LineWidth',2)
           end
       end
       if Meca.Rela(i).Codig==3
%            plot(Meca.Rela(i).x,Meca.Rela(i).y,'^b')
          Tipo=Meca.Radios(k,3);
          landa=Meca.DirecPRSD(k,:);
          landaT=([-landa(2) landa(1)])';
          landaN=((landa)./(norm(landa)))';

          %
            if Tipo==1
                R1=Meca.Radios(k,1);
                R2=Meca.Radios(k,2);

                P1=[Meca.Rela(i).x;Meca.Rela(i).y]+landaN.*R1;
                P2=[Meca.Rela(i).x;Meca.Rela(i).y]-landaN.*R2;
               t=linspace(0,2*pi,300);
               x1=R1*cos(t)+P1(1);
               y1=R1*sin(t)+P1(2);
               x2=R2*cos(t)+P2(1);
               y2=R2*sin(t)+P2(2);
               plot(x1,y1,'b','LineWidth',2)
               plot(x2,y2,'b','LineWidth',2)
            end
            if Tipo==2
                R1=Meca.Radios(k,1);
                R2=Meca.Radios(k,2);
```

```
                P1=[Meca.Rela(i).x;Meca.Rela(i).y]+landaN.*R1;
                P2=[Meca.Rela(i).x;Meca.Rela(i).y]+landaN.*R2;
               t=linspace(0,2*pi,300);
               x1=R1*cos(t)+P1(1);
               y1=R1*sin(t)+P1(2);
               x2=R2*cos(t)+P2(1);
               y2=R2*sin(t)+P2(2);
               plot(x1,y1,'b','LineWidth',2)
               plot(x2,y2,'b','LineWidth',2)
            end
            if Tipo==3
                R1=Meca.Radios(k,1);

                P1=[Meca.Rela(i).x;Meca.Rela(i).y]+landaN.*R1;
               t=linspace(0,2*pi,300);
               x1=R1*cos(t)+P1(1);
               y1=R1*sin(t)+P1(2);
               direcLanda=landaT./norm(landaT);
               Q1=[Meca.Rela(i).x;Meca.Rela(i).y]+direcLanda.*(R1/2);
               Q2=[Meca.Rela(i).x;Meca.Rela(i).y]-direcLanda.*(R1/2);
               plot(x1,y1,'b','LineWidth',2);
               line([Q1(1),Q2(1)],[Q1(2),Q2(2)],'LineWidth',2)
            end
            if Tipo==4
                R2=Meca.Radios(k,2);

                P1=[Meca.Rela(i).x;Meca.Rela(i).y]-landaN.*R2;
               t=linspace(0,2*pi,300);
               x1=R2*cos(t)+P1(1);
               y1=R2*sin(t)+P1(2);
               direcLanda=landaT./norm(landaT);
               Q1=[Meca.Rela(i).x;Meca.Rela(i).y]+direcLanda.*(R2/2);
               Q2=[Meca.Rela(i).x;Meca.Rela(i).y]-direcLanda.*(R2/2);
               plot(x1,y1,'b','LineWidth',2);
               line([Q1(1),Q2(1)],[Q1(2),Q2(2)],'LineWidth',2)
            end
          %

%           if Tipo==1 || Tipo==2
%            R1=Meca.Radios(k,1);
%            R2=Meca.Radios(k,2);
%            if i==1
%                x1=Meca.Rela(1).x;
%                y1=Meca.Rela(1).y;
%            else
%                x1=Meca.Rela(i-1).x;
%                y1=Meca.Rela(i-1).y;
%            end
%            if i==length(Meca.Rela)
%                x2=Meca.Rela(length(Meca.Rela)).x;
%                y2=Meca.Rela(length(Meca.Rela)).y;
%            else
%                x2=Meca.Rela(i+1).x;
%                y2=Meca.Rela(i+1).y;
%            end
%            t=linspace(0,2*pi,300);
%            x1=R1*cos(t)+x1;
%            y1=R1*sin(t)+y1;
%            x2=R2*cos(t)+x2;
%            y2=R2*sin(t)+y2;
%            plot(x1,y1,'b','LineWidth',2)
```

```
%             plot(x2,y2,'b','LineWidth',2)
%             end
%
%             if Tipo==3
%                 R1=Meca.Radios(k,1);
%                 t=linspace(0,2*pi,300);
%                 if i==1
%                     x1=Meca.Rela(1).x;
%                     y1=Meca.Rela(1).y;
%                 else
%                     x1=Meca.Rela(i-1).x;
%                     y1=Meca.Rela(i-1).y;
%                 end
%                 if i==length(Meca.Rela)
%                     x2=Meca.Rela(length(Meca.Rela)).x;
%                     y2=Meca.Rela(length(Meca.Rela)).y;
%                 else
%                     x2=Meca.Rela(i+1).x;
%                     y2=Meca.Rela(i+1).y;
%                 end
%
%                 x1=R1*cos(t)+x1;
%                 y1=R1*sin(t)+y1;
%                 direcLanda=landa./norm(landa);
%                 P1=[x2;y2]+direcLanda';
%                 P2=[x2;y2]-direcLanda';
%                 plot(x1,y1,'b','LineWidth',2);
%                 line([P1(1),P2(1)],[P1(2),P2(2)],'LineWidth',2)
%             end
%
%             if Tipo==4
%                 R2=Meca.Radios(k,2);
%                 t=linspace(0,2*pi,300);
%                 if i==1
%                     x1=Meca.Rela(1).x;
%                     y1=Meca.Rela(1).y;
%                 else
%                     x1=Meca.Rela(i-1).x;
%                     y1=Meca.Rela(i-1).y;
%                 end
%                 if i==length(Meca.Rela)
%                     x2=Meca.Rela(length(Meca.Rela)).x;
%                     y2=Meca.Rela(length(Meca.Rela)).y;
%                 else
%                     x2=Meca.Rela(i+1).x;
%                     y2=Meca.Rela(i+1).y;
%                 end
%
%                 x2=R2*cos(t)+x2;
%                 y2=R2*sin(t)+y2;
%                 direcLanda=landa./norm(landa);
%                 P1=[x1;y1]+direcLanda';
%                 P2=[x1;y1]-direcLanda';
%                 plot(x2,y2,'b','LineWidth',2);
%                 line([P1(1),P2(1)],[P1(2),P2(2)],'LineWidth',2)
%             end

            k=k+1;
        end
    end
    MP=1;

end
```

MUESTRARESULTADOS.M

```
function varargout = MuestraResultados(varargin)
% MUESTRARESULTADOS MATLAB code for MuestraResultados.fig
%      MUESTRARESULTADOS, by itself, creates a new MUESTRARESULTADOS or
raises the existing
%      singleton*.
%
%      H = MUESTRARESULTADOS returns the handle to a new MUESTRARESULTADOS or
the handle to
%      the existing singleton*.
%
%      MUESTRARESULTADOS('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in MUESTRARESULTADOS.M with the given input
arguments.
%
%      MUESTRARESULTADOS('Property','Value',...) creates a new
MUESTRARESULTADOS or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before MuestraResultados_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to MuestraResultados_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help MuestraResultados

% Last Modified by GUIDE v2.5 20-Jul-2016 02:48:50

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @MuestraResultados_OpeningFcn, ...
                   'gui_OutputFcn',  @MuestraResultados_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before MuestraResultados is made visible.
function MuestraResultados_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
```

```
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to MuestraResultados (see VARARGIN)

% Choose default command line output for MuestraResultados
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes MuestraResultados wait for user response (see UIRESUME)
% uiwait(handles.figure1);

Logo=imread('Logo1.png');
axes(handles.Logo);
image(Logo)
axis off
set(handles.CompA,'Visible','off');
set(handles.CompV,'Visible','off');

 Meca=evalin('base','Meca');
 ListaR=ListadoRelaciones(Meca);
 set(handles.ListaRela,'String',ListaR);
 Meca.ListaR=ListaR;
 assignin('base','Meca',Meca)

 ListaC=ListadoCuerpos(Meca);
 set(handles.ListaCuerpos,'String',ListaC);
 Meca.ListaC=ListaC;
 assignin('base','Meca',Meca)

if Meca.Prob==1 || Meca.Prob==2
   % Comprobador del resultado del problema de velocidad
         aux=sum(abs(Meca.CompruebaVel));
         ErrorV=log10(aux);
         if ErrorV==-Inf
             fprintf('\n--> Problema de velocidad correcto\n    con error
0\n')
               Verde=imread('Verde.png');
               axes(handles.CompV);
               image(Verde)
               axis off
         else
              if aux<10^-5
                 fprintf('\n--> Problema de velociad correcto\n    con un
errror del 10^%d\n',ErrorV)
                    Verde=imread('Verde.png');
                    axes(handles.CompV);
                    image(Verde)
                    axis off
              else
                 fprintf('\n--> Problema de velocidad\n    recuelto con algun
tipo de error\n')
                    Amarillo=imread('Amarillo.png');
                    axes(handles.CompV);
                    image(Amarillo)
                    axis off
              end
         end
         if Meca.Prob==1
             set(handles.CompA,'Visible','off');
```

```
        end
    %
end
if Meca.Prob==2
    % Comprobador del resultado del problema de aceleración
        aux=sum(abs(Meca.CompruebaAce));
        ErrorA=log10(aux);
        if ErrorA==-Inf
            fprintf('\n--> Problema de aceleración correcto\n    con error
0\n')
             Amarill=imread('Verde.png');
             axes(handles.CompA);
             image(Verde)
             axis off
        else
            if aux<10^-5
               fprintf('\n--> Problema de aceleración correcto\n    con un
errror del 10^%d\n',ErrorV)
             Verde=imread('Verde.png');
             axes(handles.CompA);
             image(Verde)
             axis off
            else
               fprintf('\n--> Problema de aceleración\n    recuelto con
algun tipo de error\n')
                  Amarillo=imread('Amarillo.png');
                  axes(handles.CompA);
                  image(Amarillo)
                  axis off
            end
        end
    %
end

if Meca.Dibujo==1
    axes(handles.Grafica);
    HazDibujo(Meca);
end

if Meca.Simulacion==1
    Vista=evalin('base','Vista');
    axes(handles.Grafica);
    HazSimulacion(Meca,Vista);
end


% --- Outputs from this function are returned to the command line.
function varargout = MuestraResultados_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in ListaRela.
function ListaRela_Callback(hObject, eventdata, handles)
% hObject    handle to ListaRela (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns ListaRela contents
as cell array
%        contents{get(hObject,'Value')} returns selected item from ListaRela
a=get(handles.ListaRela,'Value');
 Meca=evalin('base','Meca');
set(handles.ResC1,'String',Meca.Rela(a).Cuerpos(1));
set(handles.ResC2,'String',Meca.Rela(a).Cuerpos(2));
set(handles.ResX,'String',Meca.Rela(a).x);
set(handles.ResY,'String',Meca.Rela(a).y);
set(handles.ResVx,'String',Meca.Rela(a).Vx);
set(handles.ResVy,'String',Meca.Rela(a).Vy);
if Meca.Prob==2
   set(handles.ResAx,'String',Meca.Rela(a).Ax);
   set(handles.ResAy,'String',Meca.Rela(a).Ay);
end


% --- Executes during object creation, after setting all properties.
function ListaRela_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ListaRela (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in ListaCuerpos.
function ListaCuerpos_Callback(hObject, eventdata, handles)
% hObject    handle to ListaCuerpos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns ListaCuerpos
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
ListaCuerpos
b=get(handles.ListaCuerpos,'Value');
 Meca=evalin('base','Meca');
set(handles.ResW,'String',Meca.Cuerpo(b).W);
if Meca.Prob==2
   set(handles.ResA,'String',Meca.Cuerpo(b).A);
end

% --- Executes during object creation, after setting all properties.
function ListaCuerpos_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ListaCuerpos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in CreaInforme.
function CreaInforme_Callback(hObject, eventdata, handles)
% hObject    handle to CreaInforme (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

helpdlg('Si esta usanso MCR no se creará el informe. Por favor cierre el
programa y ejecute en Matlab 2012a o superior el archivo adjunto
CAC_2D_Matlab.m para poder crear el informe.','Advertencia del uso de MCR');

   Meca=evalin('base','Meca');
   if isempty(Meca.Nombre)
      Meca.Nombre='Mecanismo';
   end

  W=what;
DirectorioA=W.path;
DirectorioN=[DirectorioA '\Informes de archivos'];
DirectorioN=[DirectorioN '\'];

   opts.format='pdf';
   opts.showCode=false;
   opts.outputDir=DirectorioN;

   publish('CreaInforme',opts);

cd(DirectorioN)
Frase=['Informe mecanismo - ' Meca.Nombre '.pdf'];
movefile('CreaInforme.pdf',Frase);
open(Frase);

cd(DirectorioA)
```

MUESTRAVELOCIDADES.M

```
function [MV] = MuestraVelocidades(Meca)
%% Función que muestra en un plot las velocidades de los pares del mecanismo
%%

MV=0;
hold on
for i=[1:1:length(Meca.Rela)]
    x=Meca.Rela(i).x;
    y=Meca.Rela(i).y;
    Vx=Meca.Rela(i).Vx;
    Vy=Meca.Rela(i).Vy;
    P=[x;y];
    V=[Vx;Vy];
    V=V./norm(V);
    Pfin=V+P;
    quiver3(P(1),P(2),0,Pfin(1)-P(1),Pfin(2)-
P(2),0,'r','linewidth',3,'MaxHeadSize',0.3)
end
% x0=0;
% y0=0;
% z0=0;
%
% x1=1;
% y1=2;
% z1=0;
% % axis([0 2 0 5 0 0 1 5]);
% hold on
% quiver3(x0,y0,z0,x1-x0,y1-y0,z1-z0,'r','linewidth',3,'MaxHeadSize',0.3)

MV=1;
end
```

NOPP.M

```
function [Meca] = NOPP(Meca)
%% Función que resuelve el caso particular de NPP=0 %%

   M=[];
   for i=[1:1:length(Meca.Tramo)]
       Tx=Meca.Tramo(i).T(1);
       Ty=Meca.Tramo(i).T(2);
       V=[-Ty;Tx];
       M=[M,V];
   end

    M(:,end)=[];
   Meca.M=M;


end
```

NUEVASCOORD.M

```
function [Meca] = NuevasCoord(Meca)
%% Función que calcula las nuevas coordenadas del mecanismo %%

t=0.02;
Error=0;
for i=[1:1:Meca.Nrest]
    Xa=Meca.Rela(i).x;
    Ya=Meca.Rela(i).y;
    Va=[Meca.Rela(i).Vx;Meca.Rela(i).Vy];
    Aa=[Meca.Rela(i).Ax;Meca.Rela(i).Ay];

    if norm(Aa)==0
        if norm(Va)==0
            Pn=[Xa;Ya];
        else
           V=Va;
           direcV=V./norm(V);
           moduV=norm(V)*t;
           Xn=moduV.*direcV;
           Pn=[Xa;Ya]+Xn;
        end
    else
       direcA=Aa./norm(Aa);
       moduV=norm(Aa)*t/2;
%        Error=(norm(Aa)*t)/(24*Meca.Iteracion);
%        moduV=moduV+Error;
       Vn=moduV.*direcA;
       V=Va+Vn;

       direcV=V./norm(V);
       moduV=norm(V)*t;
       Xn=moduV.*direcV;

       Pn=[Xa;Ya]+Xn;
    end

       Meca.Rela(i).x=Pn(1);
       Meca.Rela(i).y=Pn(2);
       Meca.Error=Error;
end


end
```

NUEVASGDL.M

```
function [Meca] = NuevasGDL(Meca)
%% Función que calcula las nuevas velocidades iniciales de los GDL %%

t=0.02;
AAa=[];
VAa=[];
f=[];
k=0;
for i=[1:1:Meca.Nbody-1]
%     AAa=[AAa;Meca.Cuerpo(i+1).A];
    r=Meca.AGDL(i);
%     f=[f;r];
    AAa=[AAa;r];
    r=Meca.VGDL(i);
    VAa=[VAa;r];
end
% AAn=AAa;
% for i=[1:1:length(f)]
%     if f(i)==0
%         AAn(i)=0;
%     end
% end

VAn=VAa+AAa.*t;

VLn=[];
if Meca.NPP>=2
    Linea=Meca.RCP(:,1);
    for i=[1:1:length(Linea)]
        if Linea(i)==1
        else
           ALa=[Meca.Rela(Linea(i)).Ax;Meca.Rela(Linea(i)).Ay];
           VLa=[Meca.Rela(Linea(i)).Vx;Meca.Rela(Linea(i)).Vy];
            if norm(ALa)==0
                VLn=VLa;
            else
               direcA=ALa./norm(ALa);
               moduA=norm(ALa);

               velo=VLa+((moduA*t).*direcA);
               VLn=[VLn;velo];
            end
        end
    end
end

Meca.VGDL=[VAn;VLn];

end
```

PRESIMULACION.M

```
function varargout = PreSimulacion(varargin)
% PRESIMULACION MATLAB code for PreSimulacion.fig
%      PRESIMULACION, by itself, creates a new PRESIMULACION or raises the
existing
%      singleton*.
%
%      H = PRESIMULACION returns the handle to a new PRESIMULACION or the
handle to
%      the existing singleton*.
%
%      PRESIMULACION('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in PRESIMULACION.M with the given input
arguments.
%
%      PRESIMULACION('Property','Value',...) creates a new PRESIMULACION or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before PreSimulacion_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to PreSimulacion_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help PreSimulacion

% Last Modified by GUIDE v2.5 04-Jul-2016 17:40:44

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @PreSimulacion_OpeningFcn, ...
                   'gui_OutputFcn',  @PreSimulacion_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before PreSimulacion is made visible.
function PreSimulacion_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% varargin   command line arguments to PreSimulacion (see VARARGIN)
set(handles.Xmin,'String','-15');
set(handles.Ymin,'String','-15');
set(handles.Xmax,'String','15');
set(handles.Ymax,'String','15');
set(handles.Npasos,'String','50');
% Choose default command line output for PreSimulacion
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes PreSimulacion wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = PreSimulacion_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in Continuar.
function Continuar_Callback(hObject, eventdata, handles)
% hObject    handle to Continuar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Xmin=get(handles.Xmin,'String');
if isempty(Xmin)
    warndlg('Rellene el campo X mínima','Mensaje');
else
    Xmin=str2double(Xmin);
end
Ymin=get(handles.Ymin,'String');
if isempty(Xmin)
    warndlg('Rellene el campo Y mínima','Mensaje');
else
    Ymin=str2double(Ymin);
end
Xmax=get(handles.Xmax,'String');
if isempty(Xmax)
    warndlg('Rellene el campo X máxima','Mensaje');
else
    Xmax=str2double(Xmax);
end
Ymax=get(handles.Ymax,'String');
if isempty(Ymax)
    warndlg('Rellene el campo Y máxima','Mensaje');
else
    Ymax=str2double(Ymax);
end
Npasos=get(handles.Npasos,'String');
if isempty(Npasos)
    warndlg('Rellene el campo Número de pasos','Mensaje');
else
```

```
    Npasos=str2double(Npasos);
end
Vista.Xmin=Xmin;
Vista.Ymin=Ymin;
Vista.Xmax=Xmax;
Vista.Ymax=Ymax;
Vista.Npasos=Npasos;
assignin('base','Vista',Vista);
close
MuestraResultados();



function Npasos_Callback(hObject, eventdata, handles)
% hObject    handle to Npasos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Npasos as text
%        str2double(get(hObject,'String')) returns contents of Npasos as a
double


% --- Executes during object creation, after setting all properties.
function Npasos_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Npasos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Xmin_Callback(hObject, eventdata, handles)
% hObject    handle to Xmin (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Xmin as text
%        str2double(get(hObject,'String')) returns contents of Xmin as a
double


% --- Executes during object creation, after setting all properties.
function Xmin_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Xmin (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function Ymin_Callback(hObject, eventdata, handles)
% hObject    handle to Ymin (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Ymin as text
%        str2double(get(hObject,'String')) returns contents of Ymin as a
double


% --- Executes during object creation, after setting all properties.
function Ymin_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Ymin (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Ymax_Callback(hObject, eventdata, handles)
% hObject    handle to Ymax (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Ymax as text
%        str2double(get(hObject,'String')) returns contents of Ymax as a
double


% --- Executes during object creation, after setting all properties.
function Ymax_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Ymax (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Xmax_Callback(hObject, eventdata, handles)
% hObject    handle to Xmax (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of Xmax as text
%        str2double(get(hObject,'String')) returns contents of Xmax as a
double


% --- Executes during object creation, after setting all properties.
function Xmax_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Xmax (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

PROBACEL.M

```
function [Meca] = ProbAcel(Meca)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Programa para resolver el problema de aceleraciones %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Cálculo de la matriz WTH
   Meca=MatrizWTH(Meca);
%

% Cálculo de la matriz AH
   Meca=MatrizAH(Meca);
%

% Cálculo de la matriz WVH
   Meca=MatrizWVH(Meca);
%
% Aplicación de los GDL y creación de las matrices Meca.Qind y Meca.Qdep
   Meca=CondAGDL(Meca);
%

% Resolución del problema de aceleración
Meca.Qind
Meca.Qind=(Meca.Qind)-Meca.WTH-Meca.AH-Meca.WVH;
Meca.Qind
 Sol=Meca.Qdep\-Meca.Qind
 Sol=Meca.AGDL+Sol;
 Meca.SolAcel=Sol;
 if Meca.NPP~=0
     Meca=CondPrisSolA(Meca);
 end

%



end
```

PRUEBA11.M: CÓDIGO PRINCIPAL

```
function varargout = CAC_2D(varargin)
% PRUEBA11 MATLAB code for Prueba11.fig
%      PRUEBA11, by itself, creates a new PRUEBA11 or raises the existing
%      singleton*.
%
%      H = PRUEBA11 returns the handle to a new PRUEBA11 or the handle to
%      the existing singleton*.
%
%      PRUEBA11('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PRUEBA11.M with the given input arguments.
%
%      PRUEBA11('Property','Value',...) creates a new PRUEBA11 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Prueba11_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Prueba11_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Prueba11

% Last Modified by GUIDE v2.5 19-Jul-2016 04:45:04

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Prueba11_OpeningFcn, ...
                   'gui_OutputFcn',  @Prueba11_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Prueba11 is made visible.
function Prueba11_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Prueba11 (see VARARGIN)

% Choose default command line output for Prueba11
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);

% clc
Logo=imread('Logo1.png');
axes(handles.Logo);
imshow(Logo)
axis off

SPR=imread('PR_S1.jpg');
axes(handles.Simbolo_PR);
image(SPR)
axis off

SPP=imread('PP_S.jpg');
axes(handles.Simbolo_PP);
image(SPP)
axis off

SPRSD=imread('PRSD_S.jpg');
axes(handles.Simbolo_PRSD);
image(SPRSD)
axis off

   W=what;
   Directorio=W.path;
   Directorio=[Directorio '\'];
   mkdir([Directorio,'Archivos guardados'])

   W=what;
   Directorio=W.path;
   Directorio=[Directorio '\'];
   mkdir([Directorio,'Informes de archivos'])
% UIWAIT makes Prueba11 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Prueba11_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function NPR_Callback(hObject, eventdata, handles)
% hObject    handle to NPR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of NPR as text
%        str2double(get(hObject,'String')) returns contents of NPR as a
double
Nbody=get(handles.Nbody,'String');
Nrest=get(handles.Nrest,'String');
```

```
NPR=get(handles.NPR,'String');
NPP=get(handles.NPP,'String');
NPRSD=get(handles.NPRSD,'String');
Lazo=get(handles.Lazo,'String');
LazoPares=get(handles.LazoPares,'String');
NPRSD=str2double(NPRSD);
    if NPRSD>=1
       set(handles.Simulacion,'Enable','off');
       set(handles.Simulacion,'Value',0);
    end
    if NPRSD==0
      set(handles.Simulacion,'Enable','on');
    end

if isempty(Nbody) || isempty(Nrest) || isempty(NPR) || isempty(NPP) ||
isempty(NPRSD) || isempty(Lazo) || isempty(LazoPares)
    set(handles.VGDL,'Enable','off');
    set(handles.AGDL,'Enable','off');
    Frase='Faltan datos del mecanismo';
    set(handles.TextVang,'FontName','MS Sans Serif');
    set(handles.TextAang,'FontName','MS Sans Serif');
    set(handles.TextVang,'String',Frase);
    set(handles.TextAang,'String',Frase);
else
   set(handles.VGDL,'Enable','on');
   set(handles.AGDL,'Enable','on');
   Meca.Nbody=str2double(Nbody);
   Meca.Nrest=str2double(Nrest);
   Meca.NPR=str2double(NPR);
   Meca.NPP=str2double(NPP);
   Meca.NPRSD=str2double(NPRSD);
   Meca.Lazo=str2num(Lazo);
   Meca.LazoPares=str2num(LazoPares);
      RCP=[];
        Meca=BucleAC(Meca);
      for i=[1:1:length(Meca.LazoPares)]
         if Meca.LazoPares(i)==2
             C1=Meca.Lazo(i-1);
             C2=Meca.Lazo(i);
             RCP=[RCP;C1,C2];
         end
      end
      Meca.RCP=RCP;
    Meca=CreaChar(Meca);
    set(handles.TextVang,'FontName','Symbol');
    set(handles.TextAang,'FontName','Symbol');
    set(handles.TextVang,'String',Meca.GDL.CharV.ang);
    set(handles.TextAang,'String',Meca.GDL.CharA.ang);
    set(handles.TextVlin,'FontName','MS Sans Serif');
    set(handles.TextAling,'FontName','MS Sans Serif');
    set(handles.TextVlin,'String',Meca.GDL.CharV.lin);
    set(handles.TextAlin,'String',Meca.GDL.CharA.lin);

end


% --- Executes during object creation, after setting all properties.
function NPR_CreateFcn(hObject, eventdata, handles)
% hObject    handle to NPR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function NPP_Callback(hObject, eventdata, handles)
% hObject    handle to NPP (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of NPP as text
%        str2double(get(hObject,'String')) returns contents of NPP as a
double
Nbody=get(handles.Nbody,'String');
Nrest=get(handles.Nrest,'String');
NPR=get(handles.NPR,'String');
NPP=get(handles.NPP,'String');
NPRSD=get(handles.NPRSD,'String');
Lazo=get(handles.Lazo,'String');
LazoPares=get(handles.LazoPares,'String');

if isempty(Nbody) || isempty(Nrest) || isempty(NPR) || isempty(NPP) ||
isempty(NPRSD) || isempty(Lazo) || isempty(LazoPares)
    set(handles.VGDL,'Enable','off');
    set(handles.AGDL,'Enable','off');
    Frase='Faltan datos del mecanismo';
    set(handles.TextVang,'FontName','MS Sans Serif');
    set(handles.TextAang,'FontName','MS Sans Serif');
    set(handles.TextVang,'String',Frase);
    set(handles.TextAang,'String',Frase);
else
   set(handles.VGDL,'Enable','on');
   set(handles.AGDL,'Enable','on');
   Meca.Nbody=str2double(Nbody);
   Meca.Nrest=str2double(Nrest);
   Meca.NPR=str2double(NPR);
   Meca.NPP=str2double(NPP);
   Meca.NPRSD=str2double(NPRSD);
   Meca.Lazo=str2num(Lazo);
   Meca.LazoPares=str2num(LazoPares);
      RCP=[];
        Meca=BucleAC(Meca);
      for i=[1:1:length(Meca.LazoPares)]
         if Meca.LazoPares(i)==2
             C1=Meca.Lazo(i-1);
             C2=Meca.Lazo(i);
             RCP=[RCP;C1,C2];
         end
      end
      Meca.RCP=RCP;
      Meca=CreaChar(Meca);
    set(handles.TextVang,'FontName','Symbol');
    set(handles.TextAang,'FontName','Symbol');
    set(handles.TextVang,'String',Meca.GDL.CharV.ang);
    set(handles.TextAang,'String',Meca.GDL.CharA.ang);
    set(handles.TextVlin,'FontName','MS Sans Serif');
    set(handles.TextAlin,'FontName','MS Sans Serif');
```

```
    set(handles.TextVlin,'String',Meca.GDL.CharV.lin);
    set(handles.TextAlin,'String',Meca.GDL.CharA.lin);
end


% --- Executes during object creation, after setting all properties.
function NPP_CreateFcn(hObject, eventdata, handles)
% hObject    handle to NPP (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function NPRSD_Callback(hObject, eventdata, handles)
% hObject    handle to NPRSD (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of NPRSD as text
%        str2double(get(hObject,'String')) returns contents of NPRSD as a
double
Nbody=get(handles.Nbody,'String');
Nrest=get(handles.Nrest,'String');
NPR=get(handles.NPR,'String');
NPP=get(handles.NPP,'String');
NPRSD=get(handles.NPRSD,'String');
Lazo=get(handles.Lazo,'String');
LazoPares=get(handles.LazoPares,'String');
NPRSD=str2double(NPRSD);
    if NPRSD>=1
       set(handles.Simulacion,'Enable','off');
       set(handles.Simulacion,'Value',0);
    end
    if NPRSD==0
      set(handles.Simulacion,'Enable','on');
    end

if isempty(Nbody) || isempty(Nrest) || isempty(NPR) || isempty(NPP) ||
isempty(NPRSD) || isempty(Lazo) || isempty(LazoPares)
    set(handles.VGDL,'Enable','off');
    set(handles.AGDL,'Enable','off');
    Frase='Faltan datos del mecanismo';
    set(handles.TextVang,'FontName','MS Sans Serif');
    set(handles.TextAang,'FontName','MS Sans Serif');
    set(handles.TextVang,'String',Frase);
    set(handles.TextAang,'String',Frase);
else
   set(handles.VGDL,'Enable','on');
   set(handles.AGDL,'Enable','on');
   Meca.Nbody=str2double(Nbody);
   Meca.Nrest=str2double(Nrest);
   Meca.NPR=str2double(NPR);
   Meca.NPP=str2double(NPP);
   Meca.NPRSD=str2double(NPRSD);
   Meca.Lazo=str2num(Lazo);
```

```
   Meca.LazoPares=str2num(LazoPares);
      RCP=[];
        Meca=BucleAC(Meca);
      for i=[1:1:length(Meca.LazoPares)]
         if Meca.LazoPares(i)==2
             C1=Meca.Lazo(i-1);
             C2=Meca.Lazo(i);
             RCP=[RCP;C1,C2];
         end
      end
      Meca.RCP=RCP;
      Meca=CreaChar(Meca);
    set(handles.TextVang,'FontName','Symbol');
    set(handles.TextAang,'FontName','Symbol');
    set(handles.TextVang,'String',Meca.GDL.CharV.ang);
    set(handles.TextAang,'String',Meca.GDL.CharA.ang);
    set(handles.TextVlin,'FontName','MS Sans Serif');
    set(handles.TextAlin,'FontName','MS Sans Serif');
    set(handles.TextVlin,'String',Meca.GDL.CharV.lin);
    set(handles.TextAlin,'String',Meca.GDL.CharA.lin);
end

% --- Executes during object creation, after setting all properties.
function NPRSD_CreateFcn(hObject, eventdata, handles)
% hObject    handle to NPRSD (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in Resolver.
function Resolver_Callback(hObject, eventdata, handles)
% hObject    handle to Resolver (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

Nbody=get(handles.Nbody,'String');
if isempty(Nbody)
    warndlg('Rellene el campo Número de eslabones','Mensaje');
else
    Nbody=str2double(Nbody);
end
Nrest=get(handles.Nrest,'String');
if isempty(Nrest)
    warndlg('Rellene el campo Número de pares de restricción','Mensaje');
else
    Nrest=str2double(Nrest);
end
NPR=get(handles.NPR,'String');
if isempty(NPR)
    warndlg('Rellene el campo Pares de revolución','Mensaje');
else
    NPR=str2double(NPR);
end
NPP=get(handles.NPP,'String');
```

```
if isempty(NPP)
    warndlg('Rellene el campo Pares prismáticos','Mensaje');
else
    NPP=str2double(NPP);
end
NPRSD=get(handles.NPRSD,'String');
if isempty(NPRSD)
    warndlg('Rellene el campo Pares de rodadura sin
deslizamiento','Mensaje');
else
    NPRSD=str2double(NPRSD);
end
DirecPP=get(handles.DirecPP,'Data');
if isempty(DirecPP) && NPP>=1
    warndlg('Rellene el campo Direcciones de los pares
prismáticos','Mensaje');
else
%       DirecPP=str2double(DirecPP);
   assignin('base','M',DirecPP);
   if iscell(DirecPP)
      DirecPP=cell2mat(DirecPP);
   end
end
DirecPRSD=get(handles.DirecPRSD,'Data');
if isempty(DirecPRSD) && NPRSD>=1
    warndlg('Rellene el campo Direcciones de los pares de rodadura sin
deslizamiento','Mensaje');
else
%     DirecPRSD=str2double(DirecPRSD);
     if iscell(DirecPRSD)
        DirecPRSD=cell2mat(DirecPRSD);
     end
end
Radios=get(handles.Radios,'Data');
if isempty(Radios) && NPRSD>=1
    warndlg('Rellene el campo Radios de curvatura','Mensaje');
else
%     Radios=str2double(Radios);
    if iscell(Radios)
        Radios=cell2mat(Radios);
    end
end
Coord=get(handles.Coord,'Data');
if isempty(Coord)
    warndlg('Rellene el campo Coordenadas','Mensaje');
else
%     Coord=str2double(Coord)
   if iscell(Coord)
      Coord=cell2mat(Coord);
   end
end
Lazo=get(handles.Lazo,'String');
if isempty(Lazo)
    warndlg('Rellene el campo Lazo','Mensaje');
else
    Lazo=str2num(Lazo);
end
LazoPares=get(handles.LazoPares,'String');
if isempty(LazoPares)
    warndlg('Rellene el campo Lazo de los pares','Mensaje');
else
    LazoPares=str2num(LazoPares);
end
```

```
VGDL=get(handles.VGDL,'String');
if isempty(VGDL)
    warndlg('Rellene el campo Velocidades iniciales','Mensaje');
else
    VGDL=str2num(VGDL)';
end
AGDL=get(handles.AGDL,'String');
if isempty(AGDL)
    warndlg('Rellene el campo Aceleraciones iniciales','Mensaje');
else
    AGDL=str2num(AGDL)';
end

if NPRSD>=1
Radios=[Radios DirecPRSD(:,end)];
DirecPRSD(:,end)=[];
end

Velo=get(handles.Velo,'Value');
VeloyAce=get(handles.VeloyAce,'Value');
Simulacion=get(handles.Simulacion,'Value');
  if Velo==1
      Prob=1;
  end
  if VeloyAce==1
      Prob=2;
  end
       if NPRSD>=1 && Simulacion==1
        warndlg('No se podrá representar la simulación del mecanismo. Lea
condiciones de la simulación','Mensaje');
        Simulacion=0;
       end

  if Simulacion==1
       Meca.Simulacion=1;
       Prob=2;

Meca=Resuelve(Nbody,Nrest,NPR,NPP,NPRSD,Lazo,LazoPares,Coord,DirecPP,Radios,D
irecPRSD,VGDL,AGDL,Prob);
  Meca.Prob=Prob;
  Meca.Dibujo=1;
  Meca.Simulacion=Simulacion;
  Meca.Nombre=get(handles.Nombre,'String');
  assignin('base','Meca',Meca);
       PreSimulacion();
  else
      Meca.Simulacion=0;

Meca=Resuelve(Nbody,Nrest,NPR,NPP,NPRSD,Lazo,LazoPares,Coord,DirecPP,Radios,D
irecPRSD,VGDL,AGDL,Prob);
  Meca.Prob=Prob;
  Meca.Dibujo=1;
  Meca.Simulacion=Simulacion;
  Meca.Nombre=get(handles.Nombre,'String');
  assignin('base','Meca',Meca);
MuestraResultados();
  end


function VGDL_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to VGDL (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of VGDL as text
%        str2double(get(hObject,'String')) returns contents of VGDL as a
double


% --- Executes during object creation, after setting all properties.
function VGDL_CreateFcn(hObject, eventdata, handles)
% hObject    handle to VGDL (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function AGDL_Callback(hObject, eventdata, handles)
% hObject    handle to AGDL (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of AGDL as text
%        str2double(get(hObject,'String')) returns contents of AGDL as a
double


% --- Executes during object creation, after setting all properties.
function AGDL_CreateFcn(hObject, eventdata, handles)
% hObject    handle to AGDL (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Lazo_Callback(hObject, eventdata, handles)
% hObject    handle to Lazo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Lazo as text
%        str2double(get(hObject,'String')) returns contents of Lazo as a
double
Nbody=get(handles.Nbody,'String');
Nrest=get(handles.Nrest,'String');
NPR=get(handles.NPR,'String');
```

```
NPP=get(handles.NPP,'String');
NPRSD=get(handles.NPRSD,'String');
Lazo=get(handles.Lazo,'String');
LazoPares=get(handles.LazoPares,'String');

if isempty(Nbody) || isempty(Nrest) || isempty(NPR) || isempty(NPP) ||
isempty(NPRSD) || isempty(Lazo) || isempty(LazoPares)
    set(handles.VGDL,'Enable','off');
    set(handles.AGDL,'Enable','off');
    Frase='Faltan datos del mecanismo';
    set(handles.TextVang,'FontName','MS Sans Serif');
    set(handles.TextAang,'FontName','MS Sans Serif');
    set(handles.TextVang,'String',Frase);
    set(handles.TextAang,'String',Frase);
else
   set(handles.VGDL,'Enable','on');
   set(handles.AGDL,'Enable','on');
   Meca.Nbody=str2double(Nbody);
   Meca.Nrest=str2double(Nrest);
   Meca.NPR=str2double(NPR);
   Meca.NPP=str2double(NPP);
   Meca.NPRSD=str2double(NPRSD);
   Meca.Lazo=str2num(Lazo);
   Meca.LazoPares=str2num(LazoPares);
      RCP=[];
      Meca=BucleAC(Meca);
      for i=[1:1:length(Meca.LazoPares)]
         if Meca.LazoPares(i)==2
             C1=Meca.Lazo(i-1);
             C2=Meca.Lazo(i);
             RCP=[RCP;C1,C2];
         end
      end
      Meca.RCP=RCP;
      Meca=CreaChar(Meca);
    set(handles.TextVang,'FontName','Symbol');
    set(handles.TextAang,'FontName','Symbol');
    set(handles.TextVang,'String',Meca.GDL.CharV.ang);
    set(handles.TextAang,'String',Meca.GDL.CharA.ang);
    set(handles.TextVlin,'FontName','MS Sans Serif');
    set(handles.TextAlin,'FontName','MS Sans Serif');
    set(handles.TextVlin,'String',Meca.GDL.CharV.lin);
    set(handles.TextAlin,'String',Meca.GDL.CharA.lin);
end

% --- Executes during object creation, after setting all properties.
function Lazo_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Lazo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function LazoPares_Callback(hObject, eventdata, handles)
% hObject    handle to LazoPares (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of LazoPares as text
%        str2double(get(hObject,'String')) returns contents of LazoPares as a
double
Nbody=get(handles.Nbody,'String');
Nrest=get(handles.Nrest,'String');
NPR=get(handles.NPR,'String');
NPP=get(handles.NPP,'String');
NPRSD=get(handles.NPRSD,'String');
Lazo=get(handles.Lazo,'String');
LazoPares=get(handles.LazoPares,'String');

if isempty(Nbody) || isempty(Nrest) || isempty(NPR) || isempty(NPP) ||
isempty(NPRSD) || isempty(Lazo) || isempty(LazoPares)
    set(handles.VGDL,'Enable','off');
    set(handles.AGDL,'Enable','off');
    Frase='Faltan datos del mecanismo';
    set(handles.TextVang,'FontName','MS Sans Serif');
    set(handles.TextAang,'FontName','MS Sans Serif');
    set(handles.TextVang,'String',Frase);
    set(handles.TextAang,'String',Frase);
else
   set(handles.VGDL,'Enable','on');
   set(handles.AGDL,'Enable','on');
   Meca.Nbody=str2double(Nbody);
   Meca.Nrest=str2double(Nrest);
   Meca.NPR=str2double(NPR);
   Meca.NPP=str2double(NPP);
   Meca.NPRSD=str2double(NPRSD);
   Meca.Lazo=str2num(Lazo);
   Meca.LazoPares=str2num(LazoPares);
      RCP=[];
      Meca=BucleAC(Meca);
      for i=[1:1:length(Meca.LazoPares)]
         if Meca.LazoPares(i)==2
             C1=Meca.Lazo(i-1);
             C2=Meca.Lazo(i);
             RCP=[RCP;C1,C2];
         end
      end
      Meca.RCP=RCP;
      Meca=CreaChar(Meca);
    set(handles.TextVang,'FontName','Symbol');
    set(handles.TextAang,'FontName','Symbol');
    set(handles.TextVang,'String',Meca.GDL.CharV.ang);
    set(handles.TextAang,'String',Meca.GDL.CharA.ang);
    set(handles.TextVlin,'FontName','MS Sans Serif');
    set(handles.TextAlin,'FontName','MS Sans Serif');
    set(handles.TextVlin,'String',Meca.GDL.CharV.lin);
    set(handles.TextAlin,'String',Meca.GDL.CharA.lin);
end

% --- Executes during object creation, after setting all properties.
function LazoPares_CreateFcn(hObject, eventdata, handles)
% hObject    handle to LazoPares (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Nbody_Callback(hObject, eventdata, handles)
% hObject    handle to Nbody (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Nbody as text
%        str2double(get(hObject,'String')) returns contents of Nbody as a
double
Nbody=get(handles.Nbody,'String');
Nrest=get(handles.Nrest,'String');
NPR=get(handles.NPR,'String');
NPP=get(handles.NPP,'String');
NPRSD=get(handles.NPRSD,'String');
Lazo=get(handles.Lazo,'String');
LazoPares=get(handles.LazoPares,'String');
NPRSD=str2double(NPRSD)
    if NPRSD>=1
       set(handles.Simulacion,'Enable','off');
       set(handles.Simulacion,'Value',0);
    end
    if NPRSD==0
      set(handles.Simulacion,'Enable','on');
    end

if isempty(Nbody) || isempty(Nrest) || isempty(NPR) || isempty(NPP) ||
isempty(NPRSD) || isempty(Lazo) || isempty(LazoPares)
    set(handles.VGDL,'Enable','off');
    set(handles.AGDL,'Enable','off');
    Frase='Faltan datos del mecanismo';
    set(handles.TextVang,'FontName','MS Sans Serif');
    set(handles.TextAang,'FontName','MS Sans Serif');
    set(handles.TextVang,'String',Frase);
    set(handles.TextAang,'String',Frase);
else
   set(handles.VGDL,'Enable','on');
   set(handles.AGDL,'Enable','on');
   Meca.Nbody=str2double(Nbody);
   Meca.Nrest=str2double(Nrest);
   Meca.NPR=str2double(NPR);
   Meca.NPP=str2double(NPP);
   Meca.NPRSD=str2double(NPRSD);
   Meca.Lazo=str2num(Lazo);
   Meca.LazoPares=str2num(LazoPares);
      RCP=[];
       Meca=BucleAC(Meca);
      for i=[1:1:length(Meca.LazoPares)]
         if Meca.LazoPares(i)==2
             C1=Meca.Lazo(i-1);
             C2=Meca.Lazo(i);
             RCP=[RCP;C1,C2];
         end
      end
```

```
      Meca.RCP=RCP;
      Meca=CreaChar(Meca);
    set(handles.TextVang,'FontName','Symbol');
    set(handles.TextAang,'FontName','Symbol');
    set(handles.TextVang,'String',Meca.GDL.CharV.ang);
    set(handles.TextAang,'String',Meca.GDL.CharA.ang);
    set(handles.TextVlin,'FontName','MS Sans Serif');
    set(handles.TextAlin,'FontName','MS Sans Serif');
    set(handles.TextVlin,'String',Meca.GDL.CharV.lin);
    set(handles.TextAlin,'String',Meca.GDL.CharA.lin);
end

if isempty(Nbody) || isempty(Nrest)
    set(handles.NGDL,'Visible','off');
else
    set(handles.NGDL,'Visible','on');
    Meca.Nbody=str2double(Nbody);
    Meca.Nrest=str2double(Nrest);
    NGDL=3*(Meca.Nbody-1)-2*Meca.Nrest;
    set(handles.NGDL,'String',NGDL);
end

% --- Executes during object creation, after setting all properties.
function Nbody_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Nbody (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Nrest_Callback(hObject, eventdata, handles)
% hObject    handle to Nrest (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Nrest as text
%        str2double(get(hObject,'String')) returns contents of Nrest as a
double
Nbody=get(handles.Nbody,'String');
Nrest=get(handles.Nrest,'String');
NPR=get(handles.NPR,'String');
NPP=get(handles.NPP,'String');
NPRSD=get(handles.NPRSD,'String');
Lazo=get(handles.Lazo,'String');
LazoPares=get(handles.LazoPares,'String');
NPRSD=str2double(NPRSD);
    if NPRSD>=1
       set(handles.Simulacion,'Enable','off');
       set(handles.Simulacion,'Value',0);
    end
    if NPRSD==0
      set(handles.Simulacion,'Enable','on');
    end
```

```
if isempty(Nbody) || isempty(Nrest) || isempty(NPR) || isempty(NPP) ||
isempty(NPRSD) || isempty(Lazo) || isempty(LazoPares)
    set(handles.VGDL,'Enable','off');
    set(handles.AGDL,'Enable','off');
    Frase='Faltan datos del mecanismo';
    set(handles.TextVang,'FontName','MS Sans Serif');
    set(handles.TextAang,'FontName','MS Sans Serif');
    set(handles.TextVang,'String',Frase);
    set(handles.TextAang,'String',Frase);
else
   set(handles.VGDL,'Enable','on');
   set(handles.AGDL,'Enable','on');
   Meca.Nbody=str2double(Nbody);
   Meca.Nrest=str2double(Nrest);
   Meca.NPR=str2double(NPR);
   Meca.NPP=str2double(NPP);
   Meca.NPRSD=str2double(NPRSD);
   Meca.Lazo=str2num(Lazo);
   Meca.LazoPares=str2num(LazoPares);
      RCP=[];
       Meca=BucleAC(Meca);
      for i=[1:1:length(Meca.LazoPares)]
         if Meca.LazoPares(i)==2
             C1=Meca.Lazo(i-1);
             C2=Meca.Lazo(i);
             RCP=[RCP;C1,C2];
         end
      end
      Meca.RCP=RCP;
      Meca=CreaChar(Meca);
    set(handles.TextVang,'FontName','Symbol');
    set(handles.TextAang,'FontName','Symbol');
    set(handles.TextVang,'String',Meca.GDL.CharV.ang);
    set(handles.TextAang,'String',Meca.GDL.CharA.ang);
    set(handles.TextVlin,'FontName','MS Sans Serif');
    set(handles.TextAlin,'FontName','MS Sans Serif');
    set(handles.TextVlin,'String',Meca.GDL.CharV.lin);
    set(handles.TextAlin,'String',Meca.GDL.CharA.lin);
end

if isempty(Nbody) || isempty(Nrest)
    set(handles.NGDL,'Visible','off');
else
    set(handles.NGDL,'Visible','on');
    Meca.Nbody=str2double(Nbody);
    Meca.Nrest=str2double(Nrest);
    NGDL=3*(Meca.Nbody-1)-2*Meca.Nrest;
    set(handles.NGDL,'String',NGDL);
end

% --- Executes during object creation, after setting all properties.
function Nrest_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Nrest (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end

% --------------------------------------------------------------------
function Untitled_1_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_2_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function ManualUsuario_Callback(hObject, eventdata, handles)
% hObject    handle to ManualUsuario (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
open('Manual_de_usuario.pdf')


% --------------------------------------------------------------------
function Guardar_Callback(hObject, eventdata, handles)
% hObject    handle to Guardar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Nbody=get(handles.Nbody,'String');
if isempty(Nbody)
    warndlg('Rellene el campo Número de cuerpos','Mensaje');
else
    Nbody=str2double(Nbody);
end
Nrest=get(handles.Nrest,'String');
if isempty(Nrest)
    warndlg('Rellene el campo Número de pares de restricción','Mensaje');
else
    Nrest=str2double(Nrest);
end
NPR=get(handles.NPR,'String');
if isempty(NPR)
    warndlg('Rellene el campo Pares de revolución','Mensaje');
else
    NPR=str2double(NPR);
end
NPP=get(handles.NPP,'String');
if isempty(NPP)
    warndlg('Rellene el campo Pares prismáticos','Mensaje');
else
    NPP=str2double(NPP);
end
NPRSD=get(handles.NPRSD,'String');
if isempty(NPRSD)
    warndlg('Rellene el campo Pares de rodadura sin
deslizamiento','Mensaje');
else
    NPRSD=str2double(NPRSD);
end
DirecPP=get(handles.DirecPP,'Data');
if isempty(DirecPP) && NPP>=1
```

```
    warndlg('Rellene el campo Direcciones de los pares
prismáticos','Mensaje');
else
    if iscell(DirecPP)
       DirecPP=cell2mat(DirecPP);
    end
end
DirecPRSD=get(handles.DirecPRSD,'Data');
if isempty(DirecPRSD) && NPRSD>=1
    warndlg('Rellene el campo Direcciones de los pares de rodadura sin
deslizamiento','Mensaje');
else
    if iscell(DirecPRSD)
       DirecPRSD=cell2mat(DirecPRSD);
    end
end
Radios=get(handles.Radios,'Data');
if isempty(Radios) && NPRSD>=1
    warndlg('Rellene el campo Radios de curvatura','Mensaje');
else
    if iscell(Radios)
       Radios=cell2mat(Radios);
    end
end
Coord=get(handles.Coord,'Data');
if isempty(Coord)
    warndlg('Rellene el campo Coordenadas','Mensaje');
else
    if iscell(Coord)
       Coord=cell2mat(Coord);
    end
end
Lazo=get(handles.Lazo,'String');
if isempty(Lazo)
    warndlg('Rellene el campo Lazo','Mensaje');
else
    Lazo=str2num(Lazo);
end
LazoPares=get(handles.LazoPares,'String');
if isempty(LazoPares)
    warndlg('Rellene el campo Lazo de los pares','Mensaje');
else
    LazoPares=str2num(LazoPares);
end
VGDL=get(handles.VGDL,'String');
if isempty(VGDL)
    warndlg('Rellene el campo Velocidades iniciales','Mensaje');
else
    VGDL=str2num(VGDL)';
end
AGDL=get(handles.AGDL,'String');
if isempty(AGDL)
    warndlg('Rellene el campo Aceleraciones iniciales','Mensaje');
else
    AGDL=str2num(AGDL)';
end
Nombre=get(handles.Nombre,'String');


Save.Nombre=Nombre;
Save.Nbody=Nbody;
```

```
Save.Nrest=Nrest;
Save.NPR=NPR;
Save.NPP=NPP;
Save.NPRSD=NPRSD;
Save.VGDL=VGDL;
Save.AGDL=AGDL;
Save.Lazo=Lazo;
Save.LazoPares=LazoPares;
Save.Coord=Coord;
Save.DirecPP=DirecPP;
Save.DirecPRSD=DirecPRSD;
Save.Radios=Radios;

s=Guarda_Archivos(Save);
if s==0
    msgbox('Error al guardar archivo','Mensaje','error');
end
if s==1
    if isempty(Save.Nombre)
       Bien=imread('Bien.png');
       Frase='Archivo guardado con exito. Nombre:';
       Frase=[Frase ' Mecanismo'];
       msgbox(Frase,'Mensaje','custom',Bien);
    else
       Bien=imread('Bien.png');
       Frase='Archivo guardado con exito. Nombre:';
       Frase=[Frase Save.Nombre];
       msgbox(Frase,'Mensaje','custom',Bien);
    end
end



% --------------------------------------------------------------------
function Cargar_Callback(hObject, eventdata, handles)
% hObject    handle to Cargar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Nombre=get(handles.Nombre,'String');

if isempty(Nombre)
    warndlg('Introduzca el nombre del archivo a abrir en el campo: Nombre del
archivo','Mensaje');
else
    W=what;
    DirectorioA=W.path;
    DirectorioN=[DirectorioA '\Archivos guardados'];
    DirectorioN=[DirectorioN '\'];

    cd(DirectorioN);
    DirectorioF=[DirectorioN Nombre];
    DirectorioF=[DirectorioF '\'];
    E=exist(DirectorioF);
       if E==0
           Frase='No existe ningún archivo guardado con el nombre: ';
           Frase=[Frase Nombre];
           warndlg(Frase,'Mensaje');
           cd(DirectorioA);
       else
           cd(DirectorioA);
           DirectorioF=double(char(DirectorioF));
           Meca=Carga_Archivos(DirectorioF);
```

```
            cd(DirectorioA);
            Nbody=Meca.Nbody;
            set(handles.Nbody,'String',Nbody);
            Nrest=Meca.Nrest;
            set(handles.Nrest,'String',Nrest);
            NPR=Meca.NPR;
            set(handles.NPR,'String',NPR);
            NPP=Meca.NPP;
            set(handles.NPP,'String',NPP);
            NPRSD=Meca.NPRSD;
            set(handles.NPRSD,'String',NPRSD);
            Lazo=Meca.Lazo;
            set(handles.Lazo,'String',Lazo);
            LazoPares=Meca.LazoPares;
            set(handles.LazoPares,'String',LazoPares);
            VGDL=Meca.VGDL;
            set(handles.VGDL,'String',VGDL);
            AGDL=Meca.AGDL;
            set(handles.AGDL,'String',AGDL);
            Coord=Meca.Coord;
            assignin('base','Coord',Coord);
%             Coord=num2str(Coord);
            set(handles.Coord,'Data',Coord);
            DirecPP=Meca.DirecPP;
            assignin('base','DirecPP',Meca.DirecPP);
%             DirecPP=num2str(DirecPP);
            set(handles.DirecPP,'Data',DirecPP);
            DirecPRSD=Meca.DirecPRSD;
            assignin('base','DirecPRSD',Meca.DirecPRSD);
%             DirecPRSD=num2str(DirecPRSD);
            set(handles.DirecPRSD,'data',DirecPRSD);
            Radios=Meca.Radios;
            assignin('base','Radios',Meca.Radios);
%             Radios=num2str(Radios);
            set(handles.Radios,'data',Radios);

           Bien=imread('Bien.png');
           Frase='Archivo cargado con exito';
           msgbox(Frase,'Mensaje','custom',Bien);
           set(handles.VGDL,'Enable','on');
           set(handles.AGDL,'Enable','on');
       end
end

% --- Executes on selection change in Lista.
function Lista_Callback(hObject, eventdata, handles)
% hObject    handle to Lista (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns Lista contents as
cell array
%        contents{get(hObject,'Value')} returns selected item from Lista
Valor=get(handles.Lista,'Value');

if Valor==1
   set(handles.Nbody,'String','1');
   W=what;
   Directorio=W.path;
   Directorio=[Directorio '\'];
   mkdir([Directorio,'Guardar']);
```

```
end

if Valor==2
   set(handles.Nbody,'String','2');
end

% --- Executes during object creation, after setting all properties.
function Lista_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Lista (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --------------------------------------------------------------------
function Guardar_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to Guardar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

Nbody=get(handles.Nbody,'String');
if isempty(Nbody)
    warndlg('Rellene el campo Número de cuerpos','Mensaje');
else
    Nbody=str2double(Nbody);
end
Nrest=get(handles.Nrest,'String');
if isempty(Nrest)
    warndlg('Rellene el campo Número de pares de restricción','Mensaje');
else
    Nrest=str2double(Nrest);
end
NPR=get(handles.NPR,'String');
if isempty(NPR)
    warndlg('Rellene el campo Pares de revolución','Mensaje');
else
    NPR=str2double(NPR);
end
NPP=get(handles.NPP,'String');
if isempty(NPP)
    warndlg('Rellene el campo Pares prismáticos','Mensaje');
else
    NPP=str2double(NPP);
end
NPRSD=get(handles.NPRSD,'String');
if isempty(NPRSD)
    warndlg('Rellene el campo Pares de rodadura sin
deslizamiento','Mensaje');
else
    NPRSD=str2double(NPRSD);
end
DirecPP=get(handles.DirecPP,'Data');
if isempty(DirecPP) && NPP>=1
    warndlg('Rellene el campo Direcciones de los pares
prismáticos','Mensaje');
else
```

```
    if iscell(DirecPP)
       DirecPP=cell2mat(DirecPP);
    end
end
DirecPRSD=get(handles.DirecPRSD,'Data');
if isempty(DirecPRSD) && NPRSD>=1
    warndlg('Rellene el campo Direcciones de los pares de rodadura sin
deslizamiento','Mensaje');
else
    if iscell(DirecPRSD)
       DirecPRSD=cell2mat(DirecPRSD);
    end
end
Radios=get(handles.Radios,'Data');
if isempty(Radios) && NPRSD>=1
    warndlg('Rellene el campo Radios de curvatura','Mensaje');
else
    if iscell(Radios)
       Radios=cell2mat(Radios);
    end
end
Coord=get(handles.Coord,'Data');
if isempty(Coord)
    warndlg('Rellene el campo Coordenadas','Mensaje');
else
    if iscell(Coord)
       Coord=cell2mat(Coord);
    end
end
Lazo=get(handles.Lazo,'String');
if isempty(Lazo)
    warndlg('Rellene el campo Lazo','Mensaje');
else
    Lazo=str2num(Lazo);
end
LazoPares=get(handles.LazoPares,'String');
if isempty(LazoPares)
    warndlg('Rellene el campo Lazo de los pares','Mensaje');
else
    LazoPares=str2num(LazoPares);
end
VGDL=get(handles.VGDL,'String');
if isempty(VGDL)
    warndlg('Rellene el campo Velocidades iniciales','Mensaje');
else
    VGDL=str2num(VGDL)';
end
AGDL=get(handles.AGDL,'String');
if isempty(AGDL)
    warndlg('Rellene el campo Aceleraciones iniciales','Mensaje');
else
    AGDL=str2num(AGDL)';
end
Nombre=get(handles.Nombre,'String');


Save.Nombre=Nombre;
Save.Nbody=Nbody;
Save.Nrest=Nrest;
Save.NPR=NPR;
Save.NPP=NPP;
```

```
Save.NPRSD=NPRSD;
Save.VGDL=VGDL;
Save.AGDL=AGDL;
Save.Lazo=Lazo;
Save.LazoPares=LazoPares;
Save.Coord=Coord;
Save.DirecPP=DirecPP;
Save.DirecPRSD=DirecPRSD;
Save.Radios=Radios;

s=Guarda_Archivos(Save);
if s==0
    msgbox('Error al guardar archivo','Mensaje','error');
end
if s==1
    if isempty(Save.Nombre)
       Bien=imread('Bien.png');
       Frase='Archivo guardado con exito. Nombre:';
       Frase=[Frase ' Mecanismo'];
       msgbox(Frase,'Mensaje','custom',Bien);
    else
       Bien=imread('Bien.png');
       Frase='Archivo guardado con exito. Nombre:';
       Frase=[Frase Save.Nombre];
       msgbox(Frase,'Mensaje','custom',Bien);
    end
end



function Nombre_Callback(hObject, eventdata, handles)
% hObject    handle to Nombre (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Nombre as text
%        str2double(get(hObject,'String')) returns contents of Nombre as a
double


% --- Executes during object creation, after setting all properties.
function Nombre_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Nombre (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in ComprovacionesIniciales.
function ComprovacionesIniciales_Callback(hObject, eventdata, handles)
% hObject    handle to ComprovacionesIniciales (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

Error=0;
```

```
Nbody=get(handles.Nbody,'String');
if isempty(Nbody)
    warndlg('Rellene el campo Número de cuerpos','Mensaje');
    Error=1;
else
    Nbody=str2double(Nbody);
end
Nrest=get(handles.Nrest,'String');
if isempty(Nrest)
    warndlg('Rellene el campo Número de pares de restricción','Mensaje');
    Error=1;
else
    Nrest=str2double(Nrest);
end
NPR=get(handles.NPR,'String');
if isempty(NPR)
    warndlg('Rellene el campo Pares de revolución','Mensaje');
    Error=1;
else
    NPR=str2double(NPR);
end
NPP=get(handles.NPP,'String');
if isempty(NPP)
    warndlg('Rellene el campo Pares prismáticos','Mensaje');
    Error=1;
else
    NPP=str2double(NPP);
end
NPRSD=get(handles.NPRSD,'String');
if isempty(NPRSD)
    warndlg('Rellene el campo Pares de rodadura sin
deslizamiento','Mensaje');
    Error=1;
else
    NPRSD=str2double(NPRSD);
end
Lazo=get(handles.Lazo,'String');
if isempty(Lazo)
    warndlg('Rellene el campo Lazo','Mensaje');
    Error=1;
else
    Lazo=str2num(Lazo);
end
LazoPares=get(handles.LazoPares,'String');
if isempty(LazoPares)
    warndlg('Rellene el campo Lazo de los pares','Mensaje');
    Error=1;
else
    LazoPares=str2num(LazoPares);
end
VGDL=get(handles.VGDL,'String');
if isempty(VGDL)
    warndlg('Rellene el campo Velocidades iniciales','Mensaje');
    Error=1;
else
    VGDL=str2num(VGDL)';
end
AGDL=get(handles.AGDL,'String');
if isempty(AGDL)
    warndlg('Rellene el campo Aceleraciones iniciales','Mensaje');
    Error=1;
else
```

```
    AGDL=str2num(AGDL)';
end

if Error==0
  Meca=CompIni(Nbody,Nrest,NPR,NPP,NPRSD,Lazo,LazoPares,VGDL,AGDL);
  Bien=imread('Bien1.png');
  Mal=imread('Mal1.png');
   if Meca.CompIni.Comp1==1
      axes(handles.Comp1);
      image(Mal);
      axis off
   end
   if Meca.CompIni.Comp1==0
      axes(handles.Comp1);
      image(Bien);
      axis off
   end
   if Meca.CompIni.Comp2==1
      axes(handles.Comp2);
      image(Mal);
      axis off
   end
   if Meca.CompIni.Comp2==0
      axes(handles.Comp2);
      image(Bien);
      axis off
   end
   if Meca.CompIni.Comp3==1
      axes(handles.Comp3);
      image(Mal);
      axis off
   end
   if Meca.CompIni.Comp3==0
      axes(handles.Comp3);
      image(Bien);
      axis off
   end
   if Meca.CompIni.Comp4==1
      axes(handles.Comp4);
      image(Mal);
      axis off
   end
   if Meca.CompIni.Comp4==0
      axes(handles.Comp4);
      image(Bien);
      axis off
   end
end


% --------------------------------------------------------------------


% --- Executes when selected object is changed in uipanel5.
function uipanel5_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in uipanel5
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%   EventName: string 'SelectionChanged' (read only)
%   OldValue: handle of the previously selected object or empty if none was
selected
%   NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)
```

```
% --------------------------------------------------------------------
function Nuevo_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to Nuevo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

close all
Prueba11();
% Nada=imread('Nada.png');
% set(handles.Nombre,'String','');
% set(handles.Nbody,'String','');
% set(handles.Nrest,'String','');
% set(handles.NPR,'String','');
% set(handles.NPP,'String','');
% set(handles.NPRSD,'String','');
% set(handles.Lazo,'String','');
% set(handles.LazoPares,'String','');
% set(handles.VGDL,'String','');
% set(handles.AGDL,'String','');
% set(handles.TextVang,'String','');
% set(handles.TextVlin,'String','');
% set(handles.TextAang,'String','');
% set(handles.TextAlin,'String','');
% set(handles.Coord,'Data','');
% set(handles.DirecPP,'Data','');
% set(handles.DirecPRSD,'Data','');
% set(handles.Radios,'Data','');
%
%       axes(handles.Comp1);
%       image(Nada)
%       axis off
%       axes(handles.Comp2);
%       image(Nada)
%       axis off
%       axes(handles.Comp3);
%       image(Nada)
%       axis off
%       axes(handles.Comp4);
%       image(Nada)
%       axis off


% --------------------------------------------------------------------
function Cargar_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to Cargar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Nombre=get(handles.Nombre,'String');

if isempty(Nombre)
    warndlg('Introduzca el nombre del archivo a abrir en el campo: Nombre del
archivo','Mensaje');
else
    W=what;
    DirectorioA=W.path;
    DirectorioN=[DirectorioA '\Archivos guardados'];
```

```
    DirectorioN=[DirectorioN '\'];

    cd(DirectorioN);
    DirectorioF=[DirectorioN Nombre];
    DirectorioF=[DirectorioF '\'];
    E=exist(DirectorioF);
       if E==0
           Frase='No existe ningún archivo guardado con el nombre: ';
           Frase=[Frase Nombre];
           warndlg(Frase,'Mensaje');
           cd(DirectorioA);
       else
           cd(DirectorioA);
           DirectorioF=double(char(DirectorioF));
           Meca=Carga_Archivos(DirectorioF);
             cd(DirectorioA);
             Nbody=Meca.Nbody;
             set(handles.Nbody,'String',Nbody);
             Nrest=Meca.Nrest;
             set(handles.Nrest,'String',Nrest);
             NPR=Meca.NPR;
             set(handles.NPR,'String',NPR);
             NPP=Meca.NPP;
             set(handles.NPP,'String',NPP);
             NPRSD=Meca.NPRSD;
             set(handles.NPRSD,'String',NPRSD);
             Lazo=Meca.Lazo;
             set(handles.Lazo,'String',Lazo);
             LazoPares=Meca.LazoPares;
             set(handles.LazoPares,'String',LazoPares);
             VGDL=Meca.VGDL;
             set(handles.VGDL,'String',VGDL);
             AGDL=Meca.AGDL;
             set(handles.AGDL,'String',AGDL);
             Coord=Meca.Coord;
             assignin('base','Coord',Coord);
%              Coord=num2str(Coord);
             set(handles.Coord,'Data',Coord);
             DirecPP=Meca.DirecPP;
             assignin('base','DirecPP',Meca.DirecPP);
%              DirecPP=num2str(DirecPP);
             set(handles.DirecPP,'Data',DirecPP);
             DirecPRSD=Meca.DirecPRSD;
             assignin('base','DirecPRSD',Meca.DirecPRSD);
%              DirecPRSD=num2str(DirecPRSD);
             set(handles.DirecPRSD,'data',DirecPRSD);
             Radios=Meca.Radios;
             assignin('base','Radios',Meca.Radios);
%              Radios=num2str(Radios);
             set(handles.Radios,'data',Radios);

           Bien=imread('Bien.png');
           Frase='Archivo cargado con exito';
           msgbox(Frase,'Mensaje','custom',Bien);
           set(handles.VGDL,'Enable','on');
           set(handles.AGDL,'Enable','on');
       end
end

% --------------------------------------------------------------------
```

```
% --- Executes on button press in AyudaRC.
function AyudaRC_Callback(hObject, eventdata, handles)
% hObject    handle to AyudaRC (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of AyudaRC
AyudaRadios();

% --- Executes on button press in AyudaPRSD.
function AyudaPRSD_Callback(hObject, eventdata, handles)
% hObject    handle to AyudaPRSD (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of AyudaPRSD
AyudaDirecPRSD();


% --------------------------------------------------------------------
function Acercade_Callback(hObject, eventdata, handles)
% hObject    handle to Acercade (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
AcercaDe();


% --- Executes when figure1 is resized.
function figure1_ResizeFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function RefComp4_Callback(hObject, eventdata, handles)
% hObject    handle to RefComp4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
helpdlg('Esta comprobación repasa la correcta introducción de los vectores
Velocidades y Aceleraciones iniciales en cuanto a tamaño','Comprobación 1');


% --------------------------------------------------------------------
function RefComp1_Callback(hObject, eventdata, handles)
% hObject    handle to RefComp1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
helpdlg('Esta comprobación repasa la correcta introducción del Lazo de los
Pares y el número de restricciones del mecanismo','Comprobación 1');



% --------------------------------------------------------------------
function RefComp2_Callback(hObject, eventdata, handles)
% hObject    handle to RefComp2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
helpdlg('Esta comprobación repasa si coinciden el número de pares impuestos
con el número total de restricciones del mecanimo','Comprobación 1');


% --------------------------------------------------------------------
function RefComp3_Callback(hObject, eventdata, handles)
% hObject    handle to RefComp3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
helpdlg('Esta comprobación repasa la correcta introducción de los pares con
relación al Lazo de los Pares del mecanismo','Comprobación 1');
```

RELACIONES.M

```
function [Meca] = Relaciones(Meca)
%% Función que crea la subestructura Meca.Rela %%

   % Creación de Meca.Rela().Codig
      for i=[1:1:length(Meca.LazoPares)]
         Meca.Rela(i).Codig=Meca.LazoPares(i);
      end
   %
   % Creación de Meca.Rela().x y Meca.Rela().y
      for i=[1:1:Meca.Nrest]
          Meca.Rela(i).x=Meca.Coord(i,1);
          Meca.Rela(i).y=Meca.Coord(i,2);
      end
   %
   % Creación de Meca.Rela().h
      k=1;
      for i=[1:1:Meca.Nrest]
          if Meca.Rela(i).Codig==2
              h=Meca.DirecPP(k,:);
              hp=[-h(2);h(1)];
              Meca.Rela(i).hp=hp;
              k=k+1;
          end
      end
   %
   % Creación de Meca.Rela().Cuerpos
      for i=[1:1:Meca.Nrest]
          C1=Meca.Lazo(i);
          C2=Meca.Lazo(i+1);
          Meca.Rela(i).Cuerpos=[C1,C2];
      end
   %
   % Implantación de las condiciones del sólido 1
      Meca.Rela(1).Vx=0;
      Meca.Rela(1).Vy=0;
      Meca.Rela(Meca.Nrest).Vx=0;
      Meca.Rela(Meca.Nrest).Vy=0;
   %


end
```

RESUELVE.M

```
function [Meca] =
Resuelve(Nbody,Nrest,NPR,NPP,NPRSD,Lazo,LazoPares,Coord,DirecPP,Radios,DirecP
RSD,VGDL,AGDL,Prob)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Intento de implementación de resolución del problema de velocidad V6 %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Ventana de introducción de datos %%
Dibujo=0;
Simulacion=0;
NGDL=Nbody*3-Nrest*2-3;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Comprobaciones iniciales %%
   % Comprobación 1
      if length(LazoPares)+1~=length(Lazo) || length(Lazo)~=Nrest+1
          fprintf('\n --> >>ERROR<<  Comprobación 1\n')
          Comp1=1;
      else
          Comp1=0;
      end
   %
   % Comprobación 2
      if Nrest~=NPR+NPP+NPRSD
         fprintf('\n --> >>ERROR<<  Comprobación 2\n')
         Comp2=1;
      else
          Comp2=0;
      end
   %
   % Comprobación 3
   npr=0;npp=0;nprsd=0;
      for i=[1:1:length(LazoPares)]
         if LazoPares(i)==1
             npr=npr+1;
         end
         if LazoPares(i)==2
             npp=npp+1;
         end
         if LazoPares(i)==3
             nprsd=nprsd+1;
         end
      end
     if npr~=NPR || npp~=NPP || nprsd~=NPRSD
         fprintf('\n --> >>ERROR<<  Comprobación 3\n')
         Comp3=1;
     else
         Comp3=0;
     end
   %
   % Comprobación 4
      Cont=0;
      for i=[1:1:length(LazoPares)]
          if LazoPares(i)==2
              if i==1
              else
                  Cont=Cont+1;
              end
```

```
            end
        end
        if NPP==1
            if length(VGDL)~=Nbody-1;
                fprintf('\n --> <<ERROR>>  Comprobación 4\n')
                Comp4=1;
            else
                if length(VGDL)~=length(AGDL)
                   fprintf('\n --> <<ERROR>>  Comprobación 4\n')
                   Comp4=1;
                else
                    Comp4=0;
                end
            end
        else
            if length(VGDL)~=Nbody-1+Cont*2
               fprintf('\n --> <<ERROR>>  Comprobación 4\n')
               Comp4=1;
            else
                if length(VGDL)~=length(AGDL)
                    fprintf('\n --> <<ERROR>>  Comprobación 4\n')
                    Comp4=1;
                else
                    Comp4=0;
                end
            end
        end
    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Cálculos %%
    % Implantación de los datos en la matriz Meca
       Meca.Nbody=Nbody;
       Meca.Nrest=Nrest;
       Meca.NPR=NPR;
       Meca.NPP=NPP;
       Meca.NPRSD=NPRSD;
       Meca.NGDL=NGDL;
       Meca.VGDL=VGDL;
       Meca.AGDL=AGDL;
       Meca.Coord=Coord;
       Meca.DirecPP=DirecPP;
       Meca.Radios=Radios;
       Meca.DirecPRSD=DirecPRSD;
       Meca.Lazo=Lazo;
       Meca.LazoPares=LazoPares;
       Meca.CompIni.Comp1=Comp1;
       Meca.CompIni.Comp2=Comp2;
       Meca.CompIni.Comp3=Comp3;
       Meca.CompIni.Comp4=Comp4;
    %
    % Determinación de bucle abierto o cerrado
       Meca=BucleAC(Meca);
    %
    % Creación de la subestructura Meca.Rela
       Meca=Relaciones(Meca);
    %
    % Creación de los grupos para la matriz TH - Meca.Grupo().G
       Meca=CreaGrupos(Meca);
    %
    % Creación de los tramos Meca.Tramo().T
       Meca=CreaTramos(Meca);
    %
```

```
   % Diferenciación del caso de NPP=0 y NPP~=0
      if Meca.NPP==0
          % Caso particular de NPP=0
             Meca=NOPP(Meca);
          %
      else
          % Creación de la matriz TH
             Meca=MatrizTH(Meca);
          %
          % Creación de la matriz HV
             Meca=MatrizHV(Meca);
          %
          % Unión de las matrices TH y HV. Creación de la Matriz Meca.Q
             Q=[Meca.TH,-Meca.HV];
             Meca.Q=Q;
          %
          % Aplicación de la condición de par prismático (Wi=Wj) a Meca.Q
             Meca=CondPrismatico(Meca);
          %
      end
          % Creación de las lineas char para los GDL
              Meca=CreaChar(Meca);
          %
          % Aplicación de los GDL y creación de las matrices Meca.Qind y
Meca.Qdep
             Meca=CondVGDL(Meca);
          %
          % Resolución del problema de velocidad
            Sol=Meca.Qdep\-Meca.Qind;
            Sol=Meca.VGDL+Sol;
            Meca.SolVeloc=Sol;
            if Meca.NPP~=0
               Meca=CondPrisSolV(Meca);
            end
          %
   %

   % Aplicación del Método de las velocidades relativas
      Meca=VelocRela(Meca);
   %

 if Prob==2
   % Resolución del problema de aceleraciones
      Meca=ProbAcel(Meca);
   %

   % Aplicación del Método de las aceleraciones relativas
      Meca=AceleRela(Meca);
   %
 end

%%%%%%%%%%%%%%

%% Comprobaciones finales %%
   % Comprobador del resultado del problema de velocidad
         Meca=CompruebaVel(Meca);
         aux=sum(abs(Meca.CompruebaVel));
         ErrorV=log10(aux);
         if ErrorV==-Inf
             fprintf('\n--> Problema de velocidad correcto\n    con error
0\n')
         else
```

```
            if aux<10^-5
               fprintf('\n--> Problema de velociad correcto\n    con un
errror del 10^%d\n',ErrorV)
            else
               fprintf('\n--> Problema de velocidad\n    recuelto con algun
tipo de error\n')
            end
        end
   %
 if Prob==2
   % Comprobador del resultado del problema de aceleración
       Meca=CompruebaAce(Meca);
         aux=sum(abs(Meca.CompruebaAce));
         ErrorA=log10(aux);
         if ErrorA==-Inf
             fprintf('\n--> Problema de aceleración correcto\n    con error
0\n')
         else
            if aux<10^-5
               fprintf('\n--> Problema de aceleración correcto\n    con un
errror del 10^%d\n',ErrorA)
            else
               fprintf('\n--> Problema de aceleración\n    recuelto con
algun tipo de error\n')
            end
         end
   %
 end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Muestra de resultados %%
if Dibujo==1;
    axis([-15 15 -15 15])
    title('Mecanismo en la posición dada')
    % Muestreo de los pares del mecanismo
       MP=MuestraPares(Meca);
       if MP==1
       else
           fprintf('\n --> >>ERROR<< Muestreo de los pares\n')
       end
    %
    % Muestreo de los cuerpos del mecanismo
       MC=MuestraCuerpos(Meca);
       if MC==1
       else
           fprintf('\n --> >>ERROR<< Muestreo de los cuerpos\n')
       end
    %
    % Muestreo de las velocidades de cada Par
       MV=MuestraVelocidades(Meca);
       if MV==1
       else
           fprintf('\n --> >>ERROR<<  Muestreo de las velocidades\n')
       end
    %
    % Muestreo de las velocidades de cada Par
       MA=MuestraAceleraciones(Meca);
       if MA==1
       else
           fprintf('\n --> >>ERROR<<  Muestreo de las Aceleraciones\n')
```

```
        end
    %

end

%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Simulación %%
if Simulacion==1
    axis([-15 15 -15 15])
    title('Simulación en movimiento del mecanismo')
    Meca.ResTiempo=[];
    Meca.Error=0;
    for i=[1:1:9000]
        Meca.Iteracion=i;
       pause(0.05);
       cla
          % Muestreo de los pares del mecanismo
            MP=MuestraPares(Meca);
            if MP==1
            else
                fprintf('\n --> >>ERROR<< Muestreo de los pares\n')
            end
         %
         % Muestreo de los cuerpos del mecanismo
            MC=MuestraCuerpos(Meca);
            if MC==1
            else
               fprintf('\n --> >>ERROR<< Muestreo de los cuerpos\n')
            end
        %
        % Muestreo de las velocidades de cada Par
           MV=MuestraVelocidades(Meca);
           if MV==1
           else
             fprintf('\n --> >>ERROR<<  Muestreo de las velocidades\n')
           end
        %
        % Muestreo de las Aceleraciones de cada Par
           MA=MuestraAceleraciones(Meca);
             if MA==1
             else
               fprintf('\n --> >>ERROR<<  Muestreo de las Aceleraciones\n')
             end
        %
           Coord=[];
           Meca=NuevasCoord(Meca);
           for j=[1:1:Meca.Nrest]
              Coord=[Coord;Meca.Rela(j).x Meca.Rela(j).y];
           end

           Meca.ResTiempo=[Meca.ResTiempo;i
[Meca.Rela(3).Ax+Meca.Rela(3).Ay]];
           Meca=NuevasGDL(Meca);
           Meca.Coord=Coord;
           Meca=Calculador(Meca);
    end

end
%%%%%%%%%%%%%%%
end
```

TRANSLAZOPARES.M

```
function [W] = TransLazoPares(V)
%% Función que devuelve la transformación de lazo de los pares%%

   for i=[1:1:length(V)]
       W(i)=V(i);
       if V(i)==3
           W(i)=1;
       end
   end

end
```

VELOCRELA.M

```
function [Meca] = VelocRela(Meca)
%% Función que implementa el método de las velocidades relativas %%

   % Creación de Meca.Cuerpo(i).W
     Meca.Cuerpo(1).W=0;
      for i=[1:1:Meca.Nbody-1]
          Meca.Cuerpo(i+1).W=Meca.SolVeloc(i);
      end
   %

   % Implementación de Meca.SolVeloc=V41;V81..... en Meca.Rela().Vx Vy
      VPP=[];
      for i=[1:1:length(Meca.Grupo)]
          fin=Meca.Grupo(i).G(end);
          fin=Meca.Lazo(fin+1);
          if fin==1
          else
              VPP=[VPP,fin];
          end
      end
    if length(VPP)==0
        VPP=0;
    end
    VPP;
%     % Restrinción en caso de bucle abierto [Pruebas]
%        if Meca.Bucle==1
%            VPP(end)=[];
%        end
%     %
    Elim=1;
    k=1;
  if VPP==0
  else
    for i=[Meca.Nbody:1:length(Meca.SolVeloc)]
        if k>length(VPP)
        else
            R=VPP(k);
        if Elim==1
            Meca.Rela(R).Vx=Meca.SolVeloc(i);
            Elim=Elim+1;
        else
            Meca.Rela(R).Vy=Meca.SolVeloc(i);
            Elim=1;
            k=k+1;
        end
        end
    end
  end
   %

% %% Códgo nuevo V2 %%
% Meca.Rela(1).Vx=0;
% Meca.Rela(1).Vy=0;
% for i=[2:1:length(Meca.Rela)]
%     w=Meca.Cuerpo(i).W;
%     T=Meca.Tramo(i-1).T;
%     T=[-T(2);T(1)];
%     v=[Meca.Rela(i-1).Vx;Meca.Rela(i-1).Vy];
```

```
%     Vn=v+T.*w;
%     Meca.Rela(i).Vx=Vn(1);
%     Meca.Rela(i).Vy=Vn(2);
% end
% %% Código nuevo V2 final %%


%%% Código nuevo inicio %%%
MVR=zeros(2,Meca.Nbody);
   % Caso de no existir pares prismáticos NPP=0
      if Meca.NPP==0
          % Calcular matriz MT
            MT=[];
            for i=[1:1:length(Meca.Rela)]
               Rf=Meca.Lazo(i);
               Ri=Meca.Lazo(i+1);
               Posfin=[Meca.Rela(Rf).x;Meca.Rela(Rf).y];
               Posini=[Meca.Rela(Ri).x;Meca.Rela(Ri).y];
               T=Posfin-Posini;
               T=[-T(2);T(1)];
               MT=[MT,T];
            end
            for i=[1:1:length(MT(1,:))-1]
               W=Meca.Cuerpo(i+1).W;
               MT(:,i)=MT(:,i).*W;
            end
            MVR=-MT;
            MVR(:,end)=[];
            MVR=[[0;0],MVR];
          %
          % Igualación de los extremos a cero si es par de revolución
             if Meca.LazoPares(1)==1
                 MVR(:,1)=0;
             end
             if Meca.LazoPares(end)==1
                 MVR(:,end)=0;
             end
          %
          % Proceso de suma
             for i=[2:1:length(MVR(1,:))]
                 MVR(:,i)=MVR(:,i)+MVR(:,i-1);
             end
          %
          % Colocación MVR en Meca.Rela(i).Vx y Meca.Rela(i).Vy
                 MVR(:,1)=0;
                 MVR(:,end)=0;
             for i=[1:1:length(MVR(1,:))]
                 Meca.Rela(i).Vx=MVR(1,i);
                 Meca.Rela(i).Vy=MVR(2,i);
             end
          %
      end
   %
   % Caso de solo un par prismático NPP=1
      if Meca.NPP==1
          % Calcular matriz MT
            MT=[];
            for i=[1:1:length(Meca.Rela)]
               Rf=Meca.Lazo(i);
               Ri=Meca.Lazo(i+1);
               Posfin=[Meca.Rela(Rf).x;Meca.Rela(Rf).y];
```

```
            Posini=[Meca.Rela(Ri).x;Meca.Rela(Ri).y];
            T=Posfin-Posini;
            T=[-T(2);T(1)];
            MT=[MT,T];
         end
         for i=[1:1:length(MT(1,:))-1]
            W=Meca.Cuerpo(i+1).W;
            MT(:,i)=MT(:,i).*W;
         end
         MVR=-MT;
         MVR(:,end)=[];
         MVR=[[0;0],MVR];
      %
      % Igualación de los extremos a cero si es par de revolución
         if Meca.LazoPares(1)==1
             MVR(:,1)=0;
         end
         if Meca.LazoPares(end)==1
             MVR(:,end)=0;
         end
      %
      % Proceso de suma
         NVPP=[];
         for i=[1:1:length(Meca.LazoPares)]
              if Meca.LazoPares(i)==2
                 NVPP=[NVPP,i];
              end
         end
         % Suma-> Caso de NPP en primera posición
            if NVPP==1
                MVR=-MVR;
                for i=[length(MVR(1,:))-1:-1:1]
                    MVR(:,i)=MVR(:,i)+MVR(:,i+1);
                end
            end
         %
         % Suma-> Caso de NPP en última posición
            if NVPP==Meca.Nrest
                for i=[2:1:length(MVR(1,:))]
                    MVR(:,i)=MVR(:,i)+MVR(:,i-1);
                end
            end
         %
         MVR;
         % Suma-> Caso de NPP en cualquier posición
            if NVPP~=1 && NVPP~=Meca.Nrest
                for i=[2:1:NVPP]
                    MVR(:,i)=MVR(:,i)+MVR(:,i-1);
                end
                for i=[length(MVR(1,:))-1:-1:NVPP+1]
                    MVR(:,i)=-(MVR(:,i)+MVR(:,i+1));
                end
            end
         %
      %
      % Colocación MVR en Meca.Rela(i).Vx y Meca.Rela(i).Vy
             MVR(:,1)=0;
         for i=[1:1:length(MVR(1,:))]
             Meca.Rela(i).Vx=MVR(1,i);
             Meca.Rela(i).Vy=MVR(2,i);
         end
      %
end
```

```
%
% Caso de más de un par prismático NPP>=2
   if Meca.NPP>=2
       % Calcular matriz MT
         MT=[];
         for i=[1:1:length(Meca.Rela)]
            Rf=Meca.Lazo(i);
            Ri=Meca.Lazo(i+1);
            Posfin=[Meca.Rela(Rf).x;Meca.Rela(Rf).y];
            Posini=[Meca.Rela(Ri).x;Meca.Rela(Ri).y];
            T=Posfin-Posini;
            T=[-T(2);T(1)];
            MT=[MT,T];
         end
         for i=[1:1:length(MT(1,:))-1]
            W=Meca.Cuerpo(i+1).W;
            MT(:,i)=MT(:,i).*W;
         end
         for i=[1:1:length(VPP)]
            MT(:,VPP(i))=0;
         end
       %
       % Introducir las V calculadas de Meca.SolVeloc
          for i=[1:1:length(VPP)]
             MVR(1,VPP(i))=Meca.Rela(VPP(i)).Vx;
             MVR(2,VPP(i))=Meca.Rela(VPP(i)).Vy;
          end
       %
       MVR=MVR+MT;
       % Igualación de los extremos a cero si es par de revolución
          if Meca.LazoPares(1)==1
              MVR(:,1)=0;
          end
          if Meca.LazoPares(end)==1
              MVR(:,end)=0;
          end
       %
       % Proceso de suma hacia la izquierda
          NVPP=[];
          for i=[1:1:length(Meca.LazoPares)]
               if Meca.LazoPares(i)==2
                  NVPP=[NVPP,i];
               end
          end
          NVPP(end)=[];
          k=0;
          for i=[length(MVR(1,:))-1:-1:1]
              if length(NVPP)-k<=0
                  MVR(:,i)=MVR(:,i)+MVR(:,i+1);
              else
                  if i==NVPP(end-k)
                      k=k+1;
                  else
                      MVR(:,i)=MVR(:,i)+MVR(:,i+1);
                  end
              end
              MVR;
          end
       %
       % Colocación MVR en Meca.Rela(i).Vx y Meca.Rela(i).Vy
          if Meca.LazoPares(1)==1
              MVR(:,1)=0;
```

```
            end
            for i=[1:1:length(MVR(1,:))]
                Meca.Rela(i).Vx=MVR(1,i);
                Meca.Rela(i).Vy=MVR(2,i);
            end
          %
      end
   %
%%% Código nuevo final %%%


%%% Código antiguo inicio %%%
%
%
%    % Cálculo de Vx y Vy en cada Meca.Rela(i)
%        VPP;
%        MVR=zeros(2,Meca.Nrest);
%     if Meca.NPP>=2
%        for i=[1:1:length(VPP)]
%            R=VPP(i);
%            Vx=Meca.Rela(R).Vx;
%            Vy=Meca.Rela(R).Vy;
%            MVR(1,R)=Vx;
%            MVR(2,R)=Vy;
%        end
%        MVR;
%        MT=[];
%        for i=[1:1:length(Meca.Rela)]
%            Rf=Meca.Lazo(i);
%            Ri=Meca.Lazo(i+1);
%            Posfin=[Meca.Rela(Rf).x;Meca.Rela(Rf).y];
%            Posini=[Meca.Rela(Ri).x;Meca.Rela(Ri).y];
%            T=Posfin-Posini;
%            T=[-T(2);T(1)];
%            MT=[MT,T];
%        end
%        MT;
%        for i=[1:1:length(MT(1,:))-1]
%            W=Meca.Cuerpo(i+1).W;
%            MT(:,i)=MT(:,i).*W;
%        end
%        for i=[1:1:length(VPP)]
%            MT(:,VPP(i))=0;
%        end
%        MT;
%        % Copia hacia la izquierda
%        NVPP=[];
%        for i=[1:1:length(Meca.LazoPares)]
%            if Meca.LazoPares(i)==2
%                NVPP=[NVPP,i];
%            end
%        end
%
%           % Restrinción en caso de bucle abierto
%              if Meca.Bucle==1
%                  NVPP(end)=[];
%              end
%           %
%        NVPP;
%            k=0;
%            for i=[length(MVR(1,:))-1:-1:1]
%                if length(NVPP)-k<=0
%                        MVR(:,i)=MVR(:,i+1);
```

```
%                 else
%                     if i==NVPP(end-k)
%                         k=k+1;
%                     else
%                            MVR(:,i)=MVR(:,i+1);
%                     end
%                 end
%                 MVR;
%             end
%              MVR=MVR+MT;
%         %
%     end
%
%        % Caso en el que VPP=0
%         if VPP==0
%          MT=[];
%               for i=[1:1:length(Meca.Rela)]
%                   Rf=Meca.Lazo(i);
%                   Ri=Meca.Lazo(i+1);
%                   Posfin=[Meca.Rela(Rf).x;Meca.Rela(Rf).y];
%                   Posini=[Meca.Rela(Ri).x;Meca.Rela(Ri).y];
%                   T=Posfin-Posini;
%                   T=[-T(2);T(1)];
%                   MT=[MT,T];
%               end
%               for i=[1:1:length(MT(1,:))-1]
%                   W=Meca.Cuerpo(i+1).W;
%                   MT(:,i)=MT(:,i).*W;
%               end
%               MVR=MT;
%         end
%        %
%
% %    if Meca.LazoPares(1)==1
% %        MVR(:,1)=0;
% %    end
% % MVR(:,end)=[];
%    if Meca.LazoPares(end)==1
%        MVR(:,end)=0;
%    end
%
%         for i=[1:1:length(MVR(1,:))]
%             Vx=MVR(1,i);
%             Vy=MVR(2,i);
%             Meca.Rela(i).Vx=Vx;
%             Meca.Rela(i).Vy=Vy;
%         end
%    %

%%% Código antiguo final %%%


end
```