UNIVERSIDADE DA CORUÑA

Escola Politécnica Superior

Trabajo Fin de Grado
CURSO 2016/17

Simulador de carretilla elevadora

Grado en Ingeniería Mecánica

ALUMNA/O

Javier Marcote Vázquez

TUTORAS/ES

Daniel Dopico Dopico

Alberto Luaces Fernández

FECHA

Julio 2017

SIMULADOR DE CARRETILLA ELEVADORA

Resumen

Este trabajo de fin de grado consiste en el desarrollo de un simulador de una carretilla elevadora. El objetivo es la creación de una aplicación que permita observar el comportamiento de una máquina real e incluso familiarizarse con su uso mediante la simulación dinámica de sistemas multicuerpo. Esta disciplina está basada en la mecánica computacional y en los métodos numéricos para el cálculo de la dinámica de sistemas mecánicos complejos.

Con este fin, se ha modelizado una carretilla real según las especificaciones de un fabricante. Ésta, está compuesta por diversos sólidos interconectados por pares cinemáticos, dando lugar a un sistema con diversos grados de libertad de los cuales, el movimiento de unos es libre y el de otros es guiado mediante los mandos designados para el simulador. Además se han modelizado también diversos subsistemas como los que hacen referencia a los neumáticos y a la transmisión.

Este modelo ha sido preparado para su ejecución en tiempo real, integrándolo en un entorno de simulación real y permitiéndonos llevar a cabo diversas maniobras de prueba, en las cuales se incluyen detecciones de colisión entre sólidos con el fin de poder manipular una carga y posteriormente obtener resultados.

SIMULADOR DUNHA CARRETILLA ELEVADORA

Resumo

Este traballo de fin de grao consiste no desenvolvemento dun simulador dunha carretilla elevadora. O obxectivo é a creación dunha aplicación que permita observar o comportamento dunha máquina real e incluso o familiarizado co seu uso mediante a simulación dinámica de sistemas multicorpo. Esta disciplina baséase na mecánica computacional e nos métodos numéricos para o cálculo da dinámica de sistemas mecánicos complexos.

Con este fin, modelizouse unha carretilla real según as especificacións dun fabricante. Ésta, está composta por diversos sólidos interconectados por pares cinemáticos, dando lugar a un sistema con diversos graos de liberdade dos que, o movemento duns é libre e o de outros é guiado mediante os mandos designados para o simulador. Tamén se modelizaron diversos subsistemas como os que fan referencia ós neumáticos e á transmisión.

Este modelo foi preparado para a súa execución en tempo real, integrándoo nun entorno de simulación real e permitíndonos levar a cabo diversas maniobras de proba, nas que se inclúen detección de colisión entre sólidos có fin de poder manipular unha carga e posteriormente obter resultados.

.

FORKLIFT SIMULATOR

Abstract

This bachelor thesis involves the development of a forklift simulator. The objective is to create an application to observe the operation of a real machine and even learn its driving techniques through the dynamic simulation of multibody systems. This discipline is based on computational mechanics and numerical methods for calculating the dynamics of complex mechanical systems.

For this purpose, a real forklift has been modeled according to the specifications of a manufacturer.
The machine is composed of different solids interconnected by joints, resulting in a system with several degrees of freedom, being the motion of some of them free and the motion of some others is guided by the controls designated for the simulator. Moreover, different subsystems such as those of tires and transmission have also been modeled.

This model has been prepared for its real time execution in a real simulation environment. Different test maneuvers can be simulated, including collision detection between solids in order to manipulate loads and obtain results.

AGRADECIMIENTOS

En primer lugar, resaltar que el desarrollo de este trabajo en la Escuela Politécnica Superior de Ferrol ha sido una experiencia enriquecedora que finaliza con el cumplimiento de un objetivo.

Quiero agradecer su ayuda tanto a Daniel Dopico Dopico, como a Alberto Luaces Fernández, por su disposición, consejos y sus labores de dirección de este trabajo que han servido como motivación para poder llevarlo a cabo.

También me gustaría expresar mi gratitud al resto del grupo de investigación del Laboratorio de Ingeniería Mecánica (LIM), de la Universidad de Coruña por permitirme haber realizado el trabajo en sus instalaciones y orientarme en lo necesario.

Por último y recordando que no todo el trabajo se termina al salir del laboratorio, quiero mostrar mi agradecimiento más especial a mi familia y amigos por sus ánimos a lo largo de estos años, haciendo mención especial a mis padres y mi hermano.

A todos los que me habéis ayudado, GRACIAS.

UNIVERSIDADE DA CORUÑA

Escola Politécnica Superior

TRABAJO FIN DE GRADO

CURSO 2016/17

Simulador de carretilla elevadora

Grado en Ingeniería Mecánica

Memoria

MEMORIA SIMULADOR DE CARRETILLA ELEVADORA

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 INTRODUCCIÓN

El presente proyecto consiste en un simulador de una carretilla elevadora. El objetivo es el desarrollo de una aplicación que permita al usuario poder comprobar las prestaciones de la máquina real y familiarizarse y entrenarse en su uso. Pudiendo enfrentarse a situaciones que en la vida real podrían llegar a resultar peligrosas como maniobras que pongan al límite las prestaciones de la máquina en cuanto a vuelco o manejo de cargas pesadas.

1.1 Motivación

La idea surgió de un encargo de la empresa de fabricación de neumáticos Bridgestone Hispania S.A. al Laboratorio de Ingeniería Mecánica de la Universidad de la Coruña, que consistía en la simulación de una maniobra de vuelco real de una de máquina con resultado fatal para la vida del operario con objeto de ser empleada como prueba pericial en un juicio.

A partir de este trabajo se decidió realizar un simulador que fuese útil para el entorno empresarial puesto que un simulador de estas características permitiría formar al personal de empresas en el manejo de las carretillas elevadoras antes de que se enfrentase a situaciones reales, ya que se trata de un elemento básico a la hora de prevenir accidentes.

1.2 Antecedentes: Carretillas elevadoras

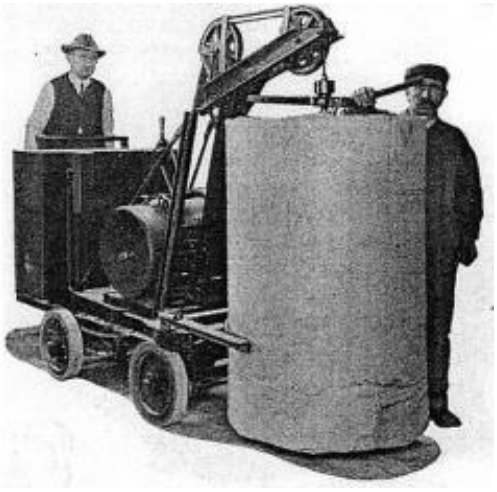

Fig 1.1 Carretilla elevadora de 1915

Se conoce que la historia de las carretillas elevadoras comienza en 1851, cuando Waterman idea el primer montacargas de la historia. Este sirvió de inspiración para Otis, que inventó el ascensor como un elevador con un sistema dentado que permitía amortiguar la caída en caso de rotura del cable. Con esta idea, en 1915 surgen las primeras carretillas elevadoras capaces de desplazar la carga tanto horizontal como verticalmente, solamente dos años más tarde de que surgieran los primeros carros capaces de moverse gracias a motores eléctricos alimentados por baterías. En este año, la plataforma diseñada podía bajar y subir las cargas gracias a un mecanismo de elevación de potencia. Pero es dos años más tarde, en 1917, cuando toma forma la idea de que el operario trabajara sentado en la propia carretilla.

Tres años más tarde, en 1920, ya se introdujo la energía hidráulica para elevar las cargas y en 1923, es donde se puede marcar el nacimiento de la carretilla elevadora tal y como se conoce hoy en día, provista de horquillas y un mástil elevador.

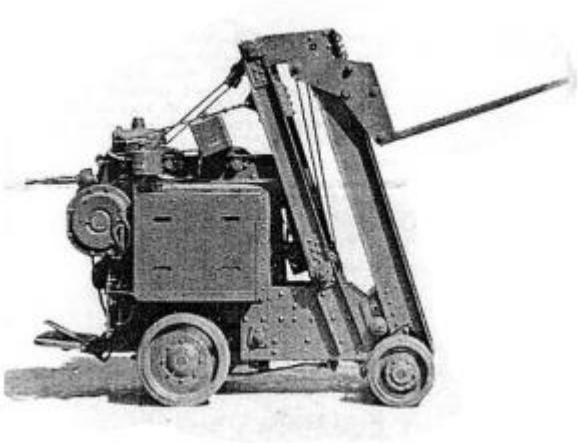

Fig 1.2 Carretilla elevadora de 1927

El siguiente gran avance no llegó hasta 1950, gracias a la normalización del pallet. En esta década, ya empezaba a surgir la necesidad de mover cargas cada vez más grandes y empezaron a surgir carretillas más potentes en las que se incorporaron nuevas tecnologías de la época como los motores diésel.

Fig 1.3 Carretilla elevadora de 1956

En 1960, empezó a preocupar la seguridad de los trabajadores y se introdujeron las jaulas de protección, puesto que hasta entonces las carretillas eran abiertas.

Veinte años después, en 1980, empezaron las preocupaciones medioambientales y se incorporaron a las carretillas nuevos motores como los de gas propano y gas natural comprimido.

A partir de esta fecha, las carretillas elevadoras han ido evolucionando a la par que el automóvil, evolucionando motores, reduciendo emisiones nocivas, y mejorando tanto en el aspecto electrónico como en el de la seguridad.

En la actualidad, cada vez son más las carretillas elevadoras eléctricas que se venden gracias a sus bajos consumos y menores mantenimientos. En lo que a diseño se refiere, aunque parece que no se ha avanzado los últimos años, se sigue mejorando en el uso de las funciones que hagan la carretilla más potente, más fácil de manejar y que tenga un mayor rendimiento. Un ejemplo de esto es que se está investigando la posibilidad de rotación sobre el chasis para mejorar la maniobrabilidad en espacios pequeños.

Pero las carretillas elevadoras, son solo uno de los múltiples tipos de carretillas que se pueden encontrar según su función. Estos tipos se pueden agrupar como:

- Porta-contenedores. Están especialmente diseñados y construidos para el manejo de contenedores en los trenes, zonas portuarias o camiones.
- Carretillas elevadoras (nuestro grupo). Son las máquinas más populares de manipulación y carga de material y se caracterizan por su capacidad de elevación. La carretilla permite la manipulación (levantamiento, transporte y apilamiento) de mercancías. Éstas se pueden utilizar para todas las aplicaciones posibles y de muchas maneras diferentes.
- Carretillas todo-terreno. Son máquinas de tracción total construidas y diseñadas para trabajar en instalaciones externas y en terrenos irregulares.

- Transpaletas/Transportes/Almacenajes. Son máquinas pequeñas que suelen ser operadas de pie, fáciles de utilizar, ágiles y flexibles lo que permite el transporte fácil y ligero de pallets y otras mercancías sin el esfuerzo manual. Pueden ser eléctricas, que están formadas por dos horquillas, ruedas traseras dobles, ruedas de carga en la parte delantera y un mango rígido operado por un aparato de elevación hidráulico a través del que las mercancías se pueden elevar para poder ser transportadas. Y también pueden ser de tijera, cuya principal diferencia es la altura de elevación gracias al mecanismo de tijera que incorporan.
- Recoge pedidos. Son utilizados principalmente para las operaciones de preparación de pedidos, en la cual el conductor está sentado o parado en una posición de conducción y los pedidos pueden ser elevados a la altura de trabajo requerido. Pueden ser de nivel bajo, medio o alto dependiendo de su altura de elevación.
- Almacenajes/Apiladores/Apiladores eléctricos. Son utilizados en la industria logística para apilar, teniendo opción manual o mecánica. Pueden ser con timón, con conductor incorporado, con conductor sentado y para pasillos estrechos o retráctiles.
- Autoportantes. Están diseñadas de tal manera que se pueden remolcar en la parte trasera del camión.
- Reach stacker. Similares a un porta contenedores pero con una gran capacidad gracias a su brazo telescópico.
- Manipulador telescópico. Son máquinas muy versátiles debido a la sección de fijación libre en la cabeza del brazo telescópico que permite la instalación de varios implementos. Se utilizan especialmente en el área de la construcción, la agricultura y los polígonos industriales.
- Carretilla de carga lateral de 4 vías. Están diseñadas para transportar y apilar objetos largos y voluminosos en espacios reducidos. Pueden ser de carga lateral o retráctil de 4 vías.

En cuanto a los simuladores, están a la orden del día en cursos de formación de carretilleros, existiendo desde algunos manejados desde la cabina de una carretilla real, hasta otros que incluso posibilitan la elección de tipo de carretilla y el manejo de sus mandos para una formación más amplia.

1.3 Descripción de la carretilla simulada

La carretilla elevadora de referencia elegida es la Linde H35D-02 de la serie 393-02, la cual admite una carga y una capacidad de pallets de hasta 3,5 t.

Fig 1.4 Fotografía de la carretilla elevadora Linde H35D-02

Con referencia al motor, tiene instalado un motor diésel de 4 cilindros y 4 tiempos, con turbocompresor y tecnología punta de inyección de la bomba. Acciona las bombas

hidráulicas de la carretilla elevadora a un régimen adecuado para la carga. El motor se refrigera por medio de un circuito cerrado de refrigeración con un depósito de expansión. Se usa una lubricación de circulación de presión con una bomba de aceite en el cárter de aceite para lubricar el motor. El aire de combustión se limpia por medio de un filtro de aire seco con un filtro de papel. Con este tipo de motores diésel se consigue un par elevado, un bajo consumo, bajas emisiones tanto de escape como de partículas y un bajo nivel de ruido.

La tracción consta de una bomba hidráulica de desplazamiento variable, dos motores de accionamiento hidráulico continuo para las ruedas (montados como una unidad de eje de accionamiento) y una bomba hidráulica (bomba de desplazamiento fijo) para el sistema hidráulico de trabajo y de la dirección. El sentido de la marcha y la velocidad se regulan mediante dos pedales aceleradores a través de la bomba hidráulica de desplazamiento variable (sentido de la marcha hacia delante y hacia atrás). Los motores de accionamiento hidráulico continuo de las ruedas se alimentan mediante la bomba hidráulica de desplazamiento variable y accionan las ruedas motrices.

La transmisión hidrostática se usa como freno de servicio. Esto quiere decir que el freno de servicio no requiere mantenimiento. Los dos frenos de discos múltiples incorporados en los motores de las ruedas se usan como freno de estacionamiento.

La dirección es un sistema hidrostático en el que el volante actúa sobre el cilindro de dirección para accionar las ruedas traseras.

El sistema eléctrico está alimentado por un alternador trifásico con una tensión de 12 V CC. Hay una batería de 12 V y 88 Ah instalada para arrancar el motor.

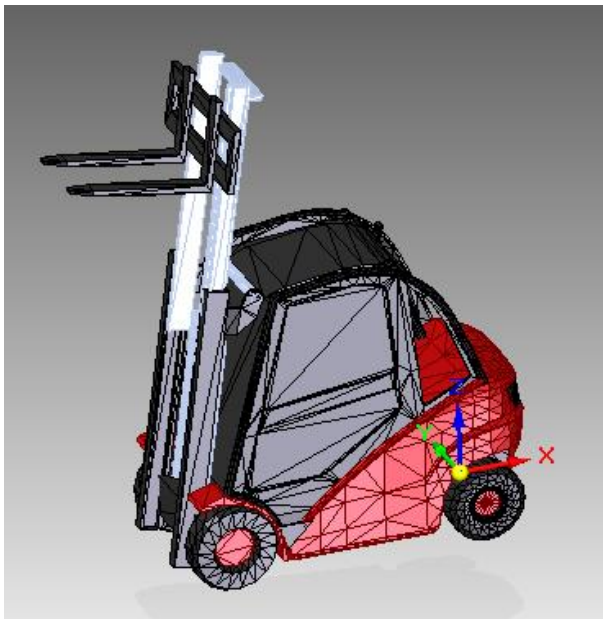

Fig 1.5 Diseño de la carretilla elevadora

Se pueden diferenciar las partes de la carretilla referenciadas en la siguiente figura:

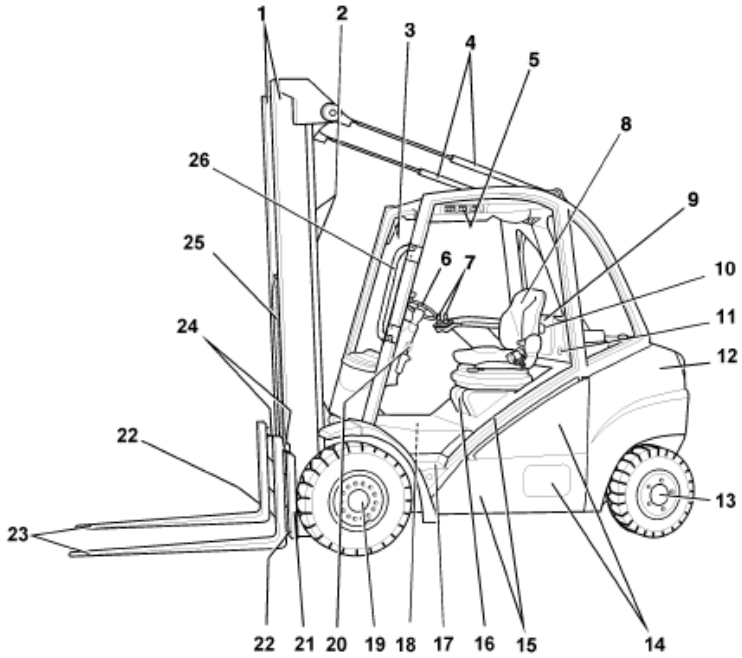

Fig 1.6 Partes de la carretilla elevadora Linde H35D-02

1 Mástil de elevación

2 Cilindro de elevación

3 Unidad de visualización

4 Cilindro de inclinación

5 Interruptor basculante para funciones adicionales (equipo especial)

6 Volante/dirección hidrostática

7 Palanca de accionamiento (palanca de mando)

8 Asiento del conductor

9 Tapa del sistema eléctrico

10 Fusibles (detrás de la tapa)

11 Enchufe de diagnóstico

12 Contrapeso

13 Eje de dirección

14 Tapas del compartimento de mantenimiento

15 Chasis con protector superior

16 Capó

17 Estribo para entrar y salir de la carretilla

18 Fusibles (en el compartimento del motor)

19 Motor de rueda izquierda

20 Tornillo de bloqueo para ajustar la columna de dirección

21 Portahorquillas

22 Dispositivos de protección de los brazos de horquilla

23 Brazos de horquilla

24 Fiador de los brazos de horquilla

25 Cadena del mástil (solo con mástiles doble y triple)

26 Asa para entrar y salir de la carretilla (equipo especial)

Y por último, obtenidos del manual de la carretilla que se puede ver en el Anexo II, se muestran los datos técnicos de ésta y sus dimensiones correspondientes:

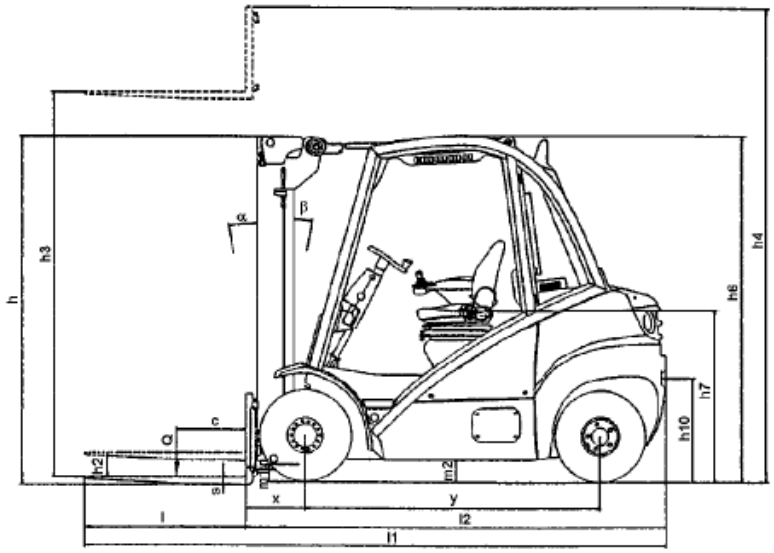

Fig 1.7 Medidas carretilla elevadora Linde H35D-02 (1)

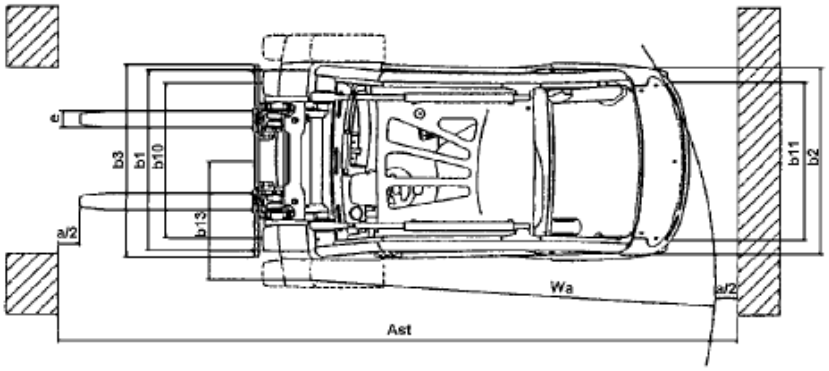

Fig 1.8 Medidas carretilla elevadora Linde H35D-02 (2)

1. Identificación			
1.1	Fabricante		Linde
1.2	Denominación de modelo del fabricante		H35D-02
1.3	Accionamiento		Diésel
1.4	Funcionamiento		Sentado
1.5	Capacidad de carga	Q [kg]	3.500
1.6	Centro de gravedad de la carga	c [mm]	500
1.7	Distancia de carga	x [mm]	450
1.8	Base de ruedas	y [mm]	1.965

Tabla 1.1 Datos técnicos carretilla. Identificación

2.Peso			
2.1	Tara	kg	4.680
2.2	Carga del eje con carga delantera/trasera	kg	7.240/940
2.3	Carga del eje sin carga delantera/trasera	kg	2.050/2.630

Tabla 1.2 Datos técnicos carretilla. Peso

3. Ruedas			
3.1	Neumáticos delanteros/traseros: poliuretano, goma		Superelásticos
3.2	Tamaño de neumáticos delanteros		27x10-12
3.3	Tamaño de neumáticos traseros		23x9-10
3.4	Número de ruedas, delanteras/traseras (x = impulsadas)		2x (4x)/2
3.5	Banda de rodadura delantera	b10 [mm]	1.008
3.6	Banda de rodadura trasera	b11 [mm]	932

Tabla 1.3 Datos técnicos carretilla. Ruedas

4. Dimensiones básicas			
4.1	Inclinación de las horquillas hacia delante/atrás	α/β (°)	5,0/8,0
4.2	Altura del mástil bajado	h1 [mm]	2264
4.3	Elevación libre	h2 [mm]	150
4.4	Elevación	h3 [mm]	3050
4.5	Altura del mástil levantado	h4 [mm]	3840
4.6	Altura del techo de protección del conductor (cabina)	h6 [mm]	2.210
4.7	Altura del asiento (mín./máx.)	h7 [mm]	1105
4.8	Altura de acoplamiento	h10 [mm]	695
4.9	Longitud total	l1 [mm]	3795
4.10	Longitud con la parte trasera de la horquilla incluida	l2 [mm]	2795
4.11	Anchura total	b1/b2 [mm]	1256
4.12	Dimensiones de los brazos de horquilla	s/e/l [mm]	50 x 120 x 1.000
4.13	Portahorquillas ISO 2328, clase/forma A, B		3A
4.14	Anchura del portahorquillas	b3 [mm]	1150
4.15	Distancia al suelo en el mástil	m1 [mm]	120
4.16	Distancia al suelo en el centro de la base de ruedas	m2 [mm]	170
4.17	Ancho del pasillo para el palé 1.000 x 1.200 transversal	Ast [mm]	4126
4.18	Ancho del pasillo para el palé 800 x 1.200 longitudinal	Ast [mm]	4326
4.19	Radio de giro	Wa [mm]	2476
4.20	Radio de pivote más pequeño	b13 [mm]	580

Tabla 1.4 Datos técnicos carretilla. Dimensiones básicas

5. Datos de rendimiento			
5.1	Velocidad de conducción con/sin carga	km/h	22/22
5.2	Velocidad de elevación con/sin carga	m/s	0,53/0,55
5.3	Velocidad de descenso con/sin carga	m/s	0,54/0,52
5.4	Fuerza de tracción con/sin carga	N	19.790/16.090
5.5	Capacidad permitida para subir pendientes con/sin carga	%	24/28
5.6	Aceleración con/sin carga	s	5,6/4,7
5.7	Freno de servicio		Hidrostático

Tabla 1.5 Datos técnicos carretilla. Datos de rendimiento

6. Accionamiento/Motor			
6.1	Modelo/fabricante del motor		VW/CPYB
6.2	Potencia nominal del motor de conformidad con ISO 1585	kW	44
6.3	Velocidad nominal	rpm	2700

6.4	Número de cilindros/Desplazamiento	cm^3	4/1.968
6.5	Consumo de combustible de conformidad con el ciclo VDI	l/h	3,4

Tabla 1.6 Datos técnicos carretilla. Accionamiento/Motor

7. Varios			
7.1	Tipo de controlador de tracción		Hidrostático/ c-v
7.2	Presión de funcionamiento para las fijaciones	bares	170
7.3	Volumen de aceite para las fijaciones	l/min	38
7.4	Nivel de ruido para el conductor de la carretilla	dB (A)	77
7.5	Gancho de remolque, clase/tipo		DIN 15170-H

Tabla 1.7 Datos técnicos carretilla. Varios

2 LA DINÁMICA DE SISTEMAS MULTICUERPO

Un sistema multicuerpo es el modelo de un conjunto de sólidos y sus restricciones. Con él se pueden estudiar la cinemática y la dinámica del mecanismo que forman.

Se modelizan los sólidos y las restricciones con unas coordenadas elegidas apropiadamente para conseguir determinar unívocamente la posición del sistema multicuerpo, una vez hecho esto, se plantean las ecuaciones de la cinemática y dinámica. Éstas, una vez resueltas proporcionan los resultados reales del movimiento del mecanismo, permitiendo calcular posiciones, velocidades, aceleraciones en cinemática o dinámica directa y fuerzas de todos los elementos y del conjunto en dinámica inversa.

2.1 Modelización en coordenadas mixtas

Las coordenadas son las variables que definirán completamente la posición del mecanismo. Son fundamentalmente, puntos, vectores, ángulos y distancias.

La elección de las coordenadas que van a definir la posición de la carretilla es un aspecto fundamental, ya que de ellas depende el resto del planteamiento del problema.

Entre los distintos tipos de coordenadas que existen para modelizar un mecanismo, las que resultan más ventajosas para nuestro modelo son las coordenadas naturales para la definición de sólidos y las coordenadas relativas para introducir los actuadores.

2.1.1 Las coordenadas naturales

Las coordenadas naturales son coordenadas cartesianas de los puntos y vectores unitarios, rígidamente unidos a los sólidos del mecanismo.

Las ventajas de dichas coordenadas son la definición simple y sistemática, la facilidad para establecer las ecuaciones de restricción, que se obtiene un número reducido de ecuaciones y que son sencillas puesto que no aparecen funciones trigonométricas (suelen ser restricciones lineales o a lo sumo cuadráticas).

El inconveniente que se puede citar es la familiarización necesaria con este tipo de coordenadas para el logro de modelizaciones correctas.

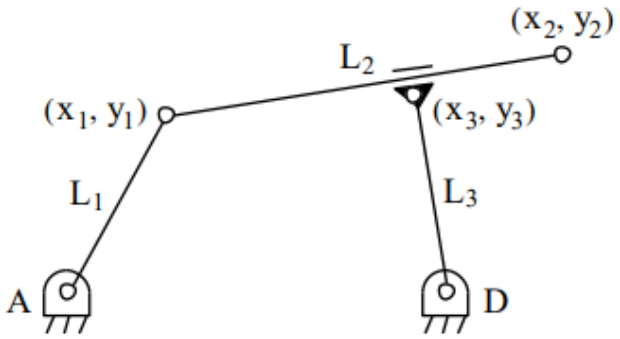

Fig 2.1 Coordenadas naturales

2.1.2 Las coordenadas relativas

Las coordenadas relativas son aquellas que sitúan cada elemento del mecanismo con respecto al anterior en la cadena cinemática, definiendo los pares cinemáticos con ecuaciones de restricción de ángulo y distancia. En cada par, serán necesarias tantas coordenadas como grados de libertad relativos permita el par entre los elementos que une.

Las ventajas son su reducido número y que facilitan la consideración de fuerzas y momentos aplicados en los pares.

Los inconvenientes son la dificultad de automatizar la determinación de los distintos lazos cinemáticos independientes del mecanismo, de manera que puedan establecerse las ecuaciones de restricción. Además, a partir de los valores numéricos de las coordenadas en un instante, no es posible determinar inmediatamente la posición de un elemento cualquiera, sino que hay que situar previamente a todos los elementos que le preceden en la cadena cinemática. Indicar también que las formulaciones a que dan lugar estas coordenadas, son complicadas y en la evaluación de los términos que resultan, aparecen numerosas funciones trigonométricas de elevado coste computacional, especialmente cuando se emplean formulaciones que conducen a ecuaciones del movimiento sin ninguna restricción.

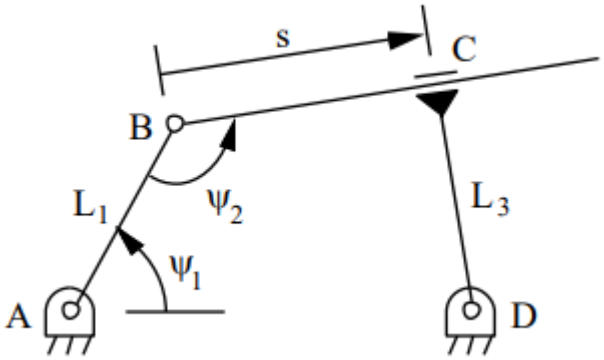

Fig 2.2 Coordenadas relativas

2.1.3 Las coordenadas mixtas

Las coordenadas mixtas son una combinación de las dos anteriores, aprovechan de las coordenadas naturales la definición sencilla y su carácter sistemático y de las coordenadas relativas aprovechan las facilidades que aportan para la consideración de fuerzas y momentos en los pares cinemáticos y la definición de ángulos y distancias para introducir los actuadores entre los sólidos.

El resultado es que se tendrán como variables todas las coordenadas cartesianas de puntos y vectores de las coordenadas naturales y adicionalmente, los ángulos y distancias definidos por las coordenadas relativas.

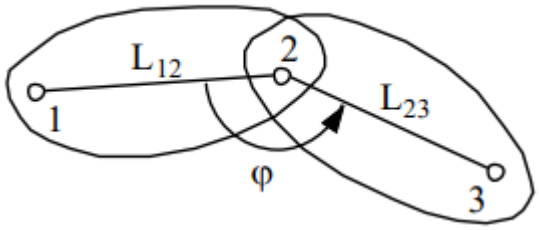

Fig 2.3 Coordenadas mixtas

2.2 El problema cinemático: problema de posición y velocidad inicial

La resolución del problema cinemático sirve para conocer el movimiento de todos los elementos del mecanismo, independientemente de las fuerzas que actúan sobre él. Al resolverlo, se obtiene el estado de las variables de modelización a partir de los valores conocidos de la posición y velocidad iniciales de los grados de libertad (GDL).

2.2.1 El problema de posición inicial

En el simulador sólo se efectúa el cálculo en el instante de tiempo inicial para conocer la posición de todos los elementos, de modo que para los valores iniciales de los GDL todas las variables del problema cumplan las restricciones impuestas.

El vector de variables es el de la ecuación (2.1), z_i son las variables que definen los GDL (ángulos y distancias).

$$\mathbf{q} = \left[q_1, q_2, ..., q_m, z_1, z_2, ..., z_n \right] \tag{2.1}$$

Dónde:

q_i : son las coordenadas cartesianas de puntos y vectores.

z_i : son las variables que definen los GDL (ángulos y distancias).

Todas estas variables están relacionadas por un vector de restricciones $\mathbf{\Phi(q)}$ que se puede ver en la ecuación (2.2), que representa un vector de ecuaciones no lineales de dimensión $m \times 1$. Desarrollando el vector $\mathbf{\Phi(q)}$ en serie de Taylor alrededor de la posición inicial y quedándose sólo con los términos de primer orden, se obtiene la ecuación (2.3).

$$\mathbf{\Phi(q)} = \mathbf{0} \tag{2.2}$$

$$\mathbf{\Phi(q)} = \mathbf{\Phi(q_0)} + \mathbf{\Phi_q(q_0)(q - q_0)} = \mathbf{0} \tag{2.3}$$

La resolución de la ecuación (2.3) se puede llevar a cabo mediante un proceso iterativo, llegando así a la ecuación (2.4)

$$\mathbf{\Phi_q(q_i)(q_{i+1} - q_i)} = -\mathbf{\Phi(q_i)} \tag{2.4}$$

Dónde $\mathbf{\Phi_q}$ es la matriz Jacobiana de las ecuaciones de restricción respecto a las variables. Nótese que los valores de $\mathbf{q}$; $z_1, z_2, ..., z_n$ correspondientes a los GDL son conocidos de antemano $z_j^{i+1} - z_j^i = 0$, por tanto, estos elementos, siempre van a ser multiplicados por elementos nulos y no se calculan. Se parte de un $\mathbf{q}_0$ estimado, cada vez que se añade un punto o un vector a un sólido se introduce la posición inicial aproximada del mismo en coordenadas globales.

Como criterio de convergencia se elige la norma-2 del vector de restricciones que se puede ver en la ecuación (2.5). Cuando en la iteración *i* ésta sea menor que una cierta tolerancia e_{max} se considera suficientemente aproximado el resultado.

$$e_i = \left\| \mathbf{\Phi}(\mathbf{q}_i) \right\| = \left(\sum_{j=1}^{m} \Phi_j^2 \right)^{\frac{1}{2}} \Bigg|_i \tag{2.5}$$

2.2.2 El problema de velocidad inicial

El problema de velocidad inicial consiste en determinar las derivadas temporales de todas las variables del mecanismo para el instante de tiempo inicial. Si se deriva la ecuación (2.2) respecto al tiempo, se obtiene:

$$\mathbf{\Phi_q}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{\Phi}_t = 0 \tag{2.6}$$

$$\mathbf{\Phi_q}(\mathbf{q})\dot{\mathbf{q}} = -\mathbf{\Phi}_t \tag{2.7}$$

Como la matriz Jacobiana ya es conocida del cálculo de posición inicial y las velocidades de los GDL son dato, es inmediato resolver la ecuación (2.6) y por tanto obtener las velocidades iniciales.

A diferencia del problema de posición inicial, el problema de velocidad inicial es lineal en las velocidades, como se puede ver en la ecuación (2.7) y por lo tanto, no iterativo.

2.3 El problema dinámico: Ecuaciones de la dinámica

La dinámica estudia la relación entre el movimiento de los cuerpos y las fuerzas que actúan sobre ellos. En el caso de un sistema mecánico, existen dos enfoques de esta cuestión que resultan interesantes, el problema dinámico directo, que consiste en obtener el movimiento del sistema conocidas las fuerzas que actúan sobre él y el problema dinámico inverso, en el que se trata de averiguar qué valor de las fuerzas es el que da lugar a un determinado movimiento.

2.3.1 Introducción a la resolución de problemas dinámicos

Para la resolución de los problemas dinámicos, se utilizan las ecuaciones de Lagrange ya que estas conducen a sistemas de tamaño mucho más reducido. Si se utilizasen las ecuaciones de Newton-Euler el número de ecuaciones sería seis veces el número de sólidos, mientras que aplicando las ecuaciones de Lagrange resulta un número proporcional a los GDL, generalmente muy inferior al anterior.

Utilizando las coordenadas mixtas explicadas anteriormente en el apartado 2.1.3 las ecuaciones de Lagrange son:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\mathbf{q}}}\right) - \frac{\partial T}{\partial \mathbf{q}} + \mathbf{\Phi}_\mathbf{q}^{\mathrm{T}}\boldsymbol{\lambda} = \mathbf{Q} \tag{2.8}$$

Y la ecuación (2.11) que se verá a continuación.

Dónde $\mathbf{\Phi}_\mathbf{q}^{\mathrm{T}}\boldsymbol{\lambda}$ son los esfuerzos requeridos para mantener las *m* restricciones entre las distintas variables dependientes y T es la energía cinética (función de la masa y la velocidad) con las coordenadas elegidas se puede escribir de la forma siguiente:

$$T = \frac{1}{2}\dot{\mathbf{q}}^{\mathrm{T}}\mathbf{M}\dot{\mathbf{q}} \tag{2.9}$$

Sustituyendo la ecuación (2.9) en la ecuación (2.8) se obtiene:

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{\Phi}_{\mathbf{q}}^{\mathrm{T}}\boldsymbol{\lambda} = \mathbf{Q} \tag{2.10}$$

Dónde:

$\mathbf{M}$: es la matriz de masas.
$\ddot{\mathbf{q}}$: es el vector de variables del problema.
$\mathbf{\Phi}_{\mathbf{q}}$: es el jacobiano del vector de restricciones.
$\boldsymbol{\lambda}$: es el vector de multiplicadores de Lagrange.
$\mathbf{Q}$: es el vector de fuerzas generalizadas.

El sistema de ecuaciones (2.10) consta de *n* ecuaciones donde *n* es el número de variables, sin embargo, existen *m* multiplicadores de Lagrange, tantos como ecuaciones de restricción, por tanto, es necesario añadir otras *m* ecuaciones, las ecuaciones de restricción. El resultado es que el sistema a resolver es una combinación de ecuaciones diferenciales y algebraicas (sistema DAE, differential algebraic equations).

$$\begin{cases} a) & \mathbf{M}\ddot{\mathbf{q}} + \mathbf{\Phi}_{\mathbf{q}}^{\mathrm{T}}\boldsymbol{\lambda} = \mathbf{Q} \\ b) & \mathbf{\Phi} = \mathbf{0} \end{cases} \tag{2.11}$$

La resolución directa del sistema (2.11) plantea multitud de problemas que se salen del ámbito de este trabajo, pero que desaconsejan el empleo de dichas ecuaciones directamente. Existen varias formulaciones para transformar y resolver el sistema de ecuaciones (2.11). Como la mayor parte de los integradores están pensados para sistemas de ecuaciones diferenciales ordinarias (ODE's, ordinary diferential equation), la mayoría buscan convertir el sistema DAE en un sistema ODE.

Se usará una formulación ALI3-P que se puede ver en el apartado 2.3.3 para resolver el problema dinámico en tiempo real durante la simulación, que necesita partir de un campo de aceleraciones previo, que será calculado con la formulación de los penalizadores descrita a continuación.

2.3.2 El primer problema dinámico: el problema de aceleración inicial

Para iniciar el integrador en la formulación ALI3-P se necesita calcular la aceleración inicial, ésta se calcula con el método de los penalizadores.

Se trata de resolver el sistema de ecuaciones (2.11) relacionando las variables del modelo con los multiplicadores de Lagrange, de este modo, se prescinde del término b) logrando un sistema con *m* ecuaciones menos.

Los multiplicadores están relacionados con las fuerzas que han de actuar en el sistema para que las ecuaciones de restricción que ligan las variables se cumplan en todo instante, el método de los penalizadores impone que los multiplicadores sean proporcionales al incumplimiento de las restricciones y sus derivadas.

Se sustituyen las restricciones por sistemas ficticios muelle-amortiguador con amortiguamiento crítico, obteniendo la siguiente expresión para $\boldsymbol{\lambda}$:

$$\boldsymbol{\lambda} = \boldsymbol{\alpha}\left(\ddot{\mathbf{\Phi}} + 2\xi\omega\dot{\mathbf{\Phi}} + \omega^2\mathbf{\Phi}\right) \tag{2.12}$$

Dónde:

$\boldsymbol{\alpha}$: es la matriz de penalizadores, es decir, la rigidez de los muelles correspondientes a cada restricción que usualmente toma valores entre 10^6 y 10^{11}.
ξ: es el amortiguamiento relativo, se le suele dar el valor ξ = 1 correspondiente al amortiguamiento crítico.
ω: es la frecuencia natural del sistema, se le suele dar el valor $\omega = 10$.

Introduciendo la ecuación (2.12) en la ecuación (2.10) se obtiene:

$$\mathbf{M}\ddot{\mathbf{q}} + \boldsymbol{\Phi}_{\mathbf{q}}^{\mathrm{T}}\boldsymbol{\alpha}\left(\ddot{\boldsymbol{\Phi}} + 2\xi\omega\dot{\boldsymbol{\Phi}} + \omega^2\boldsymbol{\Phi}\right) = \mathbf{Q} \tag{2.13}$$

$\ddot{\boldsymbol{\Phi}}$ se calcula derivando dos veces el vector de restricciones con respecto al tiempo:

$$\dot{\boldsymbol{\Phi}} = \frac{d\boldsymbol{\Phi}}{dt} = \frac{\left(\frac{\partial\boldsymbol{\Phi}}{\partial\mathbf{q}}dq + \frac{\partial\boldsymbol{\Phi}}{\partial t}dt\right)}{dt} = \boldsymbol{\Phi}_{\mathbf{q}}\dot{\mathbf{q}} + \boldsymbol{\Phi}_t \tag{2.14}$$

Dónde $\boldsymbol{\Phi}_t$ es el vector de derivadas parciales temporales de las restricciones.

Finalmente se obtiene:

$$\ddot{\boldsymbol{\Phi}} = \boldsymbol{\Phi}_{\mathbf{q}}\ddot{\mathbf{q}} + \dot{\boldsymbol{\Phi}}_{\mathbf{q}}\dot{\mathbf{q}} + \dot{\boldsymbol{\Phi}}_t \tag{2.15}$$

Sustituyendo la ecuación (2.15) en la ecuación (2.13) y reagrupando términos se obtiene la ecuación (2.16), las ecuaciones dinámicas del método de los penalizadores, las incógnitas de este sistema son únicamente las aceleraciones $\ddot{\mathbf{q}}$.

$$\left(\mathbf{M} + \boldsymbol{\Phi}_{\mathbf{q}}^{\mathrm{T}}\boldsymbol{\alpha}\boldsymbol{\Phi}_{\mathbf{q}}\right)\ddot{\mathbf{q}} = \mathbf{Q} - \boldsymbol{\Phi}_{\mathbf{q}}^{\mathrm{T}}\boldsymbol{\alpha}\left(\dot{\boldsymbol{\Phi}}_{\mathbf{q}}\dot{\mathbf{q}} + \dot{\boldsymbol{\Phi}}_t + 2\xi\omega\dot{\boldsymbol{\Phi}} + \omega^2\boldsymbol{\Phi}\right) \tag{2.16}$$

2.3.3 El problema dinámico en tiempo real: Formulación ALI3-P

En el método anterior los valores de los multiplicadores de Lagrange se calculan basándose en una aproximación, suficiente para iniciar la formulación de Lagrange aumentado en index-3 con proyecciones ortogonales (Aumented Lagrangian index-3 with ortogonal projections, ALI3-P), que calculará los valores exactos de los mismos.

La expresión de la formulación ALI3-P es:

$$\mathbf{M}\ddot{\mathbf{q}} + \boldsymbol{\Phi}_{\mathbf{q}}^{\mathrm{T}}\boldsymbol{\alpha}\boldsymbol{\Phi} + \boldsymbol{\Phi}_{\mathbf{q}}^{\mathrm{T}}\boldsymbol{\lambda}^{*} = \mathbf{Q} \tag{2.17}$$

Dónde:

$\mathbf{M}$: es la matriz de masas.
$\ddot{\mathbf{q}}$: es el vector de variables del problema.
$\boldsymbol{\Phi}_{\mathbf{q}}$: es la matriz Jacobiana del vector de restricciones.
$\boldsymbol{\lambda}^{*}$: es el vector de multiplicadores de Lagrange aproximados.
$\mathbf{Q}$: es el vector de fuerzas generalizadas.

Los multiplicadores de Lagrange son obtenidos por un proceso iterativo

$$\boldsymbol{\lambda}_{i+1}^{*} = \boldsymbol{\lambda}_i^{*} + \boldsymbol{\alpha}\boldsymbol{\Phi}_{i+1}, \quad i = 0,1,2... \tag{2.18}$$

Dónde $\boldsymbol{\lambda}_0^*$ se inicializa con valor nulo en la primera iteración en tiempo *t*=0, en los siguientes pasos de tiempo *t=n Δt,* el valor de $\boldsymbol{\lambda}_0^*$ será el valor de $\boldsymbol{\lambda}_{n-1}^*$, cuando el proceso converja, se obtendrán los valores finales de $\boldsymbol{\lambda}_n^*$ que convergen a los valores del sistema DAE original para $\boldsymbol{\lambda} \to \infty$.

El integrador empleado es la regla trapezoidal implícita de paso simple. Las ecuaciones en velocidades y aceleraciones son las siguientes:

$$\dot{\mathbf{q}}_{n+1} = \frac{2}{\Delta t}\mathbf{q}_{n+1} + \hat{\dot{\mathbf{q}}}_n \quad con \quad \hat{\dot{\mathbf{q}}}_n = -\left(\frac{2}{\Delta t}\mathbf{q}_n + \dot{\mathbf{q}}_n\right) \tag{2.19}$$

$$\ddot{\mathbf{q}}_{n+1} = \frac{4}{\Delta t^2}\mathbf{q}_{n+1} + \hat{\ddot{\mathbf{q}}}_n \quad con \quad \hat{\ddot{\mathbf{q}}}_n = -\left(\frac{4}{\Delta t^2}\mathbf{q}_n + \frac{4}{\Delta t}\dot{\mathbf{q}}_n + \ddot{\mathbf{q}}_n\right) \tag{2.20}$$

En la ecuación (2.20) se puede observar el término $\ddot{\mathbf{q}}_n$ es perteneciente al paso de tiempo anterior, lo que justifica el cálculo del campo de aceleraciones iniciales con el método de los penalizadores, en los siguientes pasos de tiempo $t = (n+1)\Delta t$ se tomará el vector de aceleraciones correspondiente al estado $t = ndt$.

Introduciendo las ecuaciones (2.19) y (2.20) en la ecuación (2.17) se obtiene el sistema:

$$\mathbf{M}\left(\frac{4}{\Delta t^2}\mathbf{q}_{n+1} + \hat{\ddot{\mathbf{q}}}_n\right) + \boldsymbol{\Phi}_{q_{n+1}}^{\mathrm{T}}\left(\boldsymbol{\alpha}\boldsymbol{\Phi}_{n+1} + \boldsymbol{\lambda}_{n+1}\right) - \mathbf{Q}_{n+1} = 0 \tag{2.21}$$

$$\frac{4}{\Delta t^2}\mathbf{M}\mathbf{q}_{n+1} + \mathbf{M}\hat{\ddot{\mathbf{q}}}_n + \boldsymbol{\Phi}_{q_{n+1}}^{\mathrm{T}}\left(\boldsymbol{\alpha}\boldsymbol{\Phi}_{n+1} + \boldsymbol{\lambda}_{n+1}\right) - \mathbf{Q}_{n+1} = 0 \tag{2.22}$$

Escalando por $\frac{4}{\Delta t^2}$ se obtiene:

$$\mathbf{M}\mathbf{q}_{n+1} + \frac{\Delta t^2}{4}\left(\mathbf{M}\hat{\ddot{\mathbf{q}}}_n + \boldsymbol{\Phi}_{q_{n+1}}^{\mathrm{T}}\boldsymbol{\alpha}\boldsymbol{\Phi}_{n+1} + \boldsymbol{\Phi}_{q_{n+1}}^{\mathrm{T}}\boldsymbol{\lambda}_{n+1} - \mathbf{Q}_{n+1}\right) = 0 \tag{2.23}$$

Para resolver el sistema no lineal (2.21) se usa el método iterativo de Newton-Raphson.

$$\left[\frac{\partial \mathbf{f}(\mathbf{q})}{\partial \mathbf{q}}\right]_i \Delta\mathbf{q}_i = -\left[\mathbf{f}(\mathbf{q})\right]_i \tag{2.24}$$

Siendo:

$$\left[\mathbf{f}(\mathbf{q})\right] = \frac{\Delta t^2}{4}\left(\mathbf{M}\ddot{\mathbf{q}} + \boldsymbol{\Phi}_q^{\mathrm{T}}\boldsymbol{\alpha}\boldsymbol{\Phi} + \boldsymbol{\Phi}_q^{\mathrm{T}}\boldsymbol{\lambda}^* - \mathbf{Q}\right) \tag{2.25}$$

La matriz tangente exacta es complicada de obtener, por tanto se utiliza la matriz tangente aproximada siguiente:

$$\left[\frac{\partial \mathbf{f}(\mathbf{q})}{\partial \mathbf{q}}\right] = \mathbf{M} + \frac{\Delta t}{2}\mathbf{C} + \frac{\Delta t^2}{4}\left(\boldsymbol{\Phi}_q^{\mathrm{T}}\boldsymbol{\alpha}\boldsymbol{\Phi}_q + \mathbf{K}\right) \tag{2.26}$$

Dónde:

K : es la contribución de las fuerzas elásticas.

$\mathbf{C}$: es la contribución de las fuerzas de amortiguamiento.

Este proceso conduce a una serie de posiciones $\mathbf{q}_{n+1}$ que satisfacen las ecuaciones del movimiento de la ecuación (2.21) y las ecuaciones de restricción, sin embargo, es posible que no se cumpla $\dot{\mathbf{\Phi}} = 0$ y $\ddot{\mathbf{\Phi}} = 0$, para solventar este problema se utilizan las proyecciones ortogonales en velocidades y aceleraciones.

Si $\dot{\mathbf{q}}^*$ y $\ddot{\mathbf{q}}^*$ son las velocidades obtenidas después de la convergencia del método de Newton-Raphson, las velocidades y aceleraciones depuradas se calculan según las siguientes expresiones:

$$\left[\mathbf{M}+\frac{\Delta t}{2}\mathbf{C}+\frac{\Delta t^2}{4}\left(\mathbf{\Phi}_q^{\mathrm{T}}\boldsymbol{\alpha}\mathbf{\Phi}_q+\mathbf{K}\right)\right]\dot{\mathbf{q}}= \\ =\left[\mathbf{M}+\frac{\Delta t}{2}\mathbf{C}+\frac{\Delta t^2}{4}\mathbf{K}\right]\dot{\mathbf{q}}^*-\frac{\Delta t^2}{4}\mathbf{\Phi}_q^{\mathrm{T}}\boldsymbol{\alpha}\mathbf{\Phi}_t \tag{2.27}$$

$$\left[\mathbf{M}+\frac{\Delta t}{2}\mathbf{C}+\frac{\Delta t^2}{4}\left(\mathbf{\Phi}_q^{\mathrm{T}}\boldsymbol{\alpha}\mathbf{\Phi}_q+\mathbf{K}\right)\right]\ddot{\mathbf{q}}= \\ =\left[\mathbf{M}+\frac{\Delta t}{2}\mathbf{C}+\frac{\Delta t^2}{4}\left(\mathbf{K}\right)\right]\ddot{\mathbf{q}}^*-\frac{\Delta t^2}{4}\mathbf{\Phi}_q^{\mathrm{T}}\boldsymbol{\alpha}\left(\dot{\mathbf{\Phi}}_q\dot{\mathbf{q}}+\dot{\mathbf{\Phi}}_t\right) \tag{2.28}$$

2.4 Modelización elegida para la carretilla elevadora

El modelo multicuerpo se desarrolló con el software MBSLIM, que es una biblioteca para la simulación dinámica de sistemas multicuerpo genéricos, desarrollada desde el año 2007 en el Laboratorio de Ingeniería Mecánica de la Universidad de A Coruña.

Se han modelizado todos los sólidos con cuatro entidades (puntos y/o vectores) de modo que con ellos se puedan representar un espacio tridimensional, es decir, que se pueda construir con ellos un sistema de referencia formado por un punto y tres vectores linealmente independientes.

Se puede demostrar que definiendo los sólidos con estos elementos, la matriz de masas es constante durante toda la simulación, no aparecerán en ella términos de fuerzas de inercia dependientes de la velocidad y sólo será necesario calcularla en el inicio de la simulación. Además las fuerzas generalizadas gravitatorias son también constantes y por lo tanto calculadas igualmente una sola vez al inicio de la simulación.

En coordenadas naturales se utilizan las siguientes restricciones para definir un sólido rígido:

- Distancia: para puntos se impone que la distancia entre dos puntos sea constante, en el caso de vectores, la norma del vector ha de ser la unidad. Es una sola ecuación.

$$\left(\mathbf{p}_i-\mathbf{p}_j\right)^{\mathrm{T}}\left(\mathbf{p}_i-\mathbf{p}_j\right)-l_{ij}^2=0 \tag{2.29}$$

$$\mathbf{v}_i^{\mathrm{T}}\mathbf{v}_i-1=0 \tag{2.30}$$

- Producto escalar constante: el ángulo entre dos vectores se mantendrá constante, si el sólido se define con más de un punto se tomarán como vectores los que unen esos puntos. Es una sola ecuación.

$$\mathbf{v}_i^{\mathrm{T}}\mathbf{v}_j - k_k = 0 \tag{2.31}$$

$$\left(\mathbf{p}_i - \mathbf{p}_j\right)^{\mathrm{T}}\mathbf{v}_k - k_l = 0 \tag{2.32}$$

$$\left(\mathbf{p}_i - \mathbf{p}_j\right)^{\mathrm{T}}\left(\mathbf{p}_i - \mathbf{p}_j\right) - k_m = 0 \tag{2.33}$$

- Combinación lineal: los elementos de modelización del solido forman un subespacio. Los puntos y vectores definidos a mayores para facilitar la definición del modelo pertenecerán a ese subespacio y además su posición en coordenadas locales permanecerá constante el tiempo, es decir los valores de los parámetros de combinación lineal serán constantes.
  Con esta restricción ocurre lo mismo que con la de producto escalar, si se modeliza con más de un punto se tomará como vector el que los une.
  Se trata de una ecuación vectorial lo que equivale a tres ecuaciones.

$$\mathbf{p}_i - \mathbf{p}_j - \alpha_i \cdot \mathbf{v}_l - \beta_i \cdot \mathbf{v}_m - \gamma_i \cdot \mathbf{v}_n = 0 \tag{2.34}$$

$$\mathbf{v}_i - \alpha_i \cdot \mathbf{v}_l - \beta_i \cdot \mathbf{v}_m - \gamma_i \cdot \mathbf{v}_n = 0 \tag{2.35}$$

  Donde $\mathbf{p}_i$ y $\mathbf{v}_i$ son el punto o vector extra que no pertenece a la definición del subespacio del sólido.

Las restricciones anteriores son introducidas automáticamente al definir el sólido en la MBSLIM pero faltaría añadir las restricciones adicionales que no son introducidas automáticamente por el solver.

Para definir los GDL se utilizan coordenadas relativas, creando así las variables adicionales que permitirán el guiado, esto se explicará en el apartado 2.4.3.

En la nomenclatura que se utiliza en la definición de los sólidos que han sido modelizados, en vez de las coordenadas de los puntos, se utilizarán vectores de posición siendo así:

$$\mathbf{r}_{i,j} = \mathbf{p}_j - \mathbf{p}_i \tag{2.36}$$

A continuación se muestran los sólidos definidos en el modelo.

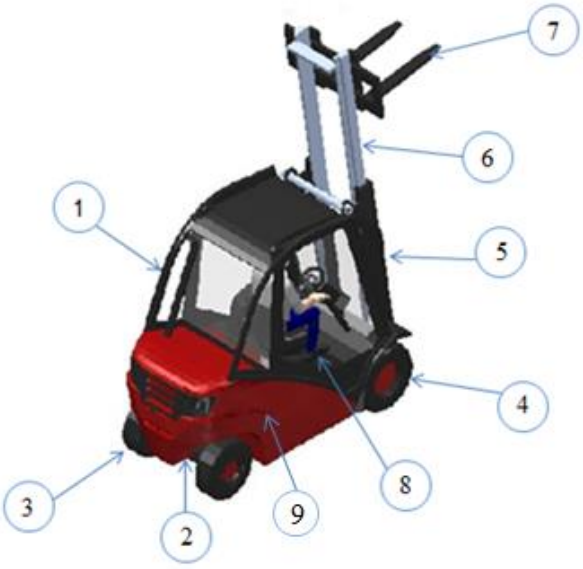

Fig 2.4 Sólidos del modelo

Número de sólido	Nombre	Cantidad
1	Chasis y cabina	1
2	Eje basculante	1
3	Ruedas traseras	2
4	Ruedas delanteras	2
5	Mástil 1	1
6	Mástil 2	1
7	Uñas y útil	1
8	Conductor	1
9	Combustible	1

Tabla 2.1 Sólidos del modelo

2.4.1 Definición del suelo

El suelo se define al cargarse la geometría del archivo edificio_suelo.obj. Este archivo incluye el suelo, el cual la MBSmodel, de la que se hablará en el apartado 4, detecta las colisiones de los sólidos de la carretilla y las distintas cargas introducidas con la malla que lo forma.

También se ha definido el suelo para cuando se quieren realizar maniobras de prueba con los actuadores de la carretilla. Para esto, se fija el chasis al suelo de forma que se puedan manejar los actuadores con la carretilla en reposo. En este modo, se define el suelo con el punto $\mathbf{p}_3$ y los vectores unitarios $\mathbf{v}_1, \mathbf{v}_2$ y $\mathbf{v}_3$ representando a $\mathbf{i},\mathbf{j},\mathbf{k}$ en ejes globales, que forman parte del chasis y creando así el plano del suelo. Tanto este punto como los vectores en este caso serán fijos y por ello no pasarán a formar parte del vector de variables.

2.4.2 Definición de sólidos de la carretilla

Ya se han visto los diferentes sólidos que componen la carretilla elevadora en la Fig 2.1, por lo que estos serán definidos en el mismo orden.

- **Chasis y cabina**

El chasis y la cabina se han modelado como un único sólido rígido con las dimensiones proporcionadas por el fabricante en el catálogo de la máquina (ver Anexo II).

Fig 2.5 Sólido 1 de la carretilla. Chasis y cabina

Como variables para definir el chasis se han utilizado:

- Seis puntos: $\mathbf{p}_3, \mathbf{p}_{15}, \mathbf{p}_{30}, \mathbf{p}_{40}, \mathbf{p}_{54}$ *y* $\mathbf{p}_{55}$
- Cuatro vectores: $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ *y* $\mathbf{v}_4$

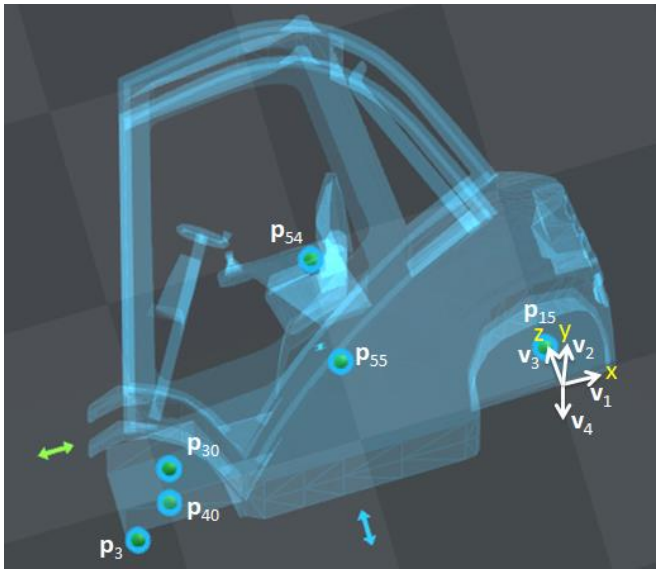

Fig 2.6 Esquema de puntos y vectores del chasis y la cabina

Cada punto con los que se representa el modelo, tiene una posición concreta que representa en cada caso: $\mathbf{p}_3$ punto del par de revolución con el mástil, $\mathbf{p}_{15}$ articulación del eje basculante trasero, $\mathbf{p}_{30}$ centro de la rueda delantera derecha, $\mathbf{p}_{40}$ centro de la rueda delantera izquierda, $\mathbf{p}_{54}$ ubicación del conductor y $\mathbf{p}_{55}$ ubicación del combustible.

Por su parte los vectores unitarios $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ *y* $\mathbf{v}_4$, representan a $\mathbf{i},\mathbf{j},\mathbf{k},-\mathbf{j}$ en ejes globales respectivamente.

Para modelizar el chasis y cabina, se han utilizado un punto y tres vectores, $\mathbf{p}_3, \mathbf{v}_1, \mathbf{v}_2$ *y* $\mathbf{v}_3$ en este mismo orden, ya que la MBSLIM construye el triedro basándose en este orden. En el Anexo I, en el fichero lectdatos.f90, se puede ver este mismo orden de introducción, quedando así el triedro definido de la siguiente forma:

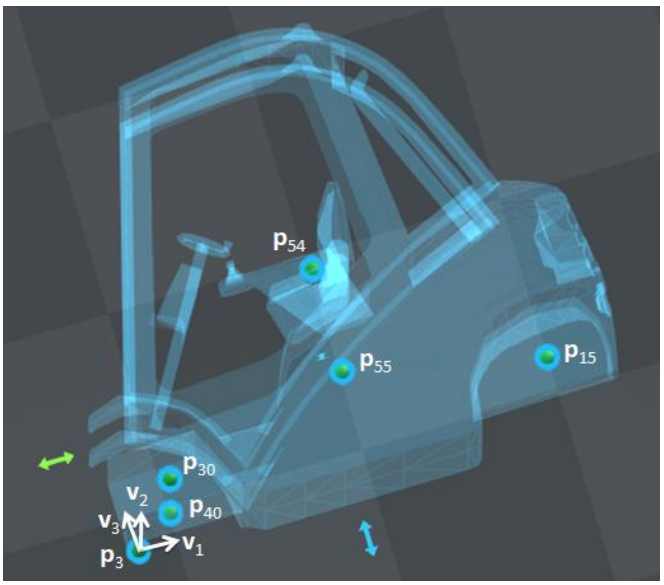

Fig 2.7 Definición del triedro del chasis y la cabina

Las restricciones introducidas son:

$$\mathbf{v}_1^{\mathrm{T}}\mathbf{v}_1 - 1 = 0 \quad (2.37)$$

$$\mathbf{v}_2^{\mathrm{T}}\mathbf{v}_2 - 1 = 0 \quad (2.38)$$

$$\mathbf{v}_3^{\mathrm{T}}\mathbf{v}_3 - 1 = 0 \quad (2.39)$$

$$\mathbf{v}_1^{\mathrm{T}}\mathbf{v}_2 - k_1 = 0 \quad (2.40)$$

$$\mathbf{v}_1^{\mathrm{T}}\mathbf{v}_3 - k_2 = 0 \quad (2.41)$$

$$\mathbf{v}_2^{\mathrm{T}}\mathbf{v}_3 - k_3 = 0 \quad (2.42)$$

$$\mathbf{r}_{3,15} - \alpha_{15}\cdot\mathbf{v}_1 - \beta_{15}\cdot\mathbf{v}_2 - \gamma_{15}\cdot\mathbf{v}_3 = 0 \quad (2.43)$$

$$\mathbf{r}_{3,30} - \alpha_{30}\cdot\mathbf{v}_1 - \beta_{30}\cdot\mathbf{v}_2 - \gamma_{30}\cdot\mathbf{v}_3 = 0 \quad (2.44)$$

$$\mathbf{r}_{3,40} - \alpha_{40}\cdot\mathbf{v}_1 - \beta_{40}\cdot\mathbf{v}_2 - \gamma_{40}\cdot\mathbf{v}_3 = 0 \quad (2.45)$$

$$\mathbf{r}_{3,54} - \alpha_{54}\cdot\mathbf{v}_1 - \beta_{54}\cdot\mathbf{v}_2 - \gamma_{54}\cdot\mathbf{v}_3 = 0 \quad (2.46)$$

$$\mathbf{r}_{3,55} - \alpha_{55}\cdot\mathbf{v}_1 - \beta_{55}\cdot\mathbf{v}_2 - \gamma_{55}\cdot\mathbf{v}_3 = 0 \quad (2.47)$$

$$\mathbf{v}_4 - \alpha_{u4}\cdot\mathbf{v}_1 - \beta_{u4}\cdot\mathbf{v}_2 - \gamma_{u4}\cdot\mathbf{v}_3 = 0 \quad (2.48)$$

El sólido tiene 30 coordenadas variables (6 puntos y 4 vectores) y 24 ecuaciones de restricción, 3 de vector unitario, 3 de producto escalar constante y 18 de combinación lineal, por lo que se introducen 30 variables y 24 incógnitas, lo que deja 6 grados de libertad.

- **Eje basculante**

La máquina va provista de un eje basculante trasero que juega un papel clave para la estabilidad de la misma. El eje basculante hace que la máquina pueda circular por terrenos irregulares sin que una de las ruedas pierda contacto con el suelo.

La desventaja del eje basculante es que la estabilidad de la máquina se convierte en la de un sólido con 3 puntos de contacto, en lugar de 4 puntos como corresponde a un vehículo con 4 ruedas.

El chasis lleva unos topes de acero que limitan el posible balanceo excesivo del eje basculante. El ángulo máximo de balanceo impuesto por los topes no lo proporciona el fabricante pero fue medido in situ estableciéndose un valor aproximado de unos 3.68º hacia cada lado, que es el valor empleado en la simulación.

El eje basculante va unido al chasis mediante un par de revolución centrado en el mismo y ubicado a una cierta altura por encima del eje que une los centros de las ruedas. Dicha altura es también importante para la estabilidad de la máquina aunque su efecto en la estabilidad es mucho más limitado que en el caso del ángulo.

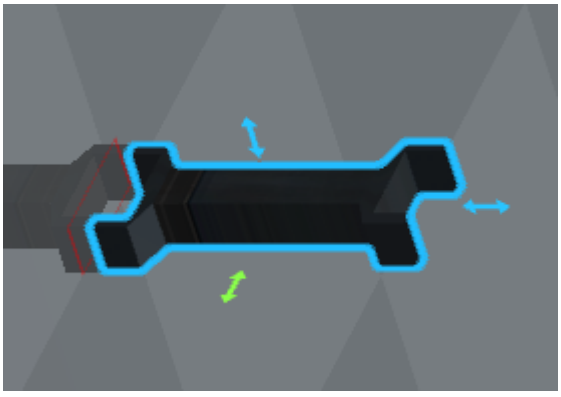

Fig 2.8 Sólido 2 de la carretilla. Eje basculante

Como variables para definir el eje basculante se han utilizado:

- Tres puntos: $\mathbf{p}_{15}$, $\mathbf{p}_{13}$ *y* $\mathbf{p}_{23}$
- Dos vectores: $\mathbf{v}_1$ *y* $\mathbf{v}_{53}$

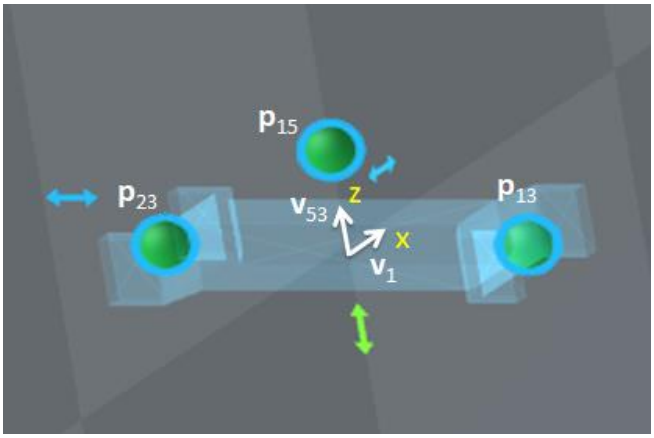

Fig 2.9 Esquema de puntos y vectores del eje basculante

Cada punto con los que se representa el modelo, tiene una posición concreta que representa en cada caso: $\mathbf{p}_{15}$ punto de articulación del eje basculante trasero, $\mathbf{p}_{13}$ punto de articulación para el giro de la rueda (trasera izquierda) y $\mathbf{p}_{23}$ punto de articulación para el giro de la rueda (trasera derecha).

Por su parte los vectores unitarios $\mathbf{v}_1$ *y* $\mathbf{v}_{53}$, representan a $\mathbf{i}$,$\mathbf{k}$ en ejes globales respectivamente.

Para modelizar el eje basculante, se han utilizado dos puntos y dos vectores, $\mathbf{p}_{13}$, $\mathbf{p}_{23}$, $\mathbf{v}_1$ *y* $\mathbf{v}_{53}$ en este mismo orden, ya que la MBSLIM construye el triedro basándose en este orden. En el Anexo I, en el fichero lectdatos.f90, se puede ver este mismo orden de introducción, quedando así el triedro definido de la siguiente forma:

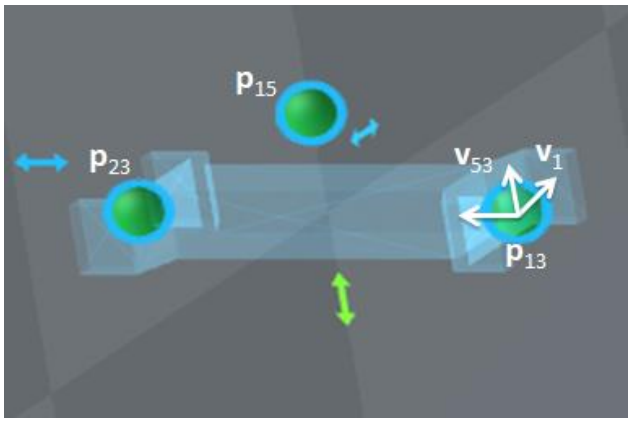

Fig 2.10 Definición del triedro del eje basculante

Tanto el punto $\mathbf{p}_{15}$, como el vector $\mathbf{v}_1$, ya han sido definidos en el chasis por lo que no se incluyen sus restricciones ni cuentan como nuevas variables. Sabiendo esto, las restricciones introducidas son:

$$\mathbf{v}_{53}^{\mathrm{T}}\mathbf{v}_{53}-1=0 \tag{2.49}$$

$$\mathbf{r}_{13,23}^{\mathrm{T}}\mathbf{r}_{13,23}-L_{13,23}^{2}=0 \tag{2.49}$$

$$\mathbf{v}_{1}^{\mathrm{T}}\mathbf{v}_{53}-k_4=0 \tag{2.51}$$

$$\mathbf{v}_1^{\mathrm{T}}\mathbf{r}_{13,23} - k_5 = 0 \quad (2.52)$$

$$\mathbf{v}_{53}^{\mathrm{T}}\mathbf{r}_{13,23} - k_6 = 0 \quad (2.53)$$

$$\mathbf{r}_{13,15} - \alpha_{15} \cdot \mathbf{r}_{13,23} - \beta_{15} \cdot \mathbf{v}_{53} - \gamma_{15} \cdot \mathbf{v}_1 = 0 \quad (2.54)$$

El sólido tiene 9 coordenadas variables (2 puntos y 1 vector) y 8 ecuaciones de restricción, 1 de vector unitario, 1 de distancia, 3 de producto escalar constante y 3 de combinación lineal, por lo que se introducen 9 variables y 8 incógnitas, lo que deja 1 grado de libertad.

- **Rueda trasera derecha**

Se han modelizado las ruedas traseras con las dimensiones reales de las de la carretilla accidentada. Éstas han sido facilitadas por la empresa propietaria de la propia carretilla.

Fig 2.11 Sólido 3.1 de la carretilla. Rueda trasera derecha

Como variables para definir la rueda trasera derecha se han utilizado:

- Dos puntos: $\mathbf{p}_{10}$ *y* $\mathbf{p}_{13}$
- Tres vectores: $\mathbf{v}_{10,}$ $\mathbf{v}_{11}$ *y* $\mathbf{v}_{12}$

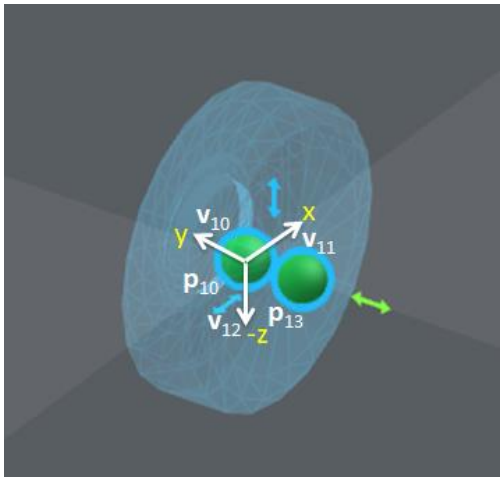

Fig 2.12 Esquema de puntos y vectores de la rueda trasera derecha

Cada punto con los que se representa el modelo, tiene una posición concreta que representa en cada caso: $\mathbf{p}_{10}$ punto centro de la rueda, y $\mathbf{p}_{13}$ punto de la articulación del pivote de dirección.

Por su parte los vectores unitarios $\mathbf{v}_{10,}\ \mathbf{v}_{11}\ y\ \mathbf{v}_{12}$, representan a $\mathbf{j},\mathbf{i},-\mathbf{k}$ en ejes globales respectivamente.

Para modelizar la rueda trasera derecha, se han utilizado un punto y tres vectores, $\mathbf{p}_{10}, \mathbf{v}_{10},\ \mathbf{v}_{11}\ y\ \mathbf{v}_{12}$ en este mismo orden, ya que la MBSLIM construye el triedro basándose en este orden. En el Anexo I, en el fichero lectdatos.f90, se puede ver este mismo orden de introducción, quedando así el triedro definido de la siguiente forma:

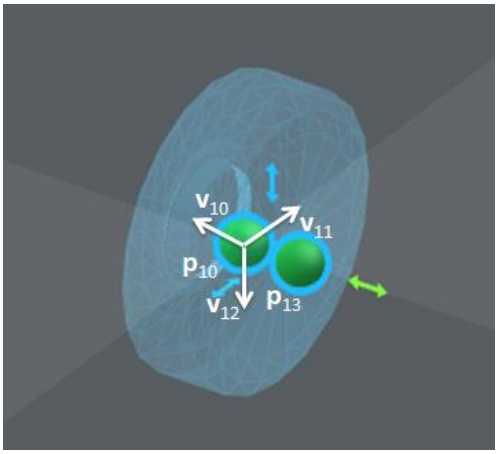

Fig 2.13 Definición del triedro de la rueda trasera derecha

El punto $\mathbf{p}_{13}$ ya ha sido definido en el eje basculante por lo que no se incluyen sus restricciones ni cuenta como nueva variable. Sabiendo esto, las restricciones introducidas son:

$$\mathbf{v}_{10}^{\mathrm{T}}\mathbf{v}_{10}-1=0 \tag{2.55}$$

$$\mathbf{v}_{11}^{\mathrm{T}}\mathbf{v}_{11}-1=0 \tag{2.56}$$

$$\mathbf{v}_{12}^{\mathrm{T}}\mathbf{v}_{12}-1=0 \tag{2.57}$$

$$\mathbf{v}_{10}^{\mathrm{T}}\mathbf{v}_{11}-k_7=0 \tag{2.58}$$

$$\mathbf{v}_{10}^{\mathrm{T}}\mathbf{v}_{12}-k_8=0 \tag{2.59}$$

$$\mathbf{v}_{11}^{\mathrm{T}}\mathbf{v}_{12}-k_9=0 \tag{2.60}$$

$$\mathbf{r}_{10,13}-\alpha_{13}\cdot\mathbf{v}_{10}-\beta_{13}\cdot\mathbf{v}_{11}-\gamma_{13}\cdot\mathbf{v}_{12}=0 \tag{2.61}$$

Además de estas restricciones, al tener un punto de articulación con el eje basculante, se añade una restricción a mayores que indica que los vectores $\mathbf{v}_{53}$, perteneciente al eje basculante y $\mathbf{v}_{10}$, perteneciente a la rueda trasera derecha, son ortogonales, o lo que es lo mismo, perpendiculares. Esto se consigue con el producto escalar.

$$\mathbf{v}_{53}^{\mathrm{T}}\mathbf{v}_{10}=0 \tag{2.62}$$

El sólido tiene 12 coordenadas variables (1 punto y 3 vectores) y 10 ecuaciones de restricción, 3 de vector unitario, 3 de producto escalar constante, 3 de combinación lineal y 1 de vectores ortogonales, por lo que se introducen 12 variables y 10 incógnitas, lo que deja 2 grados de libertad.

- **Rueda trasera izquierda**

Fig 2.14 Sólido 3.2 de la carretilla. Rueda trasera izquierda

Como variables para definir la rueda trasera izquierda se han utilizado:

- Dos puntos: $\mathbf{p}_{20}$ *y* $\mathbf{p}_{23}$
- Cuatro vectores: $\mathbf{v}_{20,}$ $\mathbf{v}_{21,}$ $\mathbf{v}_{22}$ *y* $\mathbf{v}_{23}$

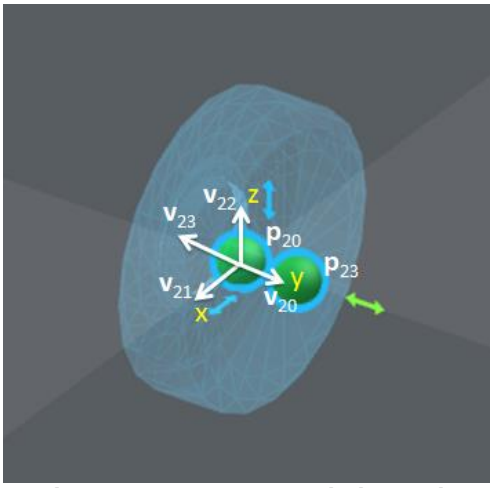

Fig 2.15 Esquema de puntos y vectores de la rueda trasera izquierda

Cada punto con los que se representa el modelo, tiene una posición concreta que representa en cada caso: $\mathbf{p}_{20}$ punto centro de la rueda, y $\mathbf{p}_{23}$ punto de la articulación del pivote de dirección.

Por su parte los vectores unitarios $\mathbf{v}_{20,}$ $\mathbf{v}_{21,}$ $\mathbf{v}_{22}$ *y* $\mathbf{v}_{23}$, representan a $\mathbf{j},\mathbf{i},\mathbf{k},-\mathbf{j}$ en ejes globales respectivamente.

Para modelizar la rueda trasera izquierda, se han utilizado un punto y tres vectores, $\mathbf{p}_{20}, \mathbf{v}_{20}, \mathbf{v}_{21}$ *y* $\mathbf{v}_{22}$ en este mismo orden, ya que la MBSLIM construye el triedro basándose en este orden. En el Anexo I, en el fichero lectdatos.f90, se puede ver este mismo orden de introducción, quedando así, el triedro definido de la siguiente forma:

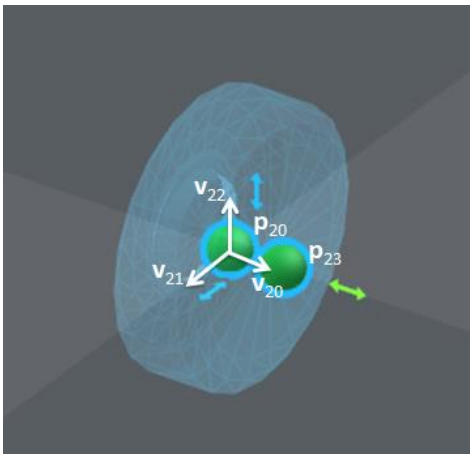

Fig 2.16 Definición del triedro de la rueda trasera izquierda

El punto $\mathbf{p}_{23}$ ya ha sido definido en el eje basculante por lo que no se incluyen sus restricciones ni cuenta como nueva variable. Sabiendo esto, las restricciones introducidas son:

$$\mathbf{v}_{20}^{\mathrm{T}}\mathbf{v}_{20}-1=0 \quad (2.63)$$

$$\mathbf{v}_{21}^{\mathrm{T}}\mathbf{v}_{21}-1=0 \quad (2.64)$$

$$\mathbf{v}_{22}^{\mathrm{T}}\mathbf{v}_{22}-1=0 \quad (2.65)$$

$$\mathbf{v}_{20}^{\mathrm{T}}\mathbf{v}_{21}-k_{10}=0 \quad (2.66)$$

$$\mathbf{v}_{20}^{\mathrm{T}}\mathbf{v}_{22}-k_{11}=0 \quad (2.67)$$

$$\mathbf{v}_{21}^{\mathrm{T}}\mathbf{v}_{22}-k_{12}=0 \quad (2.68)$$

$$\mathbf{r}_{20,23}-\alpha_{23}\cdot\mathbf{v}_{20}-\beta_{23}\cdot\mathbf{v}_{21}-\gamma_{23}\cdot\mathbf{v}_{22}=0 \quad (2.69)$$

$$\mathbf{v}_{23}-\alpha_{u23}\cdot\mathbf{v}_{20}-\beta_{u23}\cdot\mathbf{v}_{21}-\gamma_{u23}\cdot\mathbf{v}_{22}=0 \quad (2.70)$$

Además de estas restricciones, al tener un punto de articulación con el eje basculante, se ha añadido una restricción a mayores que indica que los vectores $\mathbf{v}_{53}$, perteneciente al eje basculante y $\mathbf{v}_{20}$, perteneciente a la rueda trasera izquierda, son ortogonales, o lo que es lo mismo, perpendiculares. Esto se consigue con el producto escalar.

$$\mathbf{v}_{53}^{\mathrm{T}}\mathbf{v}_{20}=0 \quad (2.71)$$

El sólido tiene 15 coordenadas variables (1 punto y 4 vectores) y 13 ecuaciones de restricción, 3 de vector unitario, 3 de producto escalar constante, 6 de combinación lineal y 1 de vectores ortogonales, por lo que se introducen 15 variables y 13 incógnitas, lo que deja 2 grados de libertad.

- **Rueda delantera derecha**

De la misma forma que las ruedas traseras, se han modelizado las ruedas delanteras con las dimensiones reales de las de la carretilla accidentada.

Fig 2.17 Sólido 4.1 de la carretilla. Rueda delantera derecha

Como variables para definir la rueda delantera derecha se han utilizado:

- Un punto: $\mathbf{p}_{30}$
- Tres vectores: $\mathbf{v}_{2,}\ \mathbf{v}_{31}\ y\ \mathbf{v}_{32}$

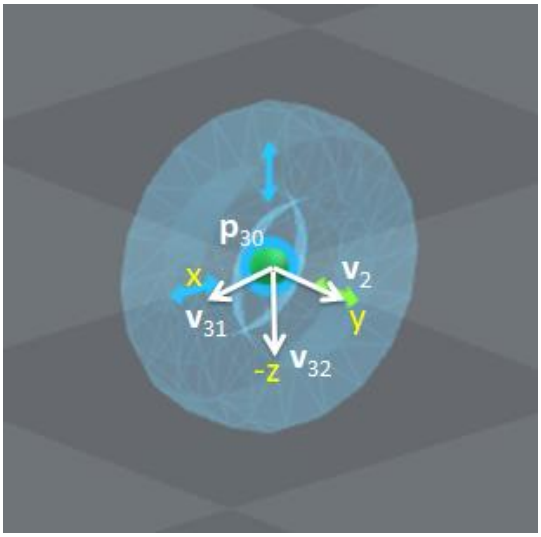

Fig 2.18 Esquema de puntos y vectores de la rueda delantera derecha

El punto $\mathbf{p}_{30}$ con el que se representa el modelo, tiene una posición concreta que representa el punto central de la rueda.

Por su parte los vectores unitarios $\mathbf{v}_{2,}\ \mathbf{v}_{31}\ y\ \mathbf{v}_{32}$, representan a $\mathbf{j},\mathbf{i},-\mathbf{k}$ en ejes globales respectivamente.

Para modelizar la rueda delantera derecha, se han utilizado un punto y tres vectores, $\mathbf{p}_{30}, \mathbf{v}_{2}, \mathbf{v}_{31}\ y\ \mathbf{v}_{32}$ en este mismo orden, ya que la MBSLIM construye el triedro basándose en este orden. En el Anexo I, en el fichero lectdatos.f90, se puede ver este mismo orden de introducción, quedando así, el triedro definido de la siguiente forma:

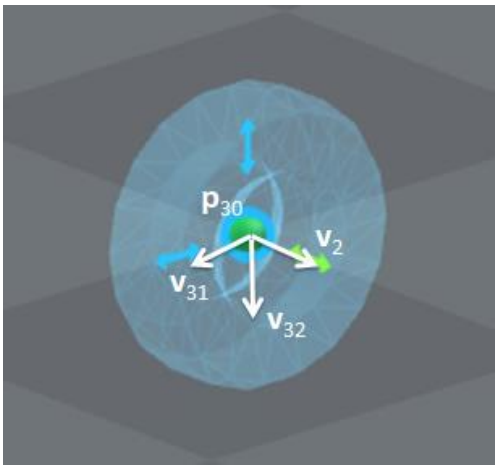

Fig 2.19 Definición del triedro de la rueda delantera derecha

El punto $\mathbf{p}_{30}$, y el vector $\mathbf{v}_2$, ya han sido definidos en el chasis por lo que no se incluyen sus restricciones ni cuentan como nuevas variables. Sabiendo esto, las restricciones introducidas son:

$$\mathbf{v}_{31}^{\mathrm{T}}\mathbf{v}_{31}-1=0 \quad (2.72)$$

$$\mathbf{v}_{32}^{\mathrm{T}}\mathbf{v}_{32}-1=0 \quad (2.73)$$

$$\mathbf{v}_{2}^{\mathrm{T}}\mathbf{v}_{31}-k_{13}=0 \quad (2.74)$$

$$\mathbf{v}_{2}^{\mathrm{T}}\mathbf{v}_{32}-k_{14}=0 \quad (2.75)$$

$$\mathbf{v}_{31}^{\mathrm{T}}\mathbf{v}_{32}-k_{15}=0 \quad (2.76)$$

El sólido tiene 6 coordenadas variables (2 vectores) y 5 ecuaciones de restricción, 2 de vector unitario y 3 de producto escalar constante, por lo que se introducen 6 variables y 5 incógnitas, lo que deja 1 grado de libertad.

- **<u>Rueda delantera izquierda</u>**

Fig 2.20 Sólido 4.2 de la carretilla. Rueda delantera izquierda

Como variables para definir la rueda delantera izquierda se han utilizado:

- Un puntos: $\mathbf{p}_{40}$
- Tres vectores: $\mathbf{v}_{2,}\ \mathbf{v}_{41,}\ \mathbf{v}_{42}\ y\ \mathbf{v}_{4}$

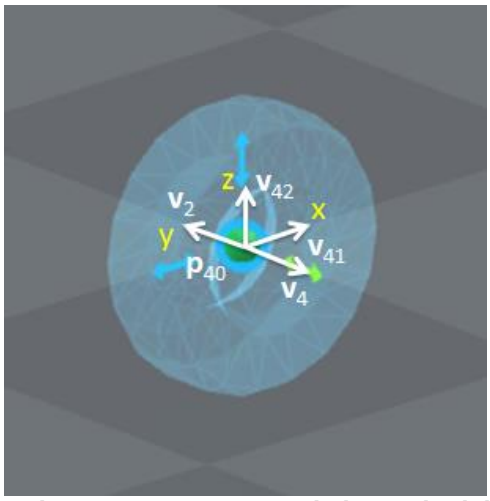

Fig 2.21 Esquema de puntos y vectores de la rueda delantera izquierda

El punto $\mathbf{p}_{40}$ con el que se representa el modelo, tiene una posición concreta que representa el punto central de la rueda.

Por su parte los vectores unitarios $\mathbf{v}_{2,}\ \mathbf{v}_{41,}\ \mathbf{v}_{42}\ y\ \mathbf{v}_4$, representan a $\mathbf{j},\mathbf{i},\mathbf{k},-\mathbf{j}$ en ejes globales respectivamente.

Para modelizar la rueda delantera izquierda, se han utilizado un punto y tres vectores, $\mathbf{p}_{40}, \mathbf{v}_2, \mathbf{v}_{41}\ y\ \mathbf{v}_{42}$ en este mismo orden, ya que la MBSLIM construye el triedro basándose en este orden. En el Anexo I, en el fichero lectdatos.f90, se puede ver este mismo orden de introducción, quedando así el triedro definido de la siguiente forma:

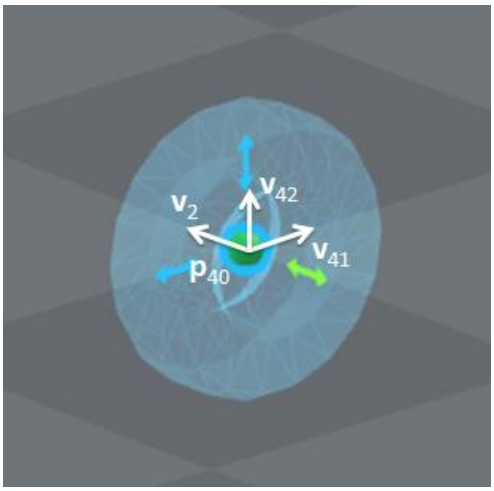

Fig 2.22 Definición del triedro de la rueda delantera izquierda

El punto $\mathbf{p}_{40}$, y los vectores $\mathbf{v}_2\ y\ \mathbf{v}_4$, ya han sido definidos en el chasis por lo que, excepto en el caso de $\mathbf{v}_4$ que al tratarse como un vector adicional al modelizado se añaden sus restricciones de combinación lineal, no se incluyen sus restricciones ni cuentan como nuevas variables. Sabiendo esto, las restricciones introducidas son:

$$\mathbf{v}_{41}^{\mathrm{T}}\mathbf{v}_{41}-1=0 \tag{2.77}$$

$$\mathbf{v}_{42}^{\mathrm{T}}\mathbf{v}_{42}-1=0 \tag{2.78}$$

$$\mathbf{v}_{2}^{\mathrm{T}}\mathbf{v}_{41}-k_{16}=0 \tag{2.79}$$

$$\mathbf{v}_{2}^{\mathrm{T}}\mathbf{v}_{42}-k_{17}=0 \tag{2.80}$$

$$\mathbf{v}_{41}^{\mathrm{T}}\mathbf{v}_{42}-k_{18}=0 \tag{2.81}$$

$$\mathbf{v}_4 - \alpha_{u4} \cdot \mathbf{v}_2 - \beta_{u4} \cdot \mathbf{v}_{41} - \gamma_{u4} \cdot \mathbf{v}_{42} = 0 \quad (2.82)$$

El sólido tiene 9 coordenadas variables (3 vectores) y 8 ecuaciones de restricción, 2 de vector unitario, 3 de producto escalar constante y 3 de combinación lineal, por lo que se introducen 9 variables y 8 incógnitas, lo que deja 1 grado de libertad.

- **Mástil 1**

El mástil 1 se ha modelado con las dimensiones proporcionadas por el fabricante en el catálogo de la máquina (ver Anexo II).

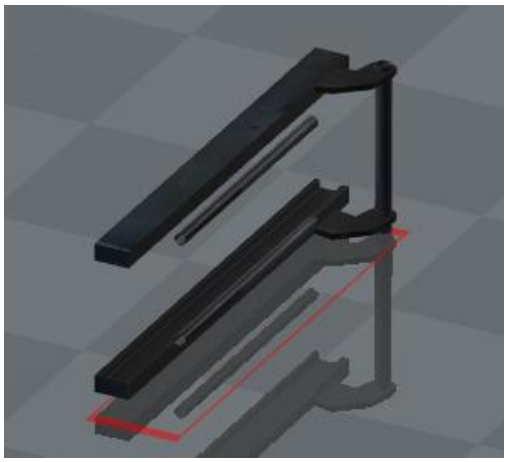

Fig 2.23 Sólido 5 de la carretilla. Mástil 1

Como variables para definir el mástil se han utilizado:

- Dos puntos: $\mathbf{p}_1$ *y* $\mathbf{p}_3$
- Tres vectores: $\mathbf{v}_5$, $\mathbf{v}_2$ *y* $\mathbf{v}_6$

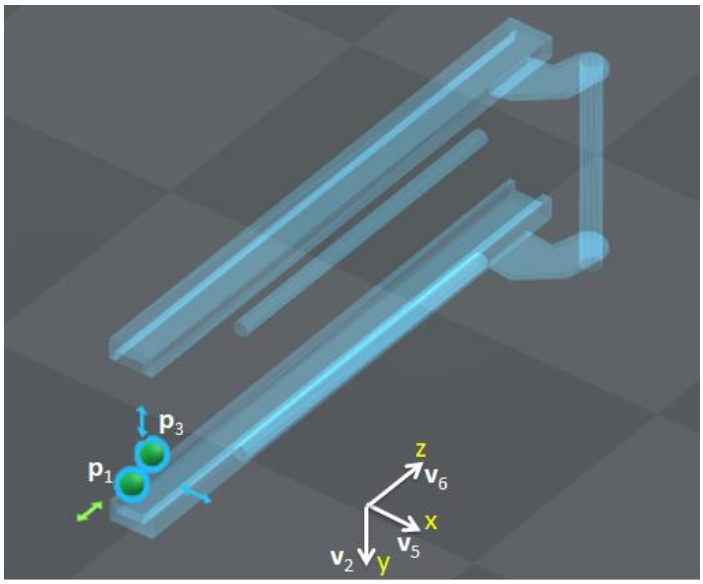

Fig 2.24 Esquema de puntos y vectores del mástil 1

Cada punto con los que se representa el modelo, tiene una posición concreta que representa en cada caso: $\mathbf{p}_1$ punto del par prismático con el mástil 2 y $\mathbf{p}_3$ punto del par de revolución con el chasis.

Por su parte los vectores unitarios, $\mathbf{v}_5$, $\mathbf{v}_2$ *y* $\mathbf{v}_6$ representan a $\mathbf{i},\mathbf{j},\mathbf{k}$ en ejes globales respectivamente.

Para modelizar el mástil 1, se han utilizado un punto y tres vectores, $\mathbf{p}_3$, $\mathbf{v}_5$, $\mathbf{v}_2$ *y* $\mathbf{v}_6$ en este mismo orden, ya que la MBSLIM construye el triedro basándose en este orden. En el Anexo I, en el fichero lectdatos.f90, se puede ver este mismo orden de introducción, quedando así el triedro definido de la siguiente forma:

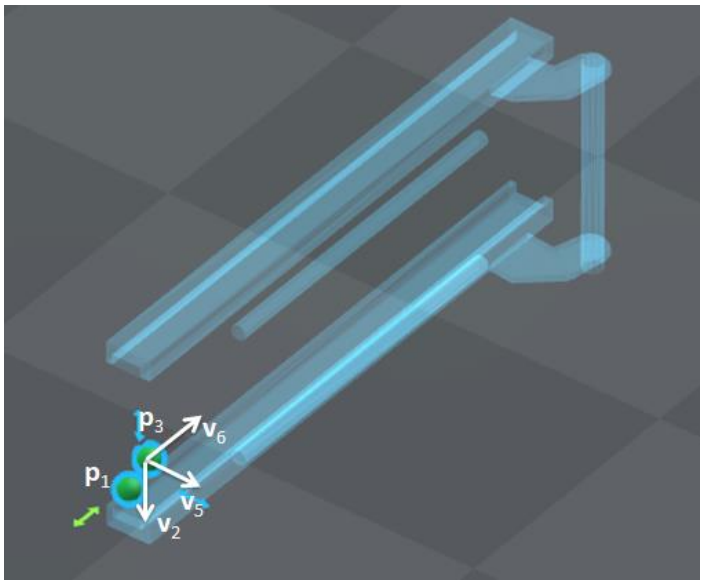

Fig 2.25 Definición del triedro del mástil 1

Tanto el punto $\mathbf{p}_3$, como el vector $\mathbf{v}_2$, ya han sido definidos en el chasis por lo que no se incluyen sus restricciones ni cuentan como nuevas variables. Sabiendo esto, las restricciones introducidas son:

$$\mathbf{v}_5^{\mathrm{T}}\mathbf{v}_5 - 1 = 0 \tag{2.83}$$

$$\mathbf{v}_6^{\mathrm{T}}\mathbf{v}_6 - 1 = 0 \tag{2.84}$$

$$\mathbf{v}_5^{\mathrm{T}}\mathbf{v}_2 - k_{18} = 0 \tag{2.85}$$

$$\mathbf{v}_5^{\mathrm{T}}\mathbf{v}_6 - k_{19} = 0 \tag{2.86}$$

$$\mathbf{v}_2^{\mathrm{T}}\mathbf{v}_6 - k_{20} = 0 \tag{2.87}$$

$$\mathbf{r}_{3,1} - \alpha_1 \cdot \mathbf{v}_5 - \beta_1 \cdot \mathbf{v}_2 - \gamma_1 \cdot \mathbf{v}_6 = 0 \tag{2.88}$$

El sólido tiene 9 coordenadas variables (1 punto y 2 vectores) y 8 ecuaciones de restricción, 2 de vector unitario, 3 de producto escalar constante y 3 de combinación lineal, por lo que se introducen 9 variables y 8 incógnitas, lo que deja 1 grado de libertad.

- **Mástil 2**

De la misma forma que el mástil 1, el mástil 2 ha sido modelado con las dimensiones proporcionadas por el fabricante en el catálogo de la máquina (ver Anexo II).

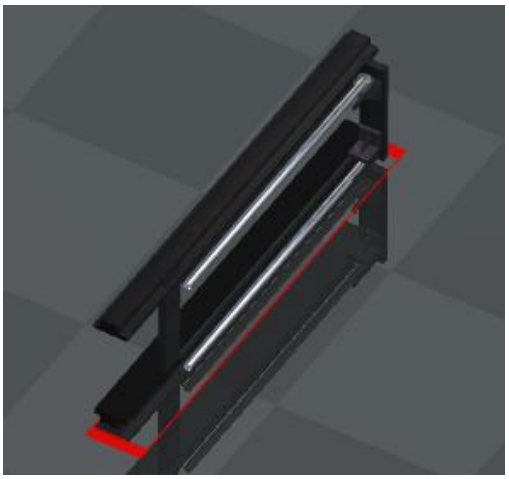

Fig 2.26 Sólido 6 de la carretilla. Mástil 2

Como variables para definir el mástil 2 se han utilizado:

- Un punto: $\mathbf{p}_6$
- Tres vectores: $\mathbf{v}_5$, $\mathbf{v}_2$ *y* $\mathbf{v}_6$

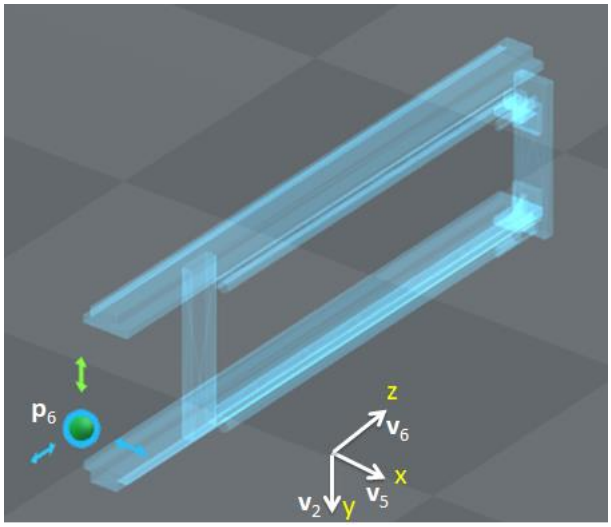

Fig 2.27 Esquema de puntos y vectores del mástil 2

El punto $\mathbf{p}_6$ con el que se representa el modelo, tiene una posición concreta que representa el punto del par prismático con el mástil 1.

Por su parte los vectores unitarios, $\mathbf{v}_5$, $\mathbf{v}_2$ *y* $\mathbf{v}_6$ representan a $\mathbf{i},\mathbf{j},\mathbf{k}$ en ejes globales respectivamente.

Para modelizar el mástil 2, se han utilizado un punto y tres vectores, $\mathbf{p}_6, \mathbf{v}_5$, $\mathbf{v}_2$ *y* $\mathbf{v}_6$ en este mismo orden, ya que la MBSLIM construye el triedro basándose en este orden. En el Anexo I, en el fichero lectdatos.f90, se puede ver este mismo orden de introducción, quedando así, el triedro definido de la siguiente forma:

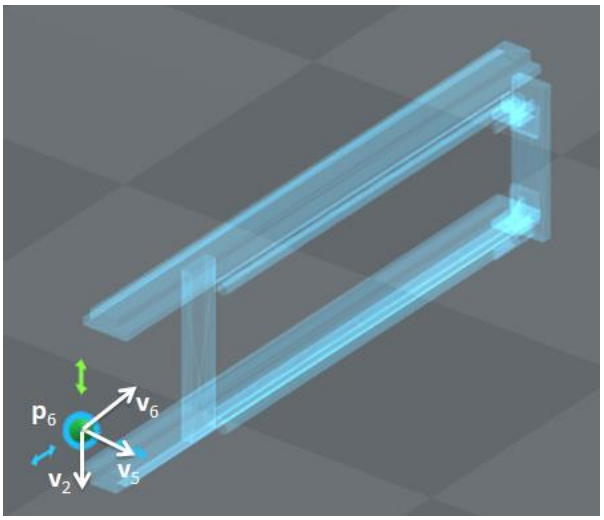

Fig 2.28 Definición del triedro del mástil 2

Los vectores $\mathbf{v}_5$, $\mathbf{v}_2$ *y* $\mathbf{v}_6$ ya han sido definidos en el mástil 1, por lo que no se incluyen sus restricciones ni cuentan como nuevas variables. Sabiendo esto, de la modelización sólo queda el punto $\mathbf{p}_6$.

Pero al tener un par prismático se introduce una restricción que indica que los puntos $\mathbf{p}_1$, del mástil 1 y $\mathbf{p}_6$, del mástil 2, están siempre alineados con $\mathbf{v}_6$, es decir, que los vectores $\mathbf{v}_6$ *y* $\mathbf{r}_{6,1}$ son paralelos entre sí. Esto se consigue con el producto vectorial.

$$\mathbf{v}_6 \wedge \mathbf{r}_{6,1} = 0 \tag{2.89}$$

El sólido tiene 3 coordenadas variables (1 punto) y 2 ecuaciones de restricción, ya que, aunque el producto vectorial expresa tres ecuaciones, una de ellas es linealmente dependiente de las otras dos, por lo que se introducen 3 variables y 0 incógnitas, lo que deja 1 grado de libertad.

- **Uñas y útil**

Las uñas y el útil, se han modelado con las dimensiones proporcionadas por el fabricante en el catálogo de la máquina (ver Anexo II).

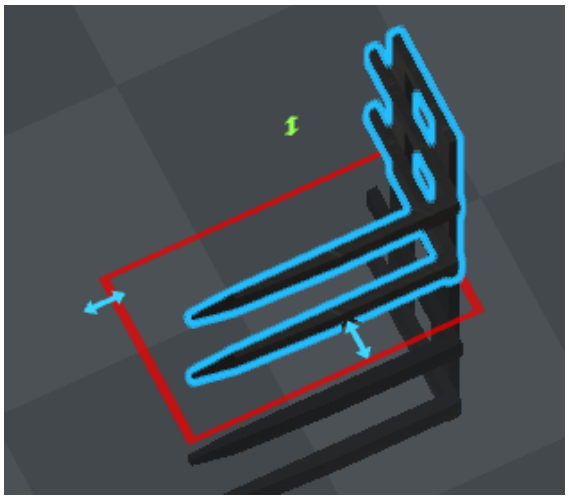

Fig 2.29 Sólido 7 de la carretilla. Uñas y útil

Como variables para definir las uñas y útil se han utilizado:

- Un punto: $\mathbf{p}_2$
- Tres vectores: $\mathbf{v}_5$, $\mathbf{v}_2$ *y* $\mathbf{v}_6$

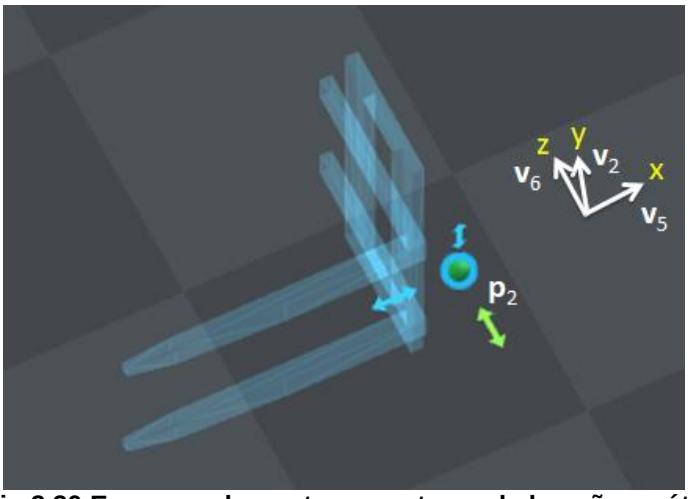

Fig 2.30 Esquema de puntos y vectores de las uñas y útil

El punto $\mathbf{p}_2$ con el que se representa el modelo, tiene una posición concreta que representa el punto del par prismático con el mástil 2.

Por su parte los vectores unitarios, $\mathbf{v}_5$, $\mathbf{v}_2$ *y* $\mathbf{v}_6$ representan a $\mathbf{i},\mathbf{j},\mathbf{k}$ en ejes globales respectivamente.

Para modelizar las uñas y útil, se han utilizado un punto y tres vectores, $\mathbf{p}_2$, $\mathbf{v}_5$, $\mathbf{v}_2$ *y* $\mathbf{v}_6$ en este mismo orden, ya que la MBSLIM construye el triedro basándose en este orden. En el Anexo I, en el fichero lectdatos.f90, se puede ver este mismo orden de introducción, quedando así, el triedro definido de la siguiente forma:

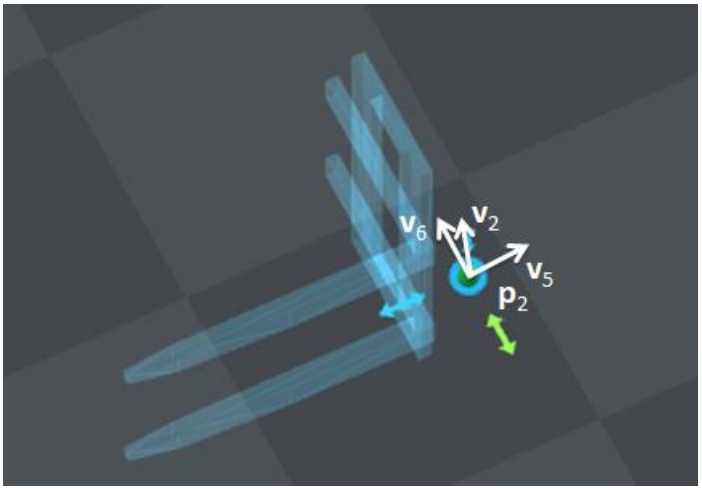

Fig 2.31 Definición del triedro de las uñas y útil

Los vectores $\mathbf{v}_5$, $\mathbf{v}_2$ *y* $\mathbf{v}_6$ ya han sido definidos en los mástiles. Sabiendo esto, sólo queda el punto $\mathbf{p}_2$.

Pero al tener un par prismático se introduce una restricción que indica que los puntos $\mathbf{p}_1$, del mástil 1 y $\mathbf{p}_2$, de las uñas y útil, están siempre alineados con $\mathbf{v}_6$, es decir que los vectores $\mathbf{v}_6$ *y* $\mathbf{r}_{2,1}$ son paralelos entre sí. Esto se consigue con el producto vectorial.

$$\mathbf{v}_6 \wedge \mathbf{r}_{2,1} = 0 \tag{2.90}$$

El sólido tiene 3 coordenadas variables (1 punto) y 2 ecuaciones de restricción, ya que, aunque el producto vectorial expresa tres ecuaciones, una de ellas es linealmente dependiente de las otras dos, por lo que se introducen 3 variables y 0 incógnitas, lo que deja 1 grado de libertad.

- **Conductor**

Dado que la masa total de la carretilla es más de cuatro toneladas y media, la masa del conductor es irrelevante en el modelo. No obstante, se ha tenido en cuenta un conductor medio de 85 kg.

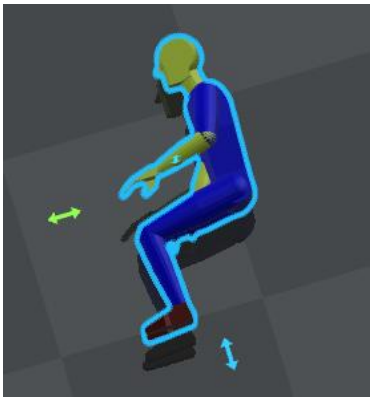

Fig 2.32 Sólido 8 de la carretilla. Conductor

Como variables para definir el conductor se han utilizado:

- Un punto: $\mathbf{p}_{54}$
- Tres vectores: $\mathbf{v}_1$, $\mathbf{v}_2$ *y* $\mathbf{v}_3$

Fig 2.33 Esquema de puntos y vectores del conductor

El punto $\mathbf{p}_{54}$ con el que se representa el modelo, tiene una posición concreta que representa la ubicación del conductor.

Por su parte los vectores unitarios, $\mathbf{v}_1$, $\mathbf{v}_2$ *y* $\mathbf{v}_3$ representan a $\mathbf{i}$,$\mathbf{j}$,$\mathbf{k}$ en ejes globales respectivamente.

Para modelizar el conductor, se han utilizado un punto y tres vectores, $\mathbf{p}_{54}$, $\mathbf{v}_1$, $\mathbf{v}_2$ *y* $\mathbf{v}_3$ en este mismo orden, ya que la MBSLIM construye el triedro basándose en este orden. En

el Anexo I, en el fichero lectdatos.f90, se puede ver este mismo orden de introducción, quedando así, el triedro definido de la siguiente forma:

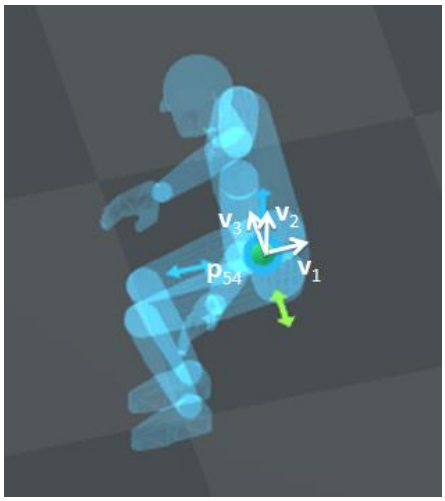

Fig 2.34 Definición del triedro del conductor

Tanto el punto $\mathbf{p}_{54}$, como los vectores $\mathbf{v}_1$, $\mathbf{v}_2$ *y* $\mathbf{v}_3$, ya han sido definidos en el chasis. Sabiendo esto, no queda nada por definir por lo que no se tienen grados de libertad.

- **Combustible**

De la misma forma que ocurrió con el conductor, la masa del combustible es irrelevante teniendo en cuenta la masa total de la carretilla. No obstante, se ha modelado el combustible como un sólido rígido aunque en realidad es un líquido que se mueve dentro del depósito. Estos efectos de movimiento del combustible dentro del depósito se desprecian en el modelo.

Como variables para definir el combustible se han utilizado:

- Un punto: $\mathbf{p}_{55}$
- Tres vectores: $\mathbf{v}_1$, $\mathbf{v}_2$ *y* $\mathbf{v}_3$

El punto $\mathbf{p}_{55}$ con el que se representa el modelo, tiene una posición concreta que representa la ubicación del combustible.

Por su parte los vectores unitarios, $\mathbf{v}_1$, $\mathbf{v}_2$ *y* $\mathbf{v}_3$ representan a $\mathbf{i}$,$\mathbf{j}$,$\mathbf{k}$ en ejes globales respectivamente.

Para modelizar el combustible, se han utilizado un punto y tres vectores, $\mathbf{p}_{55}$, $\mathbf{v}_1$, $\mathbf{v}_2$ *y* $\mathbf{v}_3$ en este mismo orden, ya que la MBSLIM construye el triedro basándose en este orden. En el anexo I, en el fichero lectdatos.f90, se puede ver este mismo orden de introducción.

Tanto el punto $\mathbf{p}_{55}$, como los vectores $\mathbf{v}_1$, $\mathbf{v}_2$ *y* $\mathbf{v}_3$, ya han sido definidos en el chasis. Sabiendo esto no queda nada por definir por lo que no se tienen grados de libertad.

2.4.3 Grados de libertad

A continuación se muestra un balance de ecuaciones/variables para ver cuantos grados de libertad existen en el modelo.

Sólido	nº variables	nº incógnitas	GDL
Chasis y cabina	30	24	6
Eje basculante	9	8	1
Ruedas trasera der.	12	10	2
Ruedas trasera izq.	15	13	2
Ruedas delantera der.	6	5	1
Ruedas delantera izq.	9	8	1
Mástil 1	9	8	1
Mástil 2	3	2	1
Uñas y útil	3	2	1
Conductor	0	0	0
Combustible	0	0	0
TOTAL	96	80	**16**

Tabla 2.2 Balance de grados de libertad

Se definen los grados de libertad en el orden que han sido citados. A continuación se muestra que existen grados de libertad que van guiados, estos llevan asociada una nueva variable, que como se ha visto anteriormente, serán las coordenadas relativas.

- **Chasis y cabina**

En el chasis y cabina existen 6 GDL, que se refieren al movimiento en un espacio tridimensional, es decir, la capacidad de moverse hacia delante/atrás, arriba/abajo, izquierda/derecha (traslación en tres ejes perpendiculares), combinados con la rotación sobre tres ejes perpendiculares (Guiñada, Cabeceo, Alabeo). El movimiento a lo largo de cada uno de los ejes es independiente de los otros, y cada uno es independiente de la rotación sobre cualquiera de los ejes.

- **Eje basculante**

En el eje basculante se define el ángulo φ_8 entre los vectores $\mathbf{v}_{53}$ y $\mathbf{v}_3$ en el plano de $\mathbf{v}_1$, que representa al ángulo de balanceo del eje, positivo en sentido anti horario mirando hacia atrás de la carretilla.

Se define mediante dos ecuaciones linealmente dependientes, una para el seno del ángulo y otra para el coseno. Dependiendo de la posición del ángulo, funcionará una mejor que la otra.

$$\begin{cases} \mathbf{v}_{53}^{\mathrm{T}}\mathbf{v}_3 - \cos\varphi_8 = 0 \\ \mathbf{v}_1^{\mathrm{T}}(\mathbf{v}_{53} \wedge \mathbf{v}_3) - \sin\varphi_8 = 0 \end{cases} \tag{2.91}$$

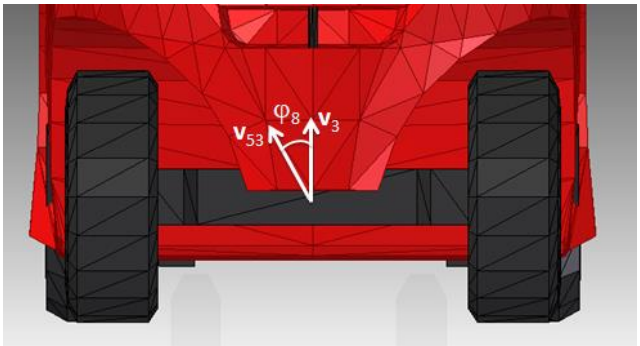

Fig 2.35 Grado de libertad φ_8

- **Rueda trasera derecha**

En la rueda trasera derecha se define el ángulo φ_1 entre los vectores $\mathbf{v}_{53}$ y $\mathbf{v}_{11}$ en el plano de $\mathbf{v}_{10}$, que representa al ángulo de giro de la rueda, siendo positivo en sentido de giro de marchas atrás. También se define el ángulo φ_6 entre los vectores $\mathbf{v}_1$ y $\mathbf{v}_{10}$ en el plano de $\mathbf{v}_{53}$, que representa el ángulo de dirección de la rueda, siendo positivo el giro de la dirección hacia la derecha.

Se definen estos ángulos mediante dos ecuaciones linealmente dependientes, una para el seno del ángulo y otra para el coseno. Dependiendo de la posición del ángulo, funcionará una mejor que la otra.

$$\begin{cases} \mathbf{v}_{53}^{\mathrm{T}}\mathbf{v}_{11} - \cos\varphi_1 = 0 \\ \mathbf{v}_{10}^{\mathrm{T}}(\mathbf{v}_{53} \wedge \mathbf{v}_{11}) - \sin\varphi_1 = 0 \end{cases} \tag{2.92}$$

$$\begin{cases} \mathbf{v}_{1}^{\mathrm{T}}\mathbf{v}_{10} - \cos\varphi_6 = 0 \\ \mathbf{v}_{53}^{\mathrm{T}}(\mathbf{v}_{1} \wedge \mathbf{v}_{10}) - \sin\varphi_6 = 0 \end{cases} \tag{2.93}$$

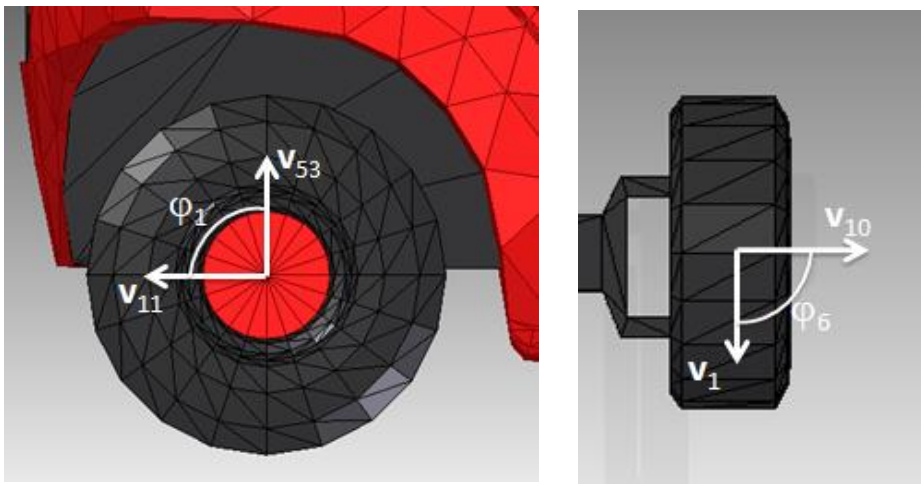

Fig 2.36 Grados de libertad φ_1 y φ_6

Además, para el ángulo φ_6, referente al ángulo de giro de dirección de la rueda, se introduce una ecuación de restricción de guiado cinemático porque como se verá en el apartado 5, será controlado por el usuario mediante un mando.

$$\varphi_6 - \varphi_6(t) = 0 \tag{2.94}$$

- **Rueda trasera izquierda**

En la rueda trasera izquierda se define el ángulo φ_2 entre los vectores $\mathbf{v}_{53}$ y $\mathbf{v}_{21}$ en el plano de $\mathbf{v}_{20}$, que representa al ángulo de giro de la rueda, siendo positivo en sentido de giro de marchas atrás. También se define el ángulo φ_7 entre los vectores $\mathbf{v}_1$ y $\mathbf{v}_{20}$ en el plano de $\mathbf{v}_{53}$, que representa el ángulo de dirección de la rueda, siendo positivo el giro de la dirección hacia la derecha.

Se definen estos ángulos mediante dos ecuaciones linealmente dependientes, una para el seno del ángulo y otra para el coseno. Dependiendo de la posición del ángulo, funcionará una mejor que la otra.

$$\begin{cases} \mathbf{v}_{53}^{\mathrm{T}}\mathbf{v}_{21} - \cos\varphi_2 = 0 \\ \mathbf{v}_{20}^{\mathrm{T}}(\mathbf{v}_{53} \wedge \mathbf{v}_{21}) - \sin\varphi_2 = 0 \end{cases} \tag{2.95}$$

$$\begin{cases} \mathbf{v}_{1}^{\mathrm{T}}\mathbf{v}_{20} - \cos\varphi_7 = 0 \\ \mathbf{v}_{53}^{\mathrm{T}}(\mathbf{v}_{1} \wedge \mathbf{v}_{20}) - \sin\varphi_7 = 0 \end{cases} \tag{2.96}$$

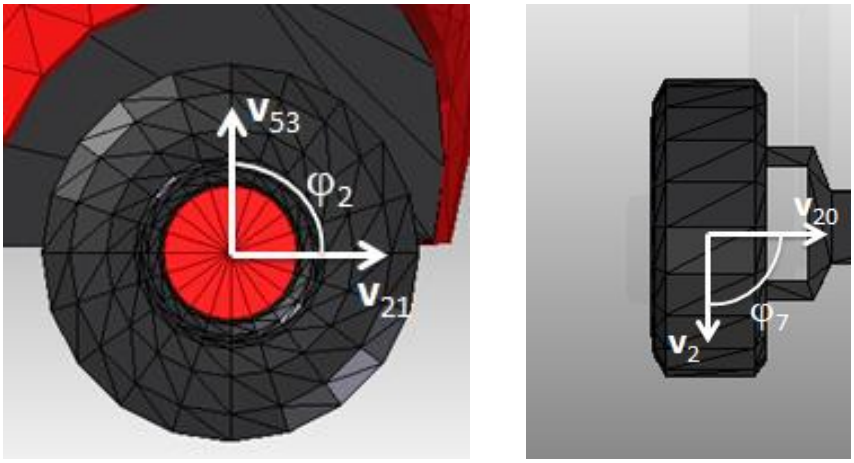

Fig 2.37 Grados de libertad φ_2 y φ_7

Además, para el ángulo φ_7, referente al ángulo de giro de dirección de la rueda, se introduce una ecuación de restricción de guiado cinemático porque como se verá en el apartado 5, será controlado por el usuario mediante un mando.

$$\varphi_7 - \varphi_7(t) = 0 \tag{2.97}$$

- **<u>Rueda delantera derecha</u>**

En la rueda delantera derecha se define el ángulo φ_3 entre los vectores $\mathbf{v}_3$ *y* $\mathbf{v}_{31}$ en el plano de $\mathbf{v}_2$, que representa al ángulo de giro de la rueda, siendo positivo en sentido de giro de marchas atrás.

Se define mediante dos ecuaciones linealmente dependientes, una para el seno del ángulo y otra para el coseno. Dependiendo de la posición del ángulo, funcionará una mejor que la otra.

$$\begin{cases} \mathbf{v}_{3}^{\mathrm{T}}\mathbf{v}_{31} - \cos\varphi_3 = 0 \\ \mathbf{v}_{2}^{\mathrm{T}}(\mathbf{v}_{3} \wedge \mathbf{v}_{31}) - \sin\varphi_3 = 0 \end{cases} \tag{2.98}$$

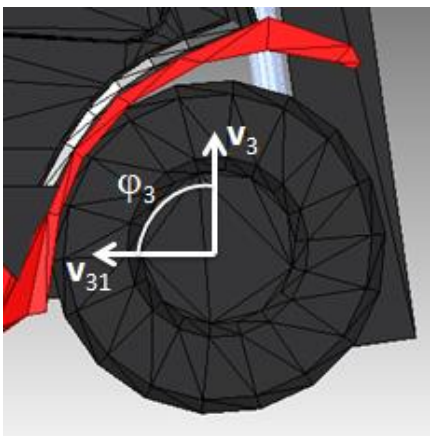

Fig 2.38 Grado de libertad φ_3

- **Rueda delantera izquierda**

En la rueda delantera izquierda se define el ángulo φ_4 entre los vectores $\mathbf{v}_3$ y $\mathbf{v}_{41}$ en el plano de $\mathbf{v}_2$, que representa al ángulo de giro de la rueda, siendo positivo en sentido de giro de marchas atrás.

Se define mediante dos ecuaciones linealmente dependientes, una para el seno del ángulo y otra para el coseno. Dependiendo de la posición del ángulo, funcionará una mejor que la otra.

$$\begin{cases} \mathbf{v}_3^{\mathrm{T}}\mathbf{v}_{41} - \cos\varphi_4 = 0 \\ \mathbf{v}_2^{\mathrm{T}}(\mathbf{v}_3 \wedge \mathbf{v}_{41}) - \sin\varphi_4 = 0 \end{cases} \tag{2.99}$$

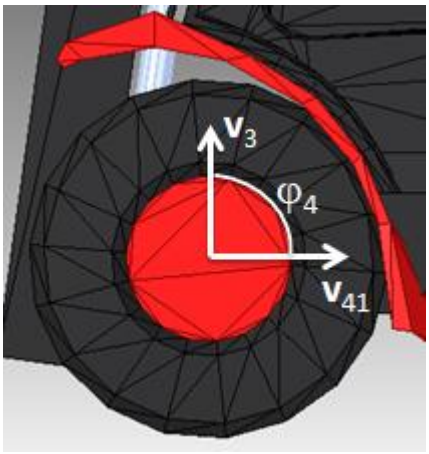

Fig 2.39 Grado de libertad φ_4

Los ángulos de giro de ambas ruedas delanteras φ_3 y φ_4, se relacionan al definirse un ángulo φ_9, que representa el ángulo diferencial de giro. Para ello, se ha introducido manualmente la siguiente restricción.

$$\varphi_9 - \frac{1}{2}(\varphi_3 + \varphi_4) = 0 \tag{2.100}$$

Para este ángulo se introduce una ecuación de restricción de guiado cinemático porque como se verá en el apartado 5, será controlado por el usuario mediante un mando.

$$\varphi_9 - \varphi_9(t) = 0 \tag{2.101}$$

- **Mástil 1**

En el mástil 1 se define el ángulo φ_5 entre los vectores $\mathbf{v}_3$ y $\mathbf{v}_6$ en el plano de $\mathbf{v}_2$, que representa al ángulo de inclinación del mástil: positivo hacia afuera/negativo hacia adentro de la máquina.

Se define mediante dos ecuaciones linealmente dependientes, una para el seno del ángulo y otra para el coseno. Dependiendo de la posición del ángulo, funcionará una mejor que la otra.

$$\begin{cases} \mathbf{v}_3^{\mathrm{T}}\mathbf{v}_6 - \cos\varphi_5 = 0 \\ \mathbf{v}_2^{\mathrm{T}}(\mathbf{v}_3 \wedge \mathbf{v}_6) - \sin\varphi_5 = 0 \end{cases} \tag{2.102}$$

Fig 2.40 Grado de libertad φ_5

Además, para el ángulo φ_5, se ha introducido una ecuación de restricción de guiado cinemático porque como se verá en el apartado 5, será controlado por el usuario mediante un mando.

$$\varphi_5 - \varphi_5(t) = 0 \tag{2.103}$$

- **Mástil 2**

En el mástil 2 se define la distancia d_2 correspondiente a un cilindro hidráulico entre los puntos $\mathbf{p}_1$ y $\mathbf{p}_6$, que representa la elevación del mástil 2 respecto al mástil 1.

Se define el cilindro hidráulico como:

$$\mathbf{r}_{6,1}{}^{\mathrm{T}}\mathbf{r}_{6,1} - d_2^2 = 0 \tag{2.104}$$

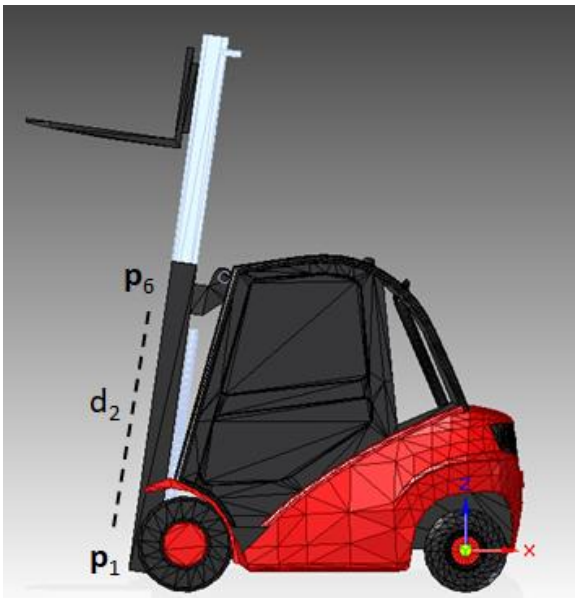

Fig 2.41 Grado de libertad d_2

- **Uñas y útil**

En las uñas y el útil se define la distancia d_1 correspondiente a un cilindro hidráulico entre los puntos $\mathbf{p}_1$ *y* $\mathbf{p}_2$, que representa la elevación de las uñas y el útil respecto al mástil1.

Se define el cilindro hidráulico como:

$$\mathbf{r}_{2,1}{}^{\mathrm{T}}\mathbf{r}_{2,1} - d_1^2 = 0 \qquad (2.105)$$

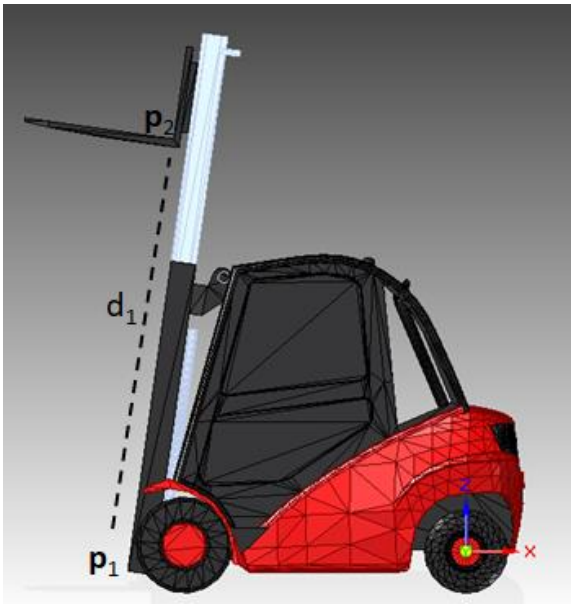

Fig 2.42 Grado de libertad d_1

Además, para la distancia d_1, se ha introducido una ecuación de restricción de guiado cinemático porque como se verá en el apartado 5, será controlada por el usuario mediante un mando.

$$d_1 - d_1(t) = 0 \qquad (2.106)$$

Por último se relaciona esta distancia d_1 con la distancia d_2 definida anteriormente puesto que el útil y las uñas se elevan el doble que el mástil 2.

$$d_1 - 2 \cdot d_2 = 0 \qquad (2.107)$$

A modo de resumen, se concluye que se consideraron 9 grados de libertad dinámicos: 6 GDL para el movimiento del chasis, 2 GDL para las rotaciones de las ruedas traseras y un 1 GDL para el balanceo del eje rígido basculante trasero. Además, se consideraron 7 GDL guiados cinemáticamente: el ángulo de cabeceo del mástil, la elevación del mástil, de las uñas/útil, los ángulos de giro representando la dirección de las ruedas traseras (relacionados por la condición de giro de Ackerman) y las rotaciones de las ruedas delanteras relacionadas en un único ángulo diferencial. Para los movimientos guiados, se emplearon restricciones reónomas. Por lo tanto se consideraron 16 grados de libertad de la máquina real.

2.4.4 Definición de los sólidos carga

Se han definido tres cargas distintas para el simulador.

- **Pallet**

La primera carga es un pallet vacío de medidas 1200 x 800 mm.

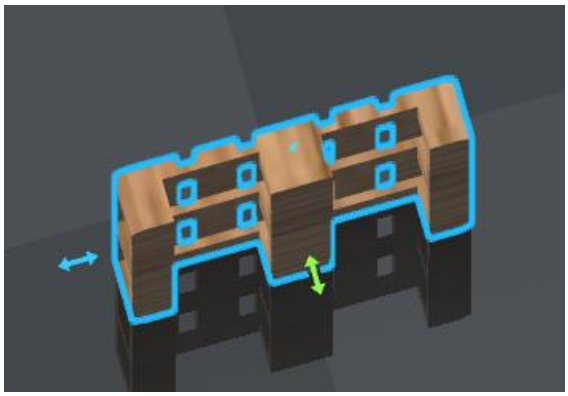

Fig 2.43 Carga 1 del simulador. Pallet

Es necesario definir el pallet porque se simulará su dinámica de la misma forma que se ha hecho con los sólidos anteriores, con la excepción de que este es un sólido libre, es decir, que no estará ligado a ningún otro.

Como variables para definir el pallet se han utilizado:

- Cuatro puntos: $\mathbf{p}_{81}$, $\mathbf{p}_{82}$, $\mathbf{p}_{83}$ *y* $\mathbf{p}_{84}$
- Un vector: $\mathbf{v}_{85}$

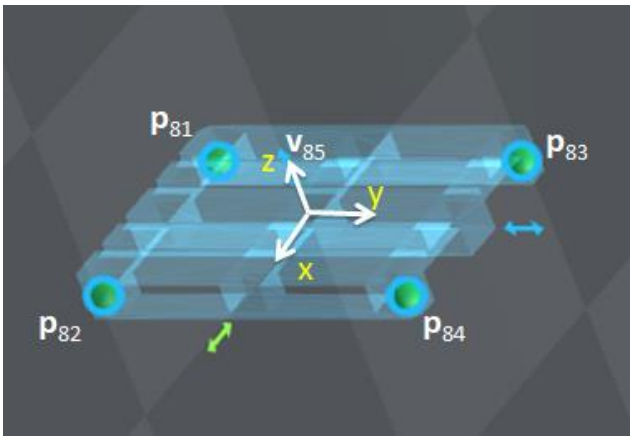

Fig 2.44 Esquema de puntos y vectores del pallet

Cada punto con los que se representa el modelo, tiene una posición concreta que en este caso representan las cuatro esquinas inferiores del pallet.

Por su parte el vector unitario $\mathbf{v}_{85}$, representa a $\mathbf{k}$ en ejes globales.

Para modelizar el pallet se han utilizado tres puntos y un vector, $\mathbf{p}_{81}$, $\mathbf{p}_{82}$, $\mathbf{p}_{83}$ *y* $\mathbf{v}_{85}$ en este mismo orden, ya que la MBSLIM construye el triedro basándose en este orden. En el Anexo I, en el fichero lectdatos.f90, se puede ver este mismo orden de introducción, quedando así el triedro definido de la siguiente forma:

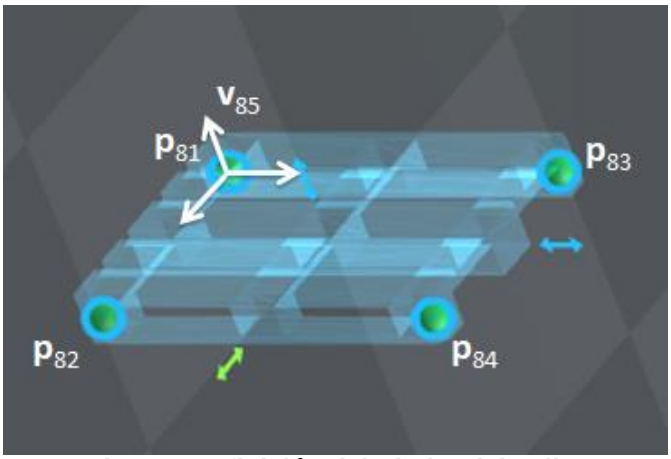

Fig 2.45 Definición del triedro del pallet

Las restricciones introducidas son:

$$\mathbf{v}_{85}^{\mathrm{T}}\mathbf{v}_{85}-1=0 \tag{2.108}$$

$$\mathbf{r}_{81,82}^{\mathrm{T}}\mathbf{r}_{81,82}-L_{81,82}^{2}=0 \tag{2.109}$$

$$\mathbf{r}_{81,83}^{\mathrm{T}}\mathbf{r}_{81,83}-L_{81,83}^{2}=0 \tag{2.110}$$

$$\mathbf{r}_{81,82}^{\mathrm{T}}\mathbf{r}_{81,83}-k_{21}=0 \tag{2.111}$$

$$\mathbf{r}_{81,82}^{\mathrm{T}}\mathbf{v}_{85}-k_{22}=0 \tag{2.112}$$

$$\mathbf{r}_{81,83}^{\mathrm{T}}\mathbf{v}_{85}-k_{23}=0 \tag{2.113}$$

$$\mathbf{r}_{81,84}-\alpha_{84}\cdot\mathbf{r}_{81,82}-\beta_{84}\cdot\mathbf{r}_{81,83}-\gamma_{84}\cdot\mathbf{v}_{85}=0 \tag{2.114}$$

El sólido tiene 15 coordenadas variables y 9 ecuaciones de restricción, 1 de vector unitario, 2 de distancia, 3 de producto escalar constante y 3 de combinación lineal, por lo que se introducen 15 variables y 9 incógnitas, lo que deja 6 grados de libertad.

Como se han visto cuando se definieron los grados de libertad del chasis, se refieren al movimiento en un espacio tridimensional, es decir, la capacidad de moverse hacia delante/atrás, arriba/abajo, izquierda/derecha (traslación en tres ejes perpendiculares), combinados con la rotación sobre tres ejes perpendiculares (Guiñada, Cabeceo, Alabeo). El movimiento a lo largo de cada uno de los ejes es independiente de los otros, y cada uno es independiente de la rotación sobre cualquiera de los ejes.

- **Virola**

La segunda carga es una virola, carga que llevaba la caretilla elevadora real en el momento del accidente. Se compone de un apoyo y el neumático, estando éste en pleno proceso de transformación.

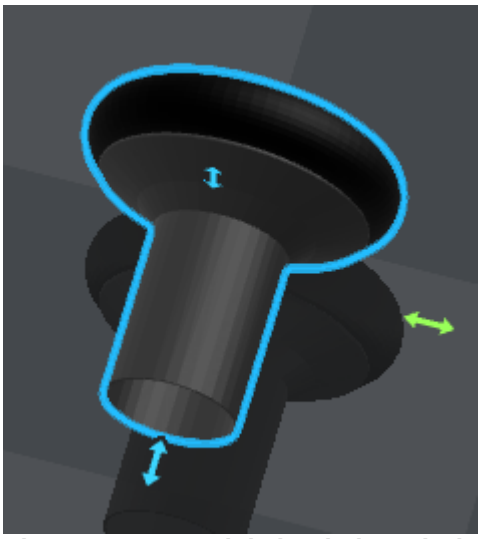

Fig 2.46 Carga 2 del simulador. Virola

Es necesario definir la virola porque se simulará su dinámica de la misma forma que hemos hecho con los sólidos anteriores, con la excepción de que este será un sólido libre, es decir, que no estará ligado a ningún otro.

Como variables para definir la virola se han utilizado:

- Cinco puntos: $\mathbf{p}_{57}, \mathbf{p}_{70}, \mathbf{p}_{71}, \mathbf{p}_{72}$ *y* $\mathbf{p}_{73}$
- Tres vectores: $\mathbf{v}_7, \mathbf{v}_8$ *y* $\mathbf{v}_9$

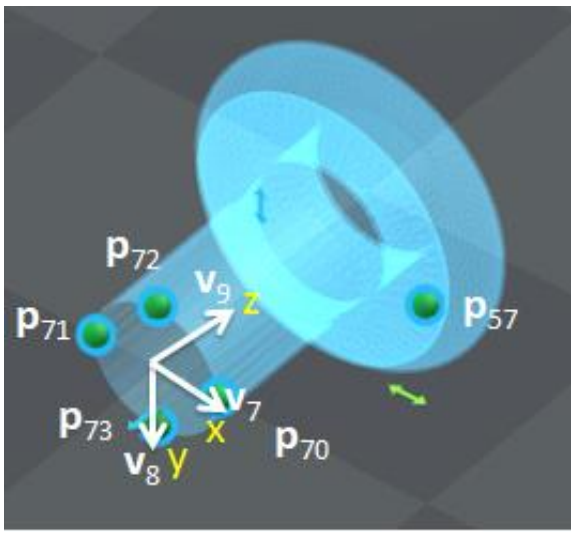

Fig 2.47 Esquema de puntos y vectores de la virola

El punto $\mathbf{p}_{57}$ con el que se representa el modelo, tiene una posición concreta que representa un punto en el neumático de la virola. El resto de los cuatro puntos $\mathbf{p}_{70}, \mathbf{p}_{71}, \mathbf{p}_{72}$ *y* $\mathbf{p}_{73}$ representan cuatro puntos de la base de la virola.

Por su parte los vectores unitarios $\mathbf{v}_7, \mathbf{v}_8$ *y* $\mathbf{v}_9$, representan a $\mathbf{i},\mathbf{j},\mathbf{k}$ en ejes globales respectivamente.

Para modelizar la virola se han utilizado un punto y tres vectores, $\mathbf{p}_{57}, \mathbf{v}_7, \mathbf{v}_8$ *y* $\mathbf{v}_9$ en este mismo orden, ya que la MBSLIM construye el triedro basándose en este orden. En el Anexo I, en el fichero lectdatos.f90, se puede ver este mismo orden de introducción, quedando así el triedro definido de la siguiente forma:

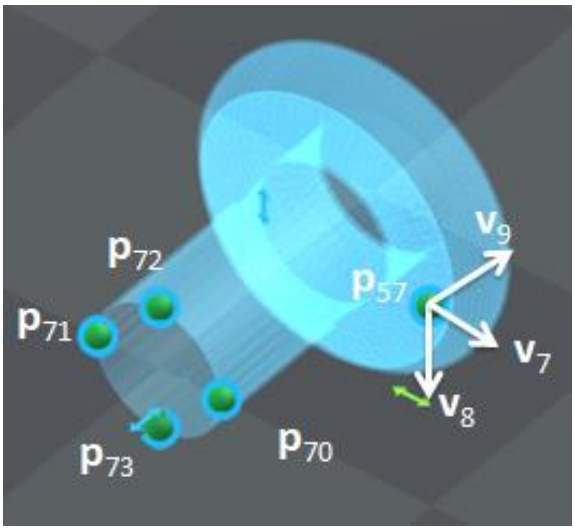

Fig 2.48 Definición del triedro de la virola

Las restricciones introducidas son:

$$\mathbf{v}_7^{\mathrm{T}}\mathbf{v}_7 - 1 = 0 \quad (2.115)$$

$$\mathbf{v}_8^{\mathrm{T}}\mathbf{v}_8 - 1 = 0 \quad (2.116)$$

$$\mathbf{v}_9^{\mathrm{T}}\mathbf{v}_9 - 1 = 0 \quad (2.117)$$

$$\mathbf{v}_7^{\mathrm{T}}\mathbf{v}_8 - k_{24} = 0 \quad (2.118)$$

$$\mathbf{v}_7^{\mathrm{T}}\mathbf{v}_9 - k_{25} = 0 \quad (2.119)$$

$$\mathbf{v}_8^{\mathrm{T}}\mathbf{v}_9 - k_{26} = 0 \quad (2.120)$$

$$\mathbf{r}_{57,70} - \alpha_{70}\cdot\mathbf{v}_7 - \beta_{70}\cdot\mathbf{v}_8 - \gamma_{70}\cdot\mathbf{v}_9 = 0 \quad (2.121)$$

$$\mathbf{r}_{57,71} - \alpha_{71}\cdot\mathbf{v}_7 - \beta_{71}\cdot\mathbf{v}_8 - \gamma_{71}\cdot\mathbf{v}_9 = 0 \quad (2.122)$$

$$\mathbf{r}_{57,72} - \alpha_{72}\cdot\mathbf{v}_7 - \beta_{72}\cdot\mathbf{v}_8 - \gamma_{72}\cdot\mathbf{v}_9 = 0 \quad (2.123)$$

$$\mathbf{r}_{57,73} - \alpha_{73}\cdot\mathbf{v}_7 - \beta_{73}\cdot\mathbf{v}_8 - \gamma_{73}\cdot\mathbf{v}_9 = 0 \quad (2.124)$$

El sólido tiene 24 coordenadas variables y 18 ecuaciones de restricción, 3 de vector unitario, 3 de producto escalar constante y 12 de combinación lineal, por lo que se introducen 24 variables y 18 incógnitas, lo que deja 6 grados de libertad.

Como se han visto con la anterior carga, el pallet, se refieren al movimiento en un espacio tridimensional, es decir, la capacidad de moverse hacia delante/atrás, arriba/abajo, izquierda/derecha (traslación en tres ejes perpendiculares), combinados con la rotación sobre tres ejes perpendiculares (Guiñada, Cabeceo, Alabeo). El movimiento a lo largo de cada uno de los ejes es independiente de los otros, y cada uno es independiente de la rotación sobre cualquiera de los ejes.

- **Pallet cargado**

La tercera carga es un pallet cargado de medidas 1200 x 1000 mm. El peso de la carga puede ser modificado en el propio programa para probar maniobras con pesos de carga diferentes.

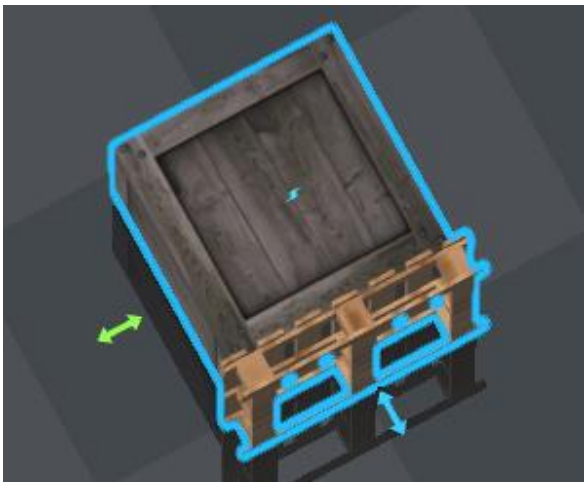

Fig 2.49 Carga 3 del simulador. Pallet cargado

Es necesario definir el pallet con la carga porque se simulará su dinámica de la misma forma que se ha hecho con los sólidos anteriores, con la excepción de que este será un sólido libre, es decir, que no estará ligado a ningún otro.

Como variables para definir el pallet cargado se han utilizado:

- Cuatro puntos: $\mathbf{p}_{91}$, $\mathbf{p}_{92}$, $\mathbf{p}_{93}$ *y* $\mathbf{p}_{94}$
- Un vector: $\mathbf{v}_{95}$

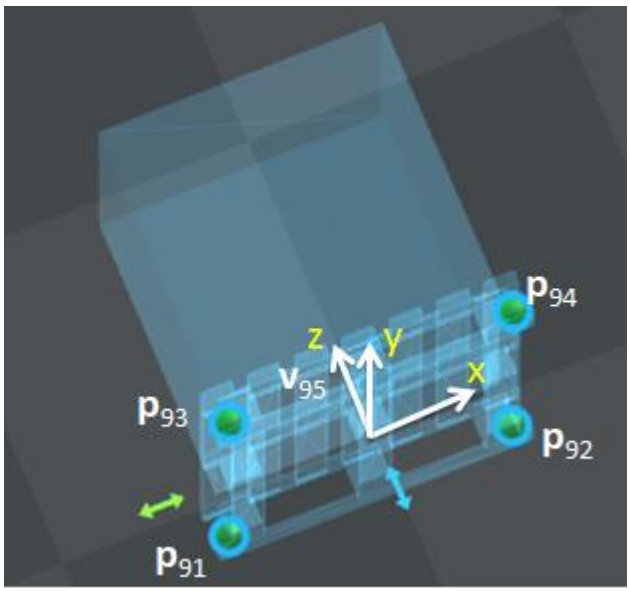

Fig 2.50 Esquema de puntos y vectores del pallet cargado

Cada punto con los que se representa el modelo, tiene una posición concreta que en este caso representan las cuatro esquinas inferiores del pallet.

Por su parte el vector unitario $\mathbf{v}_{95}$, representa a $\mathbf{k}$ en ejes globales.

Para modelizar el pallet cargado se han utilizado tres puntos y un vector, $\mathbf{p}_{91}$, $\mathbf{p}_{92}$, $\mathbf{p}_{93}$ *y* $\mathbf{v}_{95}$ en este mismo orden, ya que la MBSLIM construye el triedro basándose en este orden. En el Anexo I, en el fichero lectdatos.f90, se puede ver este mismo orden de introducción, quedando así, el triedro definido de la siguiente forma:

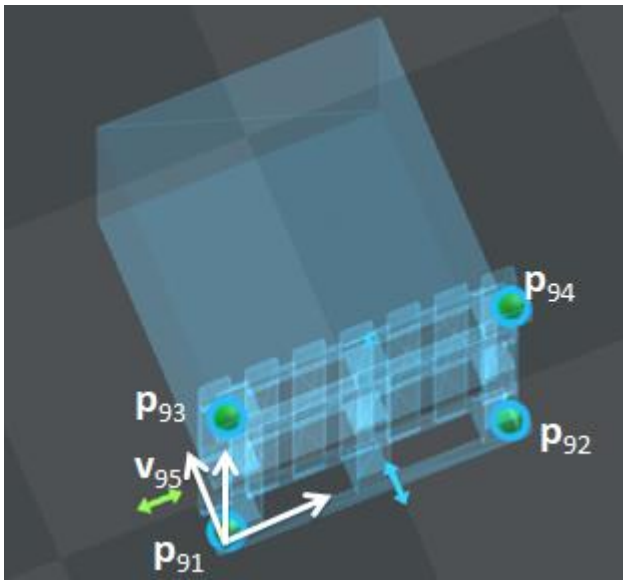

Fig 2.51 Definición del triedro del pallet cargado

Las restricciones introducidas son:

$$\mathbf{v}_{95}^{\mathrm{T}}\mathbf{v}_{95}-1=0 \tag{2.125}$$

$$\mathbf{r}_{91,92}^{\mathrm{T}}\mathbf{r}_{91,92}-L_{91,92}^{2}=0 \tag{2.126}$$

$$\mathbf{r}_{91,93}^{\mathrm{T}}\mathbf{r}_{91,93}-L_{91,93}^{2}=0 \tag{2.127}$$

$$\mathbf{r}_{91,92}^{\mathrm{T}}\mathbf{r}_{91,93}-k_{27}=0 \tag{2.128}$$

$$\mathbf{r}_{91,92}^{\mathrm{T}}\mathbf{v}_{95}-k_{28}=0 \tag{2.129}$$

$$\mathbf{r}_{91,93}^{\mathrm{T}}\mathbf{v}_{95}-k_{29}=0 \tag{2.130}$$

$$\mathbf{r}_{91,94}-\alpha_{94}\cdot\mathbf{r}_{91,92}-\beta_{94}\cdot\mathbf{r}_{91,93}-\gamma_{94}\cdot\mathbf{v}_{95}=0 \tag{2.131}$$

El sólido tiene 15 coordenadas variables y 9 ecuaciones de restricción, 1 de vector unitario, 2 de distancia, 3 de producto escalar constante y 3 de combinación lineal, por lo que se introducen 15 variables y 9 incógnitas, lo que deja 6 grados de libertad.

Como se ha visto cuando se definieron los grados de las demás cargas, se refieren al movimiento en un espacio tridimensional, es decir, la capacidad de moverse hacia delante/atrás, arriba/abajo, izquierda/derecha (traslación en tres ejes perpendiculares), combinados con la rotación sobre tres ejes perpendiculares (Guiñada, Cabeceo, Alabeo). El movimiento a lo largo de cada uno de los ejes es independiente de los otros, y cada uno es independiente de la rotación sobre cualquiera de los ejes.

2.4.5 El vector de variables

Ahora que ya se han definido las variables y las restricciones, ya se puede conocer el vector $\mathbf{q}$ de variables.

Como se ha visto en el apartado 2.4.3, existen 96 variables correspondientes a coordenadas naturales, a las que se han añadido 11 variables (9 ángulos y 2 distancias) con la definición de los grados de libertad en coordenadas relativas. Sumándole a estas, las 15, 24 y 15 variables con las que se definen las cargas, se tiene un total de 161 variables.

El vector $\mathbf{q}$ recoge el valor de todas las variables en cada instante de tiempo.

2.4.6 Fuerzas

En este apartado se hablará de los distintos tipos de fuerzas que atañen a los sólidos modelados del simulador.

2.4.6.1 Fuerzas gravitatorias

Las fuerzas gravitatorias han sido aplicadas en sentido contrario al vector unitario $\mathbf{v}_z$ en ejes globales, que se puede ver, por ejemplo, en la Fig 2.6. La gravedad toma un valor de 9,81 m/s^2 y todos los sólidos modelados en la simulación tienen esta fuerza aplicada en su centro de masas.

2.4.6.2 Modelos de fuerzas

Se distinguen los siguientes modelos de fuerzas:

- **Transmisión**

Como se ha visto en el apartado 1.3, el sentido de la marcha y la velocidad de la carretilla se regulan mediante dos pedales aceleradores a través de la bomba hidráulica de desplazamiento variable (sentido de la marcha hacia delante y hacia atrás).

Para modelizar esto, ya que la transmisión de la carretilla la guía el usuario desde dos pedales (marcha hacia delante y hacia atrás) como se verá en el apartado 5, se calcula la aceleración del ángulo φ_9, correspondiente al diferencial de la transmisión, como la aceleración angular máxima por el valor del actuador que se le está dando desde el pedal. Y en caso que el valor del actuador, sea menor al anterior dado en el instante posterior, la aceleración calculada para φ_9 es igual al caso anterior pero de signo contrario. Así se consigue que la carretilla elevadora frene proporcionalmente a lo que el usuario levanta el pie del acelerador.

- **Modelo de neumáticos**

Como modelo de neumático se ha utilizado un modelo dirigido al análisis dinámico de vehículos y al diseño de sistemas de control (TmEasy). El comportamiento vertical consiste en un modelo muelle-amortiguador que puede ser lineal o no. Las fuerzas horizontales del neumático se explican por las relaciones lineales entre el deslizamiento y las fuerzas resultantes. El deslizamiento combinado no se tiene en cuenta. Con esto, se utilizan las siguientes expresiones de fuerza:

$$F_z = a \cdot \Delta z + d_R \cdot \Delta \dot{z} \tag{2.132}$$

$$F_x = F \frac{s_x}{s_g} \tag{2.133}$$

$$F_y = F \frac{s_y}{s_g} \tag{2.134}$$

Siendo s_x y s_y los pseudodeslizamientos en los ejes globales x e y, y siendo s_g el módulo de los mismos. En nuestro caso particular, pseudodeslizamiento crítico total será estimado como 0,2.

Este modelo de fuerza se puede ver descrito con detalle en:

Rill, G., 2009. *"Dynamic Tire Forces With Smooth Transition to Stand–Still,"* 7th EUROMECH Solid Mechanics Conference, Lisbon, Portugal. [33] Hirschberg, W., Rill, G., and Weinfurter, H., 2007, "Tire Model TMeasy," Veh. Syst. Dyn., 45, pp. 101–119.

- **Fuerzas de contacto**

Se distingue entre los contactos tipo esfera-plano y malla-malla.

- **Contacto tipo esfera-plano**

La fuerza de contacto que aparece si cualquiera esfera con la que definimos un cuerpo (apartado 4.2.2) interfiere con el límite del terreno. Estas fuerzas han sido implementadas en el contacto de las cargas con el suelo. Se modeliza con un sistema muelle-amortiguador en la dirección normal al plano de contacto, aplicando así el modelo de contacto de Kelvin-Voigh.

$$\mathbf{F}_{rad}^{lin} = (-k_m \cdot (s - s_0) - amort \cdot c_m \cdot (\mathbf{v}^{\mathrm{T}} \cdot \mathbf{n}_{fl}))\mathbf{n}_{fl} \tag{2.135}$$

Donde *k* es la constante de rigidez necesaria para soportar el peso de la carga en dos puntos de apoyo con una penetración dada y *c* es la constante de amortiguamiento en cada uno de los cuatro puntos de contacto, ***v*** es la velocidad del punto del sólido que coincide con el punto de contacto entre la esfera y el entorno, ***n*** es el vector normal a la superficie del entorno en el punto de contacto.

$$k = \frac{m \cdot g}{2\Delta x} \tag{2.136}$$

$$c = \frac{\sqrt{k \cdot m}}{2} \tag{2.137}$$

- **Contacto tipo malla-malla**

Este tipo de fuerza aparece si cualquier punto de un sólido interfiere con un punto de otro sólido. Estas fuerzas han sido implementadas entre las uñas de la horquilla y los distintos tipos de carga. Además para obtener resultados de posibles vuelcos de la carretilla, se han implementado también este tipo de colisiones entre la horquilla y el chasis contra el suelo.

Igual que el contacto esfera-plano, este tipo de contacto también se modeliza con un sistema muelle-amortiguador en la dirección normal al plano de contacto, aplicando igualmente el modelo de Kelvin-Voigh. Las diferencias con el tipo de contacto esfera-plano, es la forma de calcular cuantas regiones de contacto hay, cuál es la deformación local en cada una y cuál es el vector normal a la superficie de contacto en cada región.

$$\mathbf{F}_{rad}^{lin} = (-k_m \cdot (s - s_0) - amort \cdot c_m \cdot (\mathbf{v}^{\mathrm{T}} \cdot \mathbf{n}_{fl}))\mathbf{n}_{fl} \tag{2.138}$$

Este modelo de fuerza se puede ver descrito con detalle en:

Dopico, D.; Luaces, A.; Gonzalez, M. & Cuadrado, J. *Dealing with multiple contacts in a human-in-the-loop application Multibody System Dynamics*, 2011, 25, 167-183

- **Fuerzas de rozamiento**

Las fuerzas de rozamiento son implantadas en todos los contactos que han sido mencionados anteriormente, ya sean del tipo esfera-plano, malla-malla e incluso en el modelo de neumático modelando.

Este tipo de fuerza aparece en el momento que existe un contacto normal a un punto, ya sea entre sólidos o con el suelo, añadiendo así una fuerza de fricción. De no implantar estas

fuerzas de rozamiento, los sólidos formarían un sistema conservativo y cuando estuviesen en movimiento en determinadas posiciones no volverían a estabilizarse. Se definen de la siguiente manera:

$$F_{roz} = s \cdot F_{stick} + (1-s) \cdot F_{slide} - \mu_{visc} \mathbf{v}_{proy} \tag{2.139}$$

En nuestro caso se desprecia la viscosidad por lo que se anularía el último término. Además:

$$s = exp(-(|\ (\dot{\mathbf{v}} - (\mathbf{v}^{\mathrm{T}} \cdot \mathbf{n}_{fl}) \cdot \mathbf{n}_{fl})| / v_{stick})^2) \tag{2.140}$$

$$v_{stick} = \mu \cdot g \cdot N \cdot \Delta t \tag{2.141}$$

$\mathbf{F}_{stick}$ es la fuerza de un muelle-amortiguador (proyectado perpendicularmente al plano) actuando entre los puntos p_0 y p con las constantes definidas por, $k_{stick} = m \cdot \left(\frac{1}{N \cdot \Delta t} \right)^2$ y $c_{stick} = 2 \cdot \sqrt{k_{stick} \cdot m}$ mientras que $\mathbf{F}_{slide}$ es rozamiento seco de Coulomb con coeficiente μ.

$$\mathbf{F}_{slide} = \left\{ \begin{array}{c} 0;\ |\mathbf{v}_t| = 0 \\ -\mu_{din} |\mathbf{F}_{rad}| \frac{\mathbf{v}_t}{|\mathbf{v}_t|};\ |\mathbf{v}_t| > 0 \end{array} \right\} \tag{2.142}$$

$$\mathbf{F}_{stick} = \left\{ \begin{array}{c} F_{ma2p} - (n_{fl}^T \cdot F_{ma2p}) n_{fl};\ |F_{stick}| \leq \mu_{st} |F_{rad}| \\ \mathbf{F}_{satur};\ |F_{stick} > \mu_{st} |F_{rad}| \end{array} \right\} \tag{2.143}$$

Este modelo de fuerza se puede ver descrito con detalle en:

Dopico, D.; Luaces, A.; Gonzalez, M. & Cuadrado, J. *Dealing with multiple contacts in a human-in-the-loop application Multibody System Dynamics*, 2011, 25, 167-183

2.4.6.3 Momentos tope

Respeto a los momentos tope, son los que se aplican en los puntos límite de recorrido en el giro del eje basculante. Son momentos del tipo muelle-amortiguador a torsión.

$$M = k \cdot \varphi + c \cdot \dot{\varphi} \tag{2.144}$$

Donde *k* es la constante de rigidez y se estima para una penetración máxima de un grado *y* c es el amortiguamiento crítico que también estimamos.

3 LA MBSLIM: UN SOFTWARE PARA LA SIMULACIÓN DINÁMICA DE SISTEMAS MULTICUERPO

MBSLIM es una biblioteca para la simulación dinámica de sistemas multicuerpo genéricos, desarrollada desde el año 2007 en el Laboratorio de Ingeniería Mecánica de la Universidad de A Coruña. Además de su principal utilidad, que es la simulación dinámica, la MBSLIM cuenta con algunas capacidades extra como son la simulación cinemática o la resolución del equilibrio estático de mecanismos, aunque se centra especialmente en la dinámica ya que es un problema mucho más complicado de resolver.

Para llevar a cabo las tareas de simulación, la biblioteca plantea y resuelve las ecuaciones del movimiento de mecanismos o máquinas definidos por el usuario, con ayuda de las funciones incluidas en la misma.

La novedad de MBSLIM respecto de otros paquetes de software, es que incluye formulaciones de las ecuaciones del movimiento muy avanzadas, en fase de investigación, que no están disponibles en otros códigos y que permiten resolver problemas muy específicos en los que otros métodos fallan.

El software ha sido desarrollado en Fortran 2003 como una colección de módulos y ha sido verificado en varias plataformas, compiladores y sistemas operativos distintos. La MBSLIM es completamente general y puede simular todo tipo de sistemas multicuerpo tridimensionales. La biblioteca cuenta también con un módulo que permite interactuar con Matlab, enviando datos, lanzando cálculos y recuperando resultados si fuera necesario.

Los módulos que han sido empleados de la biblioteca son los siguientes:

- constantes
- estado
- eventos
- formulacion_cinematica
- formulaciones
- fuerzas_generalizadas
- fuerzas_primitivas
- jacob
- jacob_log
- math_oper
- matlab_caller
- restric
- restricciones
- solidos
- tipos_derivados

3.1 Módulo constantes

Módulo que contiene los parámetros del solver.

Las funciones utilizadas de este módulo son:

- *inicializa_CONSTANTES* (time_step, penaltycoef, NRerrorkind, NRtol, NRmaxite, numerip, numeriv, numeris, numeria, linsolver, vartime_step, maxtime_step, mintime_step, userNRT)

Inicialización de las contantes del solver.

Dónde:

- time_step: paso de tiempo, para integradores de paso de tiempo fijo, o paso de tiempo inicial, para integradores de paso de tiempo variable.
- penaltycoef: coeficiente para los términos de penalización.
- NRerrorkind,NRtol,NRmaxite: tipo de error de integración.
- numerip,numeriv,numeris,numeria: reserva de número de puntos, vectores, distancias y ángulos. Sólo se pueden pasar la primera vez que se llama a la función.
- LinSolver: solver para los sistemas de ecuaciones lineales.
- vartime: paso de tiempo fijo o variable.El valor por defecto es paso de tiempo fijo pero si se introduce y su valor es .true. el integrador varía el paso de tiempo en función de la tolerancia NRtol.
- maxtime_step,mintime_step: pasos de tiempo máximo y mínimo para los integradores de paso variable. Si la opción de paso de tiempo variable "vartime" está activa y el paso de tiempo inicial "time_step" no es mayor que "mintime_step" y menor que "maxtime_step" entonces el paso de tiempo inicial se modifica y se ajusta a estos límites inferior o superior, ignorando el parámetro "time_step".
- userNRT: número de restricciones de usuario (escritas a mano), holónomas.

- *inicializa_callbacks (ptocolision_normal, bodycolision_normal, bodycolision_normal2body, guiada, forces, stiffness_damping_matrices, restric, jacob, jacob_log)*

Inicialización de callbacks del solver: el usuario proporciona una serie de rutinas que serán llamadas por el solver en caso de necesidad. Exige llamar previamente a constantes::inicializa_CONSTANTES.

Dónde:

- ptocolision_normal, bodycolision_normal, bodycolision_normal2body: punteros a subrutinas de detección de colisiones para fuerzas de contacto de tipo puntual, de sólido y fuerza de acción/reacción de sólido respectivamente.
- guiada: puntero a subrutina de guiado de restricciones reónomas, llamada antes del predictor.
- forces: puntero a subrutina de fuerzas de usuario.
- stiffness_damping_matrices: puntero a subrutina de matrices de rigidez-amortiguamiento combinadas de usuario. En estas rutinas han de escribirse las entradas de derivadas correspondientes a las fuerzas escritas en forces.
- restric: puntero a subrutina de restricciones de usuario.
- jacob: puntero a subrutina de matriz Jacobiana de restricciones de usuario.
- jacob_log: puntero a subrutina de matriz Jacobiana lógica de restricciones de usuario.

- *time_step ()*

Función que devuelve el paso de tiempo, útil para algoritmos de paso de tiempo variables.

Las variables utilizadas de este módulo son:

ip(n): ip recoge los índices a puntos. ip(n) devuelve el valor de la posición del punto n en el vector de variables.

iv(n): iv recoge los índices a vectores. iv(n) devuelve el valor de la posición del vector n en el vector de variables.

ia(n): ia recoge los índices a ángulos. ia(n) devuelve el valor del ángulo de la posición n en el vector de variables.

is(n): is recoge los índices a distancias. is(n) devuelve el valor de la distancia de la posición n en el vector de variables.

DIM: Contiene el número de variables.

NIN: Contiene el número de variables independientes.

dt: Indica el paso del tiempo dt=0,01s.

3.2 Módulo estado

Módulo de variables de estado del solver.

Las variables utilizadas de este módulo son:

tSim: Tiempo de simulación.

q: Vector que contiene en cada instante el valor de todas las variables del modelo.

Pos (i_MOTOR): Indica el valor del estado del actuador i_MOTOR con el que trabajamos.

Vel (i_MOTOR): Indica el valor del estado de la velocidad del actuador i_MOTOR con el que trabajamos.

Ace (i_MOTOR): Indica el valor del estado de la aceleración actuador i_MOTOR con el que trabajamos.

Las funciones utilizadas de este módulo son:

SETgdl (gdl_in, zp_in): Define en el solver las variables de $\mathbf{q}$ que serán grado de libertad y las inicializa con una velocidad inicial.

- gdl_in: vector que contiene los índices de las variables que serán GDL.
- zp_in: vector donde se inicializa la velocidad de cada grado de libertad.

a (index): Devuelve un ángulo.
- index: nombre del ángulo.

ap (index): Devuelve la derivada de un ángulo respecto del tiempo.
- index: nombre del ángulo.

r (index): Devuelve la posición de un punto en globales.
- index: nombre del punto.

rx (index): Devuelve la coordenada "x" de un punto INTEGRABLE del modelo en globales.
- index: nombre del punto.

ry (index): Devuelve la coordenada "y" de un punto INTEGRABLE del modelo en globales.
- index: nombre del punto.

rz (index): Devuelve la coordenada "z" de un punto INTEGRABLE del modelo en globales.
- index: nombre del punto.

v (index): Devuelve la posición de un vector en globales.
- index: nombre del vector.

3.3 Módulo eventos

Módulo de eventos de simulación.

Añade eventos de errores de integración generados por el propio solver o bien eventos de usuario que el propio usuario se debe encargar de llamar e interpretar.

La variable utilizada del módulo es:

gestionar_eventos: Indica información sobre problemas ocurridos durante la simulación.

3.4 Módulo formulación_cinemática

Módulo de formulaciones para simulación cinemática.

Las funciones utilizadas de este módulo son:

- *ini_posvel(t)*

Subrutina que lleva a cabo el preproceso cinemático y resolución de posición y velocidad inicial para la formulación de ALI3_P y comprueba que no existan triedros reflejados respecto de su definición local. También ajusta el tiempo inicial de simulación al tiempo indicado por el usuario.

Siendo t el tiempo inicial de simulación. El solver ajusta el tiempo de simulación estado::tSim a este valor.

3.5 Módulo formulaciones

Módulo de genérico de formulaciones. Contiene las funciones genéricas que llaman a las funciones de las distintas formulaciones.

Las funciones utilizadas de este módulo son:

- reac_restrics(nq,nr)

Subrutina genérica para obtener las reacciones asociadas a las restricciones del modelo. Dónde:

- nq: índices de fuerzas generalizadas para las que se quieren sacar las fuerzas de reacción generalizadas.
- nr: índices de restricciones para las que se quieren las fuerzas de reacción generalizadas. Las restricciones no-holónomas no contribuyen a estas reacciones.

- *finaliza_modelo*

Subrutina genérica que llama a las subrutinas de finalización del modelo para las distintas formulaciones disponibles.

Las variables utilizadas de este módulo son:

- *dealloc_memory*

Variable genérica que reserva memoria.

- *ini_acel*

Subrutina que lleva a cabo el preproceso cinemático y resolución de posición y velocidad inicial.

- *calculo*

  Avanza un paso de tiempo realizando los cálculos de la dinámica multibody.

3.6 Módulo fuerzas_generalizadas

Módulo de fuerzas generalizadas (dependientes de las coordenadas).

Este módulo:

- Crea las estructuras de fuerzas generalizadas para cada tipo de fuerza primitiva contemplada en fuerzas_primitivas.
- Ensambla dichas fuerzas generalizadas en el vector de fuerzas generalizadas global.
- Llama a las contribuciones de dichas fuerzas a las matrices de rigidez y amortiguamiento generalizadas del sistema.

Las variables utilizadas de este módulo son:

- force_normal

  Tipo de dato de los objetos sobre los que actúan fuerzas normales.

- force_tiretmeasylin

  Tipo de dato de los objetos sobre los que actúan fuerzas de neumático.

Las funciones utilizadas de este módulo son:

- *genForce(indsa, F, Km, Cm)*

Interfaz para escribir fuerzas generalizadas sobre puntos de sólidos, tanto no integrables como expresados directamente en coordenadas globales.

Es una función que escribe una fuerza sobre una coordenada 's', de distancia, o un par sobre una coordenada 'a', de ángulo, del modelo. Se emplea mediante la sobrecarga fuerzas_generalizadas::genForce.

Dónde:

- indsa: distancia dada por is(indsa) o ángulo dado por ia(indsa).
- F: fuerza aplicada sobre la distancia o par aplicado sobre el ángulo. En este caso es un escalar. Si únicamente se desea reservar el espacio en la tangente, se debe introducir F=0.
- Km,Cm: rigidez y amortiguamiento de la fuerza/par sobre la coordenada de desplazamiento. Son escalares.

- *ADDforce_normal_linl(km, cm ,rad, hystercm, cuerpo/s, idpt, handlefn, nmaxcolis, fnlimit)*

  Interfaz de fuerza normal lineal para colisiones de punto o de sólido.

  Añade una fuerza de contacto normal de punto o de sólido: modelo lineal.

  Dónde:

  - km,cm: constante elástica y constante de amortiguamiento del contacto.
  - hyster_cm: parámetro que controla la histéresis. Varía entre 0 (no se disipa en el retorno) y 1 (disipación del 100% en el retorno).
  - rad: radio de contacto
  - cuerpo/s: sólido sobre el que se aplica la fuerza (Tipo derivado BODY con atributo TARGET).
  - idpt: nombre del punto del cuerpo sobre el que se aplica la fuerza en caso de colisión puntual. Si se omite este parámetro la colisión es de sólido en lugar de puntual.

- handlefn: puntero a la fuerza normal creada. Sirve como entrada a las funciones de rozamiento para añadir rozamiento a un contacto normal existente.
- nmaxcolis: número máximo de colisiones simultáneas. Define el número máximo permitido de caras en colisión simultánea con un punto.
- fnlimit: fuerza límite de contacto normal. Limita la fuerza de contacto a este valor opcional. Sirve para evitar fuerzas de contacto extraordinariamente altas en aplicaciones en las que, debido a un paso de tiempo excesivamente alto o una evaluación simplificada de las colisiones, puedan surgir fuerzas arbitrariamente elevadas. Esta limitación es independiente para cada una de las caras de colisión con el punto.

- *ADDforce_friction_DOPI(handle, mu_din, vstick, kstick, cstick, p0updcoef, mu_st, mu_visc)*

Añade fricción a un contacto normal sobre un punto.

Dónde:

- handlefn: puntero a una fuerza normal creada previamente. El rozamiento se añade al contacto normal indicado por este parámetro.
- mu_din: coeficiente de rozamiento seco de Coulomb (dinámico).
- vstick: velocidad umbral para el sticktion a bajas velocidades. Por debajo de vstick se contempla la posibilidad de sticktion. Para tener una estimación del valor de vstick necesario para el sistema se puede usar la siguiente expresión $v_{stick} = \mu \cdot g \cdot N \cdot \Delta t$ siendo N el número de pasos de tiempo necesarios para que se pare el sistema a partir de la aplicación del sticktion. Valores de N entre 1 y 5 funcionan bien, por debajo de 1 empieza a haber problemas.
- kstick, cstick: constantes de rigidez y amortiguamiento del muelle amortiguador de sticktion. Para calcular kstick se aproxima el sistema por uno de masa puntual y se aplica la siguiente fórmula: $k_{stick} = m \cdot \left(\frac{1}{N \cdot \Delta t} \right)^2$. El amortiguamiento cstick se puede elegir para que el sistema sea críticamente amortiguado: $c_{stick} = 2 \cdot \sqrt{k_{stick} \cdot m}$.
- p0updcoef: contola la actualización del punto de soldadura p0 en los siguientes supuestos: 1) Se superó la fuerza máxima del contacto. 2) La velocidad es superior a la velocidad umbral vstick. Un valor de "0" supone un destensado total del muelle de sticktion, mientras que un valor de "1" conserva siempre una tensión igual al máximo valor de rozamiento disponible.
- mu_st: coeficiente de rozamiento estático o de sticktion. En general es ligeramente superior al coeficiente de rozamiento dinámico.
- mu_visc: coeficiente de rozamiento viscoso.

- *ADDforce_tiretmeasylin(handle, slipc, mu, veps, cx, dx, cy, dy)*

Añade neumático a un contacto normal sobre una rueda: modelo TMeasy con curvas linealizadas. La rueda tiene que ser un sólido 3D modelizado con 1 punto y 3 vectores, siendo el primero de los vectores de modelización el vector según el eje de la rueda y los otros dos perpendiculares formando un triedro orientado a derechas.

Dónde:

- handlefn: puntero a una fuerza normal creada previamente. El neumático se añade al contacto normal indicado por este parámetro.
- slipc: pseudodeslizamiento crítico total.
- mu: coeficiente de fricción máximo.
- veps: valor pequeño.

- *modifyforce_normall(handlefn, km, ce_cm, rad, expn_hystercm, nmaxcolis, nulldeltap0, fnlimit, ncolis, deltap0)*

Modifica una fuerza de contacto normal ya existente sobre un sólido (fuerza de acción de cualquier tipo, pero sin reacción).

Dónde:

- handlefn: puntero a la fuerza normal creada, tal y como lo devuelve la rutina "ADDforce_normal_****" empleada para crear la fuerza normal.
- km, cm: constante elástica y constante de amortiguamiento del contacto.
- rad: radio del contacto.
- expn_hystercm: parámetro que controla la histéresis o exponente de Hertz, dependiendo del tipo de fuerza normal que se modifique.
- nmaxcolis: número máximo de colisiones simultáneas. Define el número máximo permitido de caras en colisión simultánea con un punto.
- nulldeltap0: valor tomado para deltap0 cuando el cálculo arroja un valor nulo, ya que en este caso la fuerza de contacto se hace singular.
- fnlimit: fuerza límite de contacto normal. Limita la fuerza de contacto a este valor opcional. Sirve para evitar fuerzas de contacto extraordinariamente altas en aplicaciones en las que, debido a un paso de tiempo excesivamente alto o una evaluación simplificada de las colisiones, puedan surgir fuerzas arbitrariamente elevadas. Esta limitación es independiente para cada una de las caras de colisión con el punto.
- ncolis: número real de colisiones simultáneas en el instante actual. No es un parámetro del contacto, sino una variable de estado del contacto pero se incluye en esta función porque es necesario modificarlo para reiniciar simulaciones con contacto a partir de un estado previamente guardado.
- deltap0: valor real de la velocidad de impacto inicial para el contacto. No es un parámetro del contacto, sino una variable de estado del contacto pero se incluye en esta función porque es necesario modificarlo para reiniciar simulaciones con contacto a partir de un estado previamente guardado.

3.7 Módulo fuerzas_primitivas

En este módulo están definidas todas aquellas fuerzas de interés para el usuario, como fuerzas de muelles entre puntos o distancias, fuerzas de contacto, fuerzas elásticas, fuerzas de neumático, fuerzas de freno, etc. Se trata de fuerzas primitivas por lo tanto deben ser programadas de forma independiente de las coordenadas para poder ser usadas por el solver global en coordenadas naturales y por el solver recursivo en coordenadas relativas indistintamente.

La función utilizada del módulo es:

- *Fmuelle_amortiguador (km, cm, s, s0, sp, fm)*

Fuerza de un muelle-amortiguador lineal que actúa sobre una distancia.

$$f_m^{lin} = -k_m(s - s_0) - c_m \cdot \dot{s} \tag{3.1}$$

Dónde:

- km, cm: constante elástica del muelle y constante de amortiguamiento del amortiguador.
- s, sp: índices a la variable de distancia y su derivada.
- s0: longitud natural del muelle.

- fm: fuerza del muelle-amortiguador.

3.8 Módulo jacob

Módulo de jacobiano de restricciones primitivas. Como ocurría con JACOB_LOG_MOD, no es un módulo de usuario, es usado por el solver, excepto si son necesarias restricciones que no estén implementadas en CONSTRAINTS_MOD, en cuyo caso el usuario debe emplearlo directamente.

La función utilizada del módulo es:

- *j_set (ir, indq, val)*

Introduce el valor en la matriz jacobiana de restricciones si la posición está reservada en la matriz jacobiano lógico. Esta función está pensada para dar valor a la matriz jacobiana de restricciones escritas a mano por el usuario cuando no existe ninguna restricción de la biblioteca que se adapte a las necesidades del usuario.

Dónde:

- ir: índice de la restricción (fila) en la que se quiere escribir de 1 a userNRT. Recuérdese que el número de restricciones de usuario deseadas se debe inicializar en CONSTANTES::inicializa_CONSTANTES mediante la variable userNRT.
- indq: índice de la variable respecto a la que se deriva. Columna de Fiq.
- val: valor del elemento de la matriz jacobiana Fiq_user(ir,indq)=val.

3.9 Módulo jacob_log

Módulo de jacobiano lógico de restricciones primitivas. No es un módulo de usuario, es usado por el solver, excepto si son necesarias restricciones que no estén implementadas en CONSTRAINTS_MOD, en cuyo caso el usuario debe emplearlo directamente.

La MBSLIM utiliza matrices sparse para el cálculo, el jacobiano lógico reserva espacios de memoria en la matriz del jacobiano, para que después el módulo JACOB_MOD pueda introducir valores en ellos.

La función utilizada del módulo es:

- *jl_set (ir, indq, val)*

Reserva espacio en la matriz jacobiano para restricciones escritas a mano por el usuario. Cuando no existe ninguna restricción primitiva que se adapte a las necesidades del usuario, es necesario emplear esta subrutina para escribir en el vector.

Dónde:

- ir: índice de la restricción en la que se quiere escribir. Fila de Fiq_log. Recuérdese que las restricciones de usuario comienzan en CONSTANTES::indr y el número de restricciones de usuario deseadas se debe inicializar en CONSTANTES::inicializa_CONSTANTES mediante la variable userNRT.
- indq: índice de la variable respecto a la que se deriva. Columna de Fiq_log.
- val: valor del elemento de la matriz jacobiana lógica Fiq_user_log (ir, indq)=val

3.10 Módulo math_oper

Módulo de operaciones matemáticas no intrínsecas.

Las funciones utilizadas de este módulo son:

pvect (u, v): Calcula el producto vectorial de dos vectores.

norma (r): Calcula la norma de un vector.

norm (r): Normaliza un vector.

3.11 Módulo matlab_caller

Módulo de gestión de sesiones de MATLAB engine.

Las funciones utilizadas de este módulo son:

- *matlab_caller(t_graph, y_graph, figur, linetype, linewidth)*

Plotea dos vectores de datos reales y vs x.

Dónde:

- t_graph: vector en abscisas.
- y_graph: vector en ordenadas.
- figur: figura de MATLAB.
- linetype: tipo de línea,color de línea y marker, según los códigos de MATLAB. La cadena debe tener 4 caracteres o menos, ya que es la máxima combinación posible. Si se pasa una cadena más larga se leen sólo los 4 primeros caracteres y se desprecia el resto.
- linewidth: ancho de línea.

- *matlab_evalstring(string)*

Evalúa expresión de MATLAB.

Siendo string, la cadena de texto a evaluar.

3.12 Módulo restric

Módulo de restricciones primitivas. No es un módulo de usuario, es usado por el solver, excepto si son necesarias restricciones que no estén implementadas en CONSTRAINTS_MOD, en cuyo caso el usuario debe emplearlo directamente.

Las funciones utilizadas de este módulo son:

- *r_set_esc (ir, val)*

Dar valor a restricciones (versión escalar): esta función está pensada para dar valor a restricciones escritas a mano por el usuario. Cuando no existe ni siquiera una restricción primitiva que se adapte a las necesidades del usuario, es necesario emplear esta subrutina para escribir en el vector fi. Se puede usar a través de su interfaz r_set.

Dónde:

- ir: índice de la restricción en la que se quiere escribir. Recuérdese que las restricciones de usuario comienzan en CONSTANTES::indr y el número de restricciones de usuario deseadas se debe inicializar en CONSTANTES::inicializa_CONSTANTES mediante la variable userNRT.
- val: valor de la restricción fi(ir)=val.

- r_set_vec (ir, val)

Dar valor a restricciones (versión vectorial): idéntica a r_set_esc pero escribe tres restricciones consecutivas a la vez (restricción vectorial). Se puede usar a través de su interfaz r_set.

Dónde:

- ir: índice de la restricción en la que se quiere escribir. Recuérdese que las restricciones de usuario comienzan en CONSTANTES::indr y el número de restricciones de usuario deseadas se debe inicializar en CONSTANTES::inicializa_CONSTANTES mediante la variable userNRT.
- val: valor de la restricción fi(ir)=val.

3.13 Módulo restricciones

Módulo que gestiona las restricciones.

Este módulo:

- Crea las restricciones de sólido rígido para cada tipo de modelización.
- Añade las restricciones primitivas correspondientes.
- Añade las restricciones primitivas adicionales solicitadas por el usuario.
- Elimina restricciones.
- Gestiona la creación del vector de restricciones, su matriz jacobiana y todas sus derivadas y términos asociados.

Las funciones utilizadas de este módulo son:

- *ADDrestric_distdef (mip1, mip2, mis, penaltycoef_rel)*

Añade una restricción primitiva de definición de distancia variable entre 2 puntos.

$$(\mathbf{p}_2-\mathbf{p}_1)^{\mathrm{T}}(\mathbf{p}_2-\mathbf{p}_1)-s_i^2=0 \tag{3.2}$$

Dónde:

- mip1, mip2: índices a los puntos $\mathbf{p}_1$ *y* $\mathbf{p}_2$.
- mis: índice a la variable de distancia s_i.
- penaltycoef_rel (opcional): factor que multiplica al penalizador "penaltycoef" introducido en CONSTANTES::inicializa_CONSTANTES para la restricción.

- *ADDrestric_angdef2vc (miu, miv, miw, mia, pn, pw, penaltycoef_rel)*

Añade una restricción primitiva de definición de ángulo variable entre 2 vectores unitarios $\mathbf{u}$ *y* $\mathbf{v}$. El ángulo se mide alrededor de un tercer vector unitario $\mathbf{w}$, que no tiene por qué ser perpendicular a los otros dos. Esta restricción son en realidad dos restricciones redundantes, una de coseno definida mediante producto escalar y otra de seno definida mediante producto mixto:

$$\begin{cases}\mathbf{u}^{\mathrm{T}}\mathbf{v}-\mathbf{p}_w+\mathbf{p}_n\cos\varphi=0\\ \mathbf{w}^{\mathrm{T}}(\mathbf{u}\wedge\mathbf{v})-\mathbf{p}_n\sin\varphi=0\end{cases} \tag{3.3}$$

Dónde:

- miu, miv, miw : índices a los vectores $\mathbf{u},\ \mathbf{v}$ *y* $\mathbf{w}$.
- mia: índice a la variable de ángulo φ.
- $\mathbf{p}_w$: factor de perpendicularidad con $\mathbf{w}$.
- $\mathbf{p}_n$: factor de perpendicularidad entre $\mathbf{u}$ *y* $\mathbf{v}$.

- penaltycoef_rel (opcional): factor que multiplica al penalizador "penaltycoef" introducido en CONSTANTES::inicializa_CONSTANTES para la restricción.

El vector unitario $\mathbf{w}$ tiene que ser fijo con respecto a $\mathbf{u}$ y $\mathbf{v}$, y cada uno de éstos últimos debe pertenecer a un sólido rígido. Si no se pasa el parámetro $\mathbf{p}_w$, se asume que vale cero, es decir, que al menos uno de los dos vectores $\mathbf{u}$ o $\mathbf{v}$ es perpendicular a $\mathbf{w}$.

Si tampoco se pasa el parámetro $\mathbf{p}_n$, se asume que vale uno, es decir, que los dos vectores $\mathbf{u}$ y $\mathbf{v}$ son perpendiculares a $\mathbf{w}$.

- *ADDrestric_dotproduct2vc(miu, miv, cte, penaltycoef_rel)*

Añade una restricción primitiva de producto escalar entre dos vectores.

$$\mathbf{u}^{\mathrm{T}}\mathbf{v} - cte = 0 \tag{3.4}$$

Dónde:

- miu, miv: índices a los vectores $\mathbf{u}$ y $\mathbf{v}$.
- cte: valor del producto escalar.
- penaltycoef_rel(opcional): factor que multiplica al penalizador "penaltycoef" introducido en CONSTANTES::inicializa_CONSTANTES para la restricción.

- ADDrestric_crossproduct2pt1vc (mip1, mip2, miv, penaltycoef_rel)

Añade una restricción primitiva de producto vectorial nulo entre dos vectores, uno de los cuales está formado por 2 puntos.

$$(\mathbf{p}_2 - \mathbf{p}_1) \wedge \mathbf{v} = 0 \tag{3.5}$$

Dónde:

- mip1,mip2: índices a los puntos $\mathbf{p}_1$ y $\mathbf{p}_2$.
- miv: índice al vector $\mathbf{v}$.
- penaltycoef_rel: factor que multiplica al penalizador "penaltycoef" introducido en CONSTANTES::inicializa_CONSTANTES para la restricción.

- *ADDrestric_gear (misa1, misa2, krt, kfi0,* penaltycoef_rel*)*

Añade una restricción primitiva de engranaje genérico entre dos variables.

$$sa_1 - k_{rt} \cdot sa_2 - k_{\varphi 0} = 0 \tag{3.6}$$

Dónde:

- misa1, misa2: índice a las variables relacionadas.
- k_{rt} y $k_{\varphi 0}$: relación de velocidades y offset del engranaje.
- penaltycoef_rel: factor que multiplica al penalizador "penaltycoef" introducido en CONSTANTES::inicializa_CONSTANTES para la restricción.

- *ADDrestric_guiacoord (misa, i_MOTOR,* penaltycoef_rel*)*

Añade una restricción primitiva de guiado cinemático de una distancia o ángulo.

$$sa - f(t) = 0 \tag{3.7}$$

Dónde:

- misa: índice al punto o vector correspondiente a la variable guiada.
- i_MOTOR: índice a la posición del actuador cinemático. El bucle de tiempo debe evaluar la función de guiado cinemático y sus derivadas que deben ser introducidas

en los vectores pos, vel, ace del módulo ESTADO en cada paso: pos (i_MOTOR), vel (i_MOTOR), ace (i_MOTOR) contienen la función de guiado cinemático y sus derivadas para la variable "sa".

- penaltycoef_rel: factor que multiplica al penalizador "penaltycoef" introducido en CONSTANTES::inicializa_CONSTANTES para la restricción.

3.14 Módulo solidos

Este módulo:

- Añade sólidos al modelo.
- Añade masa, momentos de inercia, centro de gravedad a los sólidos existentes.
- Añade puntos y vectores a los sólidos existentes.
- Gestiona la creación del vector de variables del modelo y de puntos y vectores fijos, apuntando los correspondientes índices de puntos, vectores, ángulos, etc. a sus posiciones.
- Elimina sólidos del modelo si es el caso.
- Añade y calcula la información correspondiente a la modelización de cada sólido, creando también sus restricciones de sólido rígido.

Las funciones utilizadas de este módulo son:

- *SETgroundnpt (npoints)*

  Función para reservar puntos en el suelo.

  Siendo npoints, el número de puntos fijos.

- *ADDgroundpt (idpt, pt)*

  Función para añadir puntos al suelo.

  Dónde:

  - idpt: crea el punto con el índice idpt.
  - pt: coordenadas del punto en globales.

- *SETgroundnvc (nvectors)*

  Función para reservar vectores en el suelo.

  Siendo npoints, el número de puntos fijos.

- *ADDgroundvc (idvc, vc)*

  Función para añadir vectores al suelo.

  Dónde:

  - idvc: crea el vector con el índice idvc.
  - vc: coordenadas del vector en globales.

- *SOLIDO3D_ADDbody (cuerpo)*

  Función para añadir cuerpos a la simulación.

  Siendo cuerpo, el nombre con el que se creará el cuerpo.

- *ADDmass (cuerpo, masa)*

  Función para añadir masa a los cuerpos.

  Dónde:

- cuerpo: nombre del cuerpo al que se añadirá masa.
- masa: valor de la masa del cuerpo.

- *ADDrg_loc (cuerpo, rg_loc)*

  Subrutina para añadir centro de gravedad a un sólido 3d.

  Dónde:

  - cuerpo: nombre del cuerpo al que se añadirá centro de gravedad.
  - rg_loc: vector de posición en coordenadas locales del centro de gravedad.

- *ADDv_Ig (cuerpo, v_Ig)*

  Subrutina para añadir un tensor de inercia a un sólido 3d.

  Dónde:

  - cuerpo: nombre del cuerpo3D al que se añadirá tensor de inercia.
  - v_Ig: tensor de inercia del sólido en ejes locales, respecto de su centro de gravedad, dado en su forma compacta.

$$I_g = \left[I_{xx}\ I_{yy}\ I_{zz}\ p_{xy}\ p_{xz}\ p_{yz} \right] \tag{3.8}$$

- *SETnpt (cuerpo, npoints)*

  Función para reservar número de puntos de los cuerpos tipo sólido 3D.

  Dónde:

  - cuerpo: sólido donde se añadirán los puntos.
  - pt: número de puntos del sólido.

- *ADDpt (cuerpo, idpt, pt, integrable, pt0)*

  Función para añadir puntos a un cuerpo tipo sólido 3D.

  Dónde:

  - cuerpo: sólido donde se añadirán los puntos.
  - idpt: crea el vector con el índice idpt.
  - pt: coordenadas del punto en locales.
  - integrable: variable lógica para indicar si el punto es integrable o no.
  - pt0: coordenadas del punto en globales.

- *SETnvc (cuerpo, nvectors)*

  Función para reservar número de vectores de los cuerpos tipo sólido 3D.

  Dónde:

  - cuerpo: sólido donde se añadirán los vectores.
  - nvectors: número de vectores del sólido

- *ADDvc (cuerpo, idvc, vc, integrable, vc0)*

  Función para añadir vectores a un cuerpo tipo sólido 3D.

  Dónde:

  - cuerpo: sólido donde se añadirán los vectores.
  - idvc: crea el vector con el índice idvc.
  - vc: coordenadas del vector en locales.
  - integrable: variable lógica para indicar si el vector es integrable o no.

- vc0: coordenadas del vector en globales.

- *ADDdistvar (iis, s0)*

  Reserva una variable de distancia adicional.

  Dónde:

  - iis: índice de la variable de distancia.
  - s0: valor inicial.

- *ADDangvar (iia, a0)*

  Reserva una variable de ángulo adicional.

  Dónde:

  - iia: índice de la variable de distancia.
  - a0: valor inicial.

- *MODELIZ1pt3vc (cuerpo, ip1, iv1, iv2, iv3, penaltycoef_rel)*

  Función que define la modelización en un cuerpo: 1 punto y 3 vectores.

  Dónde:

  - cuerpo: cuerpo a modelizar
  - ip1, iv1, iv2, iv3: puntos y vectores del sólido usados para modelizar. Deben ser todos ellos entidades integrables, de lo contrario se recibe un error de creación de restricciones sobre entidades no integrables.
  - penaltycoef_rel: factor que multiplica al penalizador "penaltycoef" introducido en CONSTANTES::inicializa_CONSTANTES para todas las restricciones de sólido rígido del sólido. Introducir un valor distinto de 1.d0 implica que el penalizador deja de ser un escalar y pasa a ser una matriz de penalizadores.

- *MODELIZ2pt2vc (cuerpo, ip1, ip2, iv2, iv3, penaltycoef_rel)*

  Función que define la modelización en un cuerpo: 2 puntos y 2 vectores.

  Dónde:

  - cuerpo: cuerpo a modelizar
  - ip1,ip2,iv2,iv3: puntos y vectores del sólido usados para modelizar. Deben ser todos ellos entidades integrables, de lo contrario se recibe un error de creación de restricciones sobre entidades no integrables.
  - penaltycoef_rel: factor que multiplica al penalizador "penaltycoef" introducido en CONSTANTES::inicializa_CONSTANTES para todas las restricciones de sólido rígido del sólido. Introducir un valor distinto de 1.d0 implica que el penalizador deja de ser un escalar y pasa a ser una matriz de penalizadores.

- *MODELIZ3pt1vc(cuerpo, ip1, ip2, ip3, iv3, penaltycoef_rel)*

  Función que define la modelización en un cuerpo: 3 puntos y 1 vector.

  Dónde:

  - cuerpo: cuerpo a modelizar
  - ip1,ip2,ip3,iv3: puntos y vectores del sólido usados para modelizar. Deben ser todos ellos entidades integrables, de lo contrario se recibe un error de creación de restricciones sobre entidades no integrables.
  - penaltycoef_rel: factor que multiplica al penalizador "penaltycoef" introducido en CONSTANTES::inicializa_CONSTANTES para todas las restricciones de sólido

rígido del sólido. Introducir un valor distinto de 1.d0 implica que el penalizador deja de ser un escalar y pasa a ser una matriz de penalizadores.

- *MODELIZ4pt (cuerpo, ip1, ip2, ip3, ip4, penaltycoef_rel)*

  Función que define la modelización en un cuerpo: 4 puntos.

  Dónde:

  - cuerpo: cuerpo a modelizar
  - ip1, ip2, ip3, ip4: puntos y vectores del sólido usados para modelizar. Deben ser todos ellos entidades integrables, de lo contrario se recibe un error de creación de restricciones sobre entidades no integrables.
  - penaltycoef_rel: factor que multiplica al penalizador "penaltycoef" introducido en CONSTANTES::inicializa_CONSTANTES para todas las restricciones de sólido rígido del sólido. Introducir un valor distinto de 1.d0 implica que el penalizador deja de ser un escalar y pasa a ser una matriz de penalizadores.

- *MODELIZ2pt1vc (cuerpo, ip1, ip2, iv1, penaltycoef_rel)*

  Función que define la modelización en una barra: 2 puntos y 1 vector.

  Dónde:

  - cuerpo: cuerpo a modelizar
  - ip1,ip21,iv1: puntos y vectores del sólido usados para modelizar. Deben ser todos ellos entidades integrables, de lo contrario se recibe un error de creación de restricciones sobre entidades no integrables.
  - penaltycoef_rel: factor que multiplica al penalizador "penaltycoef" introducido en CONSTANTES::inicializa_CONSTANTES para todas las restricciones de sólido rígido del sólido. Introducir un valor distinto de 1.d0 implica que el penalizador deja de ser un escalar y pasa a ser una matriz de penalizadores.

- *MatTrans (cuerpo)*

  Función para consultar la matriz de transformación de un solido3D.

  Siendo cuerpo, el cuerpo del que se va a consultar la matriz de transformación.

- *ptglob (cuerpo, idpt)*

  Función de usuario para consultar la posición de un punto no integrable de un SOLIDO3D.

  Dónde:

  - cuerpo: necesariamente debe ser SOLIDO3D.
  - idpt: identificador del punto.

- *rgglob (cuerpo)*

Función de usuario para consultar la posición del CDM de un sólido en coordenadas globales.

Siendo cuerpo, un sólido para el que se quiere consultar el CDM. Si el sólido es tipo barra el CDM se asume situado en el centro de la misma.

3.15 Módulo tipos_derivados

Módulo de definición de tipos derivados y funciones y subrutinas para manejar los tipos de datos asociados.

Los tipos de datos utilizados del módulo son:

- *Body*

Tipo derivado parámetros que controlan información sobre los cuerpos definidos.

- *Handlefnormal*

Tipo derivado parámetros que controlan lista fuerza normal según sea acción/reacción o no. Sus parámetros los asigna automáticamente evalprimitiveforces.

- idfn,id: en lista de fuerza normal de sólo reacción con suelo.
- isreaction: variable lógica que identifica a qué tipo de fuerza normal.

4 SALIDA GRÁFICA Y DETECCIÓN DE COLISIONES

La salida gráfica es la visualización interactiva que permite al usuario controlar el sistema y observar su evolución. El primer paso realizado es la descripción del diseño CAD de los objetos 3D y sus tipos de archivo de trabajo. Para esto, existen dos etapas diferentes, el modelado y la representación interactiva. El modelado implica la creación y representación fiel de las superficies por lo que la fidelidad prima sobre el rendimiento. Por otra parte, la representación interactiva usa modelos simplificados (representación por elementos sencillos como triángulos), pero permite la visualización de animaciones a 60Hz o más. De esta representación se encarga la MBSmodel, que es una biblioteca, desarrollada en el Laboratorio de Ingeniería Mecánica de la UDC, que incluye todo lo que atañe a la geometría, no solamente en lo que se refiere a la visualización sino que también a la detección de colisiones. La MBSmodel en su faceta de visualización utiliza la OpenSceneGraph, OSG, otra biblioteca de visualización a más bajo nivel que se encarga del visor con el que interactúa el usuario.

El problema de la OSG es que no puede leer materiales, sistemas de coordendas y texturas de los tipos de archivo CAD, por lo que se han convertido a un formato común compatible, el .obj. Esta conversión implica que después de tener descritos nuestros diseños en Solid Edge, los tendremos que exportar a Blender, que es una suite de código abierto multiplataforma orientada a la creación de objetos tridimensionales con la capacidad de manipular objetos, su apariencia y su orientación/posición con respecto a los ejes locales, donde ya podremos guardarlos en el formato .obj además de poder definirlos en color y textura obteniendo también un archivo .mtl, que hace referencia a estas características.

Se define .obj como un formato de definición de geometría desarrollado por Wavefront Technologies para su paquete de animación Advanced Visualizer. Es un formato simple que representa solamente la geometría. Esta geometría se representa con una malla compuesta por triángulos sencillos que representan nuestros sólidos. Se incluyen la posición de cada vértice de los triángulos que componen la malla, las normales y las caras que definen cada polígono como una lista de vértices de caras y texturas.

4.1 Modelización de la carretilla y de los entornos

Para la modelización de la carga, el software seleccionado es el Solid Edge en su versión ST8. También se ha utilizado para los últimos retoques, colores, texturas y su correspondiente conversión a .obj, el software Blender. Todo el modelizado de la máquina ha sido previamente encargado a terceros y lo han realizado en 3DStudio.

4.1.1 Diseño de los sólidos

Como se acaba de ver, el Solid Edge fue el software elegido puesto que es un programa de CAD y esto nos permite una representación más fiel de la geometría por lo que los resultados serán más exactos. Este programa también permite el diseño paramétrico de las piezas y además incluye funcionalidades que han sido útiles para la estimación de geometría de masas, como el cálculo de masas, centros de gravedad y tensores de inercia de los sólidos diseñados.

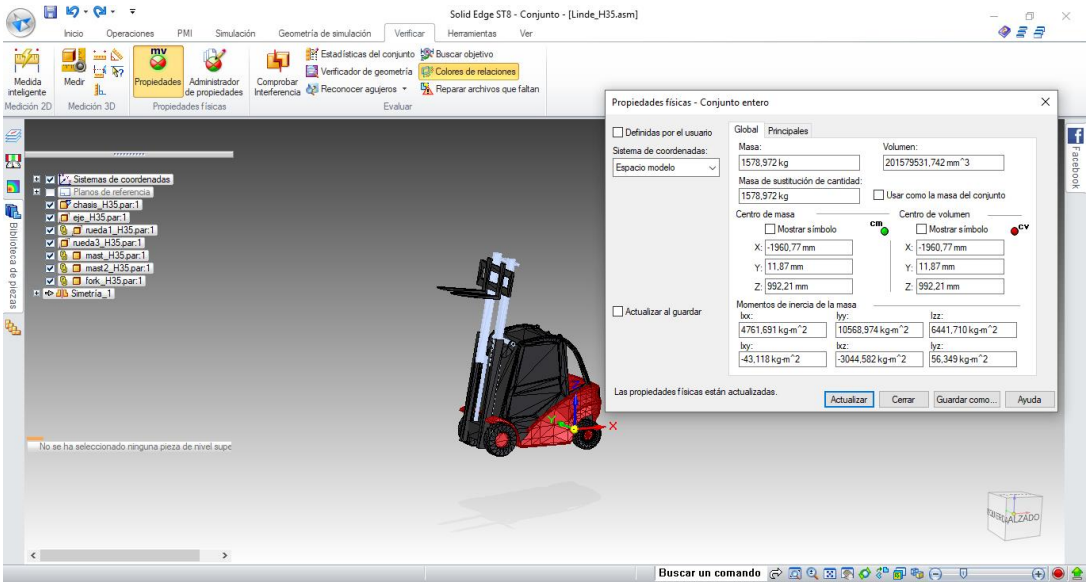

Fig 4.1 Edición y propiedades físicas con Solid Edge

Otra funcionalidad que se puede resaltar es la creación de planos de los sólidos 3D diseñados poniendo de ejemplo la Fig 2.4, vista anteriormente y que representa los diferentes sólidos que forman el modelo de carretilla.

Este software también permite crear ensamblajes formando un prototipo modificable, que permita comprobar que tiene la funcionalidad deseada para el simulador.

En lo referente al formato, de Solid Edge se puede exportar los archivos en formato .stl, que es un formato de archivo informático de diseño asistido por computadora (CAD) que define geometría de objetos en malla 3D, excluyendo información como color, texturas o propiedades físicas que sí incluyen otros formatos CAD. Este formato se puede importar directamente a Blender, mencionado anteriormente, en el cual se puede dar color y textura a los sólidos y exportarlos directamente en formato .obj.

Otra alternativa a Blender para poner texturas y manipular el sólido, por ejemplo, añadir fácilmente esferas en los puntos correspondientes a la modelización como se ha visto durante todo el apartado 2.4.2 donde se definen los sólidos modelizados, es el 3D Builder.

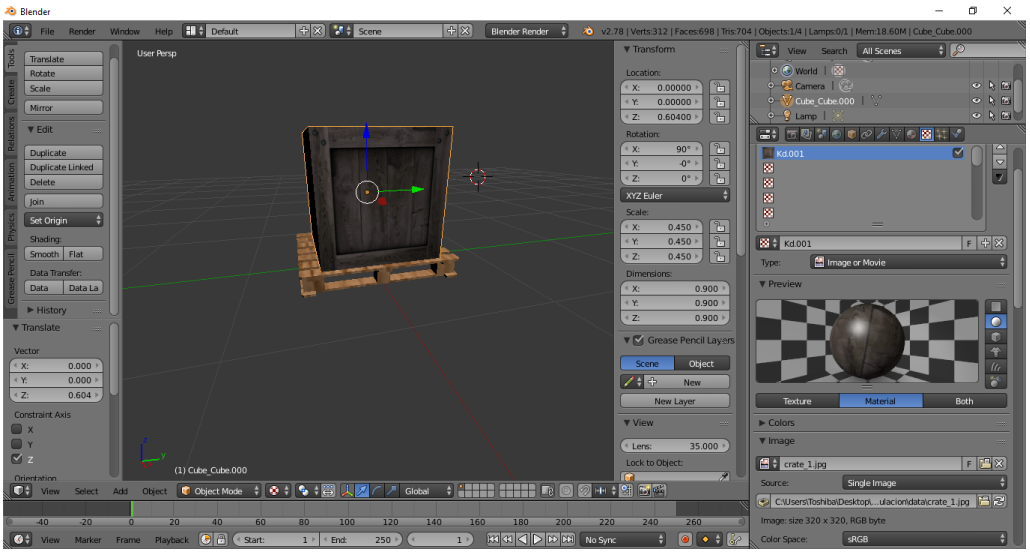

Fig 4.2 Edición con blender

Descrito ya el proceso por el cual se crearon los diversos sólidos que componen el modelo, se puede decir que los sólidos que componen la carretilla se han modelado con las dimensiones proporcionadas por el fabricante en el catálogo de la máquina.

A partir de los datos del catálogo del fabricante, disponibles en el anexo II, también se han podido obtener algunos parámetros esenciales, porque influyen en su estabilidad y comportamiento, para la creación del modelo de carretilla, tales como son la masa de la carretilla completa, o la posición longitudinal y vertical de su centro de masas. La distribución de masa entre distintos sólidos que forman la carretilla es desconocida, pero al conocer la masa de la carretilla completa, las hemos estimado. Otros parámetros tales como la posición vertical del centro de masas o los momentos de inercia, no son proporcionados por el fabricante y han debido de ser estimados a partir de modelos de CAD. Para ello, se ha estimado la masa de cada sólido y con ella y la geometría (que da la distribución de masas), se puede estimar el tensor de inercia, $\mathbf{I}_g$, correspondiente a cada sólido.

A continuación se muestra el modelo CAD de cada sólido y sus propiedades físicas estimadas.

- **Chasis y cabina**

En cuanto al centro de masas del chasis, se supone la masa simétrica respecto del plano longitudinal de la máquina y por lo tanto el centro de masas centrado lateralmente. La ubicación longitudinal no se conoce, pero sí que se conoce la ubicación longitudinal del centro de masas de la máquina completa, estimado a partir de las cargas de las ruedas delanteras y traseras en vacío.

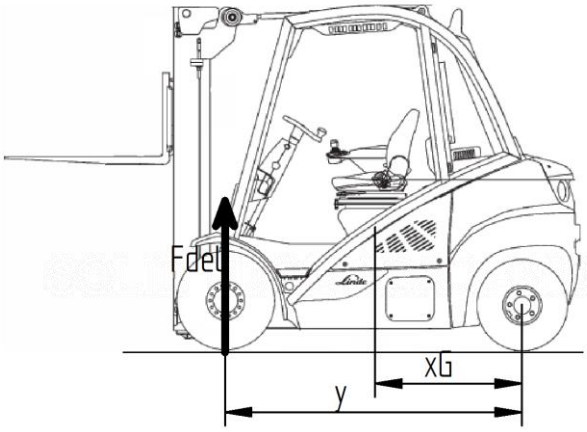

Fig 4.3 Posición longitudinal del centro de masas de la carretilla

$$x_G = \frac{F_{del}\, y}{\text{tara}} \qquad (4.1)$$

El valor de x_G de la máquina completa, proporciona una relación que las coordenadas longitudinales de los centros de masas de los distintos cuerpos deben cumplir.

En cuanto a la altura del centro de masas y al tensor de inercia, no se disponen de datos del fabricante y el mismo ha tenido que ser estimado a partir de los modelos de CAD desarrollados que vemos a continuación.

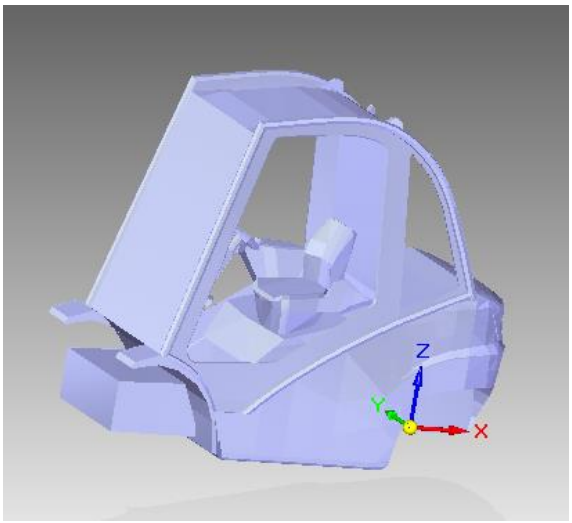

Fig 4.4 Modelo en SolidEdge del chasis y cabina

Sus propiedades físicas estimadas son:

$$m_{chasisycabina} = 3086,66\ kg \tag{4.2}$$

$$\mathbf{r}_{g\ chasisycabina} = (-0'1885\ ,\ 0\ ,\ 0'5)\ m \tag{4.3}$$

$$\mathbf{I}_{g\ chasisycabina} = \begin{pmatrix} 615,568 & 5,345 & -28,019 \\ 5,345 & 918,414 & 1,341 \\ -28,019 & 1,341 & 617,661 \end{pmatrix} \mathrm{kg \cdot m^2} \tag{4.4}$$

- **<u>Eje basculante</u>**

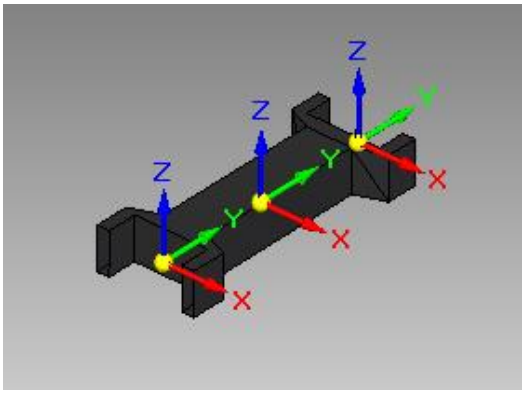

Fig 4.5 Modelo en SolidEdge del eje basculante

Sus propiedades físicas estimadas son:

$$m_{eje} = 50\ kg \tag{4.5}$$

$$\mathbf{r}_{g\ eje} = (0\ ,\ 0\ ,\ 0)\ m \tag{4.6}$$

$$\mathbf{I}_{g\ eje} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \mathrm{kg \cdot m^2} \tag{4.7}$$

- **Ruedas traseras**

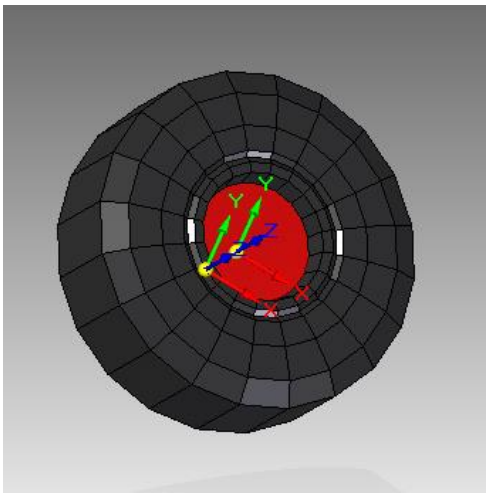

Fig 4.6 Modelo en SolidEdge de las ruedas traseras

Sus propiedades físicas estimadas son:

$$m_{ruedastraseras} = 53{,}11\ kg \tag{4.8}$$

$$\mathbf{r}_{g\ ruedastraseras} = (0\ ,\ 0\ ,\ 0)\ m \tag{4.9}$$

$$\mathbf{I}_{g\ ruedastraseras} = \begin{pmatrix} 1{,}271 & 0 & 0 \\ 0 & 1{,}271 & 0 \\ 0 & 0 & 2{,}260 \end{pmatrix} \mathrm{kg \cdot m^2} \tag{4.10}$$

- **Ruedas delanteras**

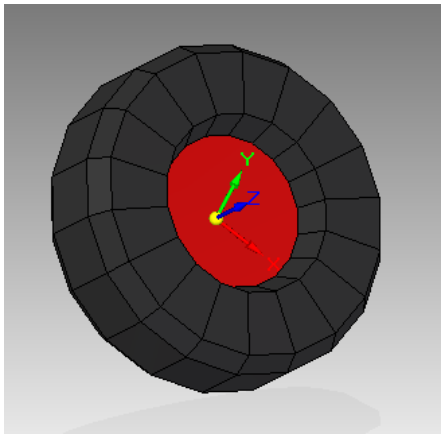

Fig 4.7 Modelo en SolidEdge de las ruedas delanteras

Sus propiedades físicas estimadas son:

$$m_{ruedasdelanteras} = 82{,}56\ kg \tag{4.11}$$

$$\mathbf{r}_{g\ ruedasdelanteras} = (0\ ,\ 0\ ,\ 0)\ m \tag{4.12}$$

$$\mathbf{I}_{g\ ruedasdelanteras} = \begin{pmatrix} 2{,}829 & 0 & 0 \\ 0 & 2{,}829 & 0 \\ 0 & 0 & 4{,}965 \end{pmatrix} \mathrm{kg \cdot m^2} \tag{4.13}$$

- **Mástil 1**

La masa del mástil1 también ha sido proporcionada por REYCA, servicio oficial de atención al cliente de Linde, siendo:

$$m_{mástil1} = 250\ kg \tag{4.14}$$

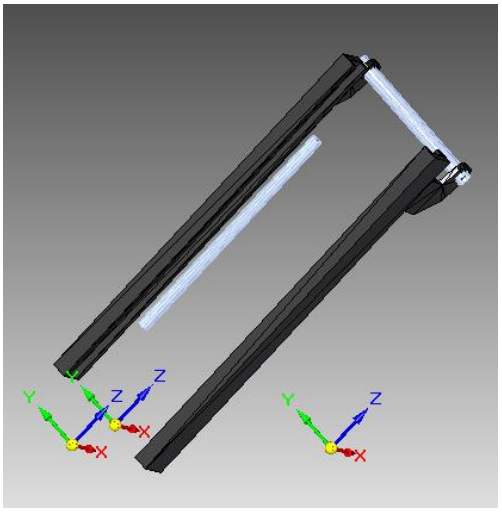

Fig 4.8 Modelo en SolidEdge del mástil 1

Sus propiedades físicas estimadas son:

$$\mathbf{r}_{g\ mástil1} = (-2'2674\ ,\ 0\ ,\ 0'9946)\ m \tag{4.15}$$

$$\mathbf{I}_{g\ mástil1} = \begin{pmatrix} 126,392 & -0,054 & -1,313 \\ -0,054 & 109,147 & -0,196 \\ -1,313 & -0,196 & 18,615 \end{pmatrix} \mathrm{kg \cdot m^2} \tag{4.16}$$

- **Mástil 2**

La masa del mástil2 ha sido proporcionada por REYCA, servicio oficial de atención al cliente de Linde, siendo:

$$m_{mástil2} = 250\ kg \tag{4.17}$$

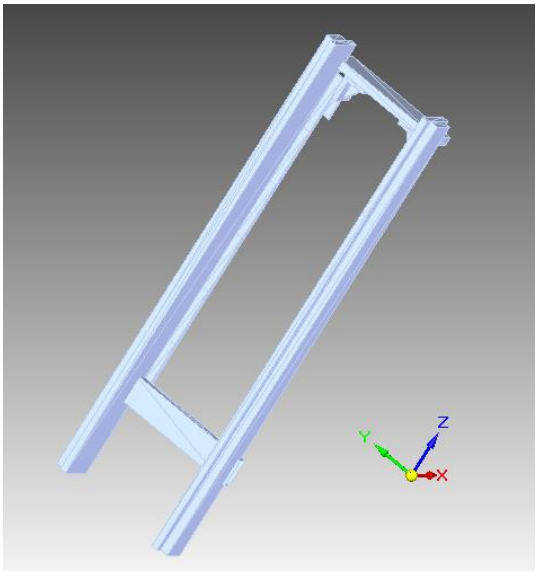

Fig 4.9 Modelo en SolidEdge del mástil 2

Sus propiedades físicas estimadas son:

$$\mathbf{r}_{g\ mástil2} = (-2'2674\ ,\ 0\ ,\ 0'9946)\ m \tag{4.18}$$

$$\mathbf{I}_{g\ mástil2} = \begin{pmatrix} 126,392 & -0,054 & -1,313 \\ -0,054 & 109,147 & -0,197 \\ -1,313 & -0,197 & 18,615 \end{pmatrix} \mathrm{kg \cdot m^2} \tag{4.19}$$

- **Uñas y útil**

La masa de las uñas y el útil también ha sido proporcionada por REYCA, servicio oficial de atención al cliente de Linde, siendo:

$$m_{uñasyútil} = 772\ kg \tag{4.20}$$

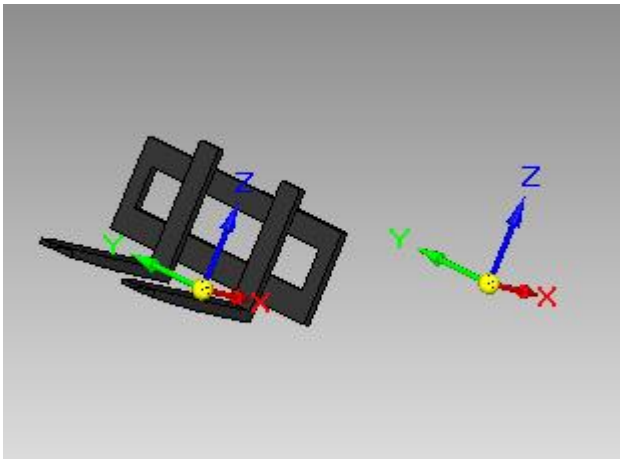

Fig 4.10 Modelo en SolidEdge de las uñas y útil

Sus propiedades físicas estimadas son:

$$\mathbf{r}_{g\ uñasyútil} = (-2'5753\ ,\ 0\ ,\ -0'1438)\ m \tag{4.21}$$

$$\mathbf{I}_{g\ uñasyútil} = \begin{pmatrix} 118,681 & 0 & 27,739 \\ 0 & 99,400 & 0 \\ 27,739 & 0 & 147,933 \end{pmatrix} \mathrm{kg \cdot m^2} \tag{4.22}$$

- **Conductor**

Para el modelo de conductor no se ha diseñado un archivo CAD, pero ya se ha visto representado como un archivo .obj en la Fig 2.32 Sólido 8 de la carretilla. Conductor

Sus propiedades físicas estimadas son:

$$m_{conductor} = 85\ kg \tag{4.23}$$

$$\mathbf{r}_{g\ conductor} = (0\ '2955,\ 0\ ,\ 0'7075)\ m \tag{4.24}$$

$$\mathbf{I}_{g\ conductor} = \begin{pmatrix} 7,974 & 0,047 & 2,642 \\ 0,047 & 9,157 & -0,059 \\ 2,642 & -0,059 & 3,595 \end{pmatrix} \mathrm{kg \cdot m^2} \tag{4.25}$$

- **Combustible**

Para el modelo de combustible, no se ha diseñado un archivo CAD.

Sus propiedades físicas estimadas son:

$$m_{combustible} = 37,44\ kg \tag{4.26}$$

$$\mathbf{r}_{g\ combustible} = (0\ ,\ 0\ ,\ 0)\ m \tag{4.27}$$

$$\mathbf{I}_{g\ combustible} = \begin{pmatrix} 0,789 & 0 & 0 \\ 0 & 0,789 & 0 \\ 0 & 0 & 0,789 \end{pmatrix} \mathrm{kg \cdot m^2} \tag{4.28}$$

- **Cilindros hidráulicos**

Además, también previamente se ha encargado a terceros el diseño de los cilindros hidráulicos de la inclinación del mástil en los cuales no se han tenido en cuenta sus propiedades físicas puesto que se han considerado despreciables con respecto a los órdenes de magnitud del resto de elementos.

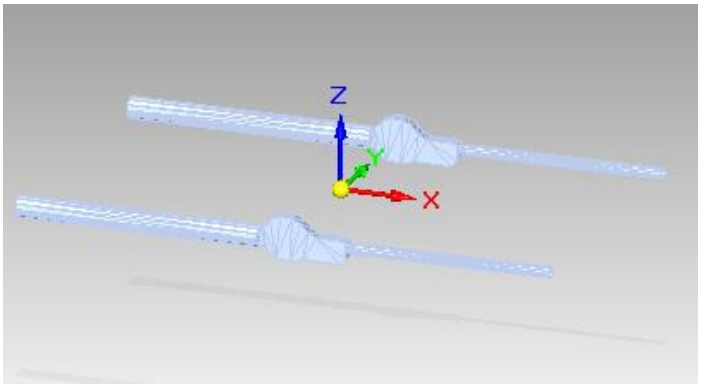

Fig 4.11 Modelo en SolidEdge de los cilindros hidráulicos

- **Pallet**

En el caso del pallet no se ha estimado la masa si no que se ha obtenido directamente del Solid Egde indicando como material, un tipo de madera cuya densidad es $\rho = 600,00\ kg/m^3$.

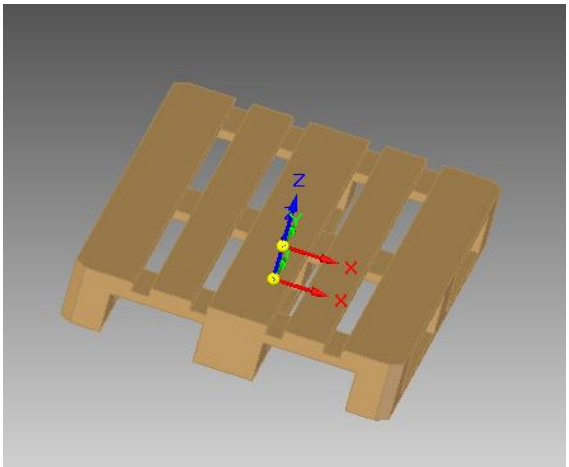

Fig 4.12 Modelo Solid Edge del pallet

Sus propiedades físicas obtenidas directamente en el CAD son:

$$m_{pallet} = 27,296\ kg \tag{4.29}$$

$$\mathbf{r}_{g\ pallet} = (0\ ,\ 0\ ,\ 0'08653)\ m \tag{4.30}$$

$$\mathbf{I}_{g\ \text{pallet}} = \begin{pmatrix} 4,079 & 0 & 0 \\ 0 & 1,799 & 0 \\ 0 & 0 & 5,749 \end{pmatrix} \text{kg}\cdot\text{m}^2 \tag{4.31}$$

- **Virola**

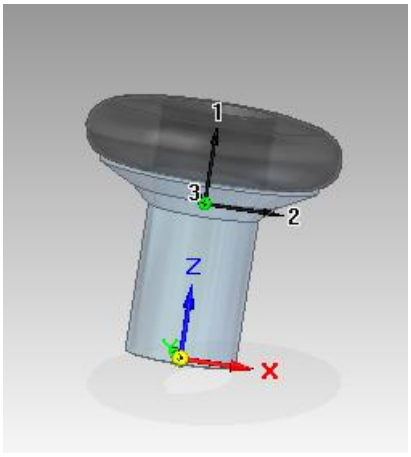

Fig 4.13 Modelo en SolidEdge de la virola

Sus propiedades físicas estimadas son:

$$m_{virola} = 239,40\ kg \tag{4.32}$$

$$\mathbf{r}_{g\ virola} = (0\ ,\ 0\ ,\ 0'75)\ m \tag{4.33}$$

$$\mathbf{I}_{g\ virola} = \begin{pmatrix} 11,077 & 0 & 0 \\ 0 & 11,077 & 0 \\ 0 & 0 & 12,149 \end{pmatrix} \text{kg}\cdot\text{m}^2 \tag{4.34}$$

- **Pallet cargado**

Se ha introducido un pallet de 1200x1000mm con una carga cuadrada a la que se le puede variar la masa para ver el comportamiento de la carretilla en diferentes situaciones. En principio, se propone una masa de la carga de 1500 kg.

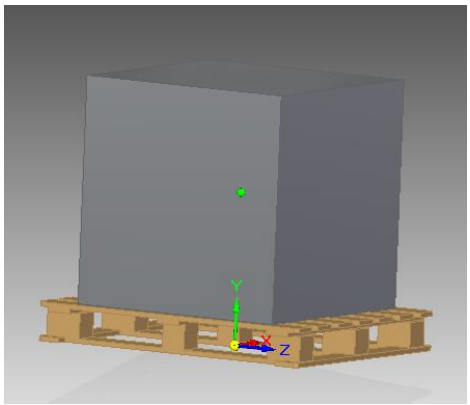

Fig 4.14 Modelo en SolidEdge del pallet cargado

Sus propiedades físicas estimadas son:

$$m_{palletcargado} = m_{pallet} + m_{carga} = 27,296 + 1500 = 1527,296\ kg \quad (4.35)$$

$$\mathbf{r}_{g\ palletcargado} = (0\ ,\ 0\ ,\ 0'599)\ m \quad (4.36)$$

$$\mathbf{I}_{g\ palletcargado} = \begin{pmatrix} 239,775 & 0 & 0 \\ 0 & 234,846 & 0 \\ 0 & 0 & 215,775 \end{pmatrix} \mathrm{kg \cdot m^2} \quad (4.37)$$

4.1.2 Diseño del escenario

Para el diseño del escenario de la carretilla se ha utilizado un escenario previamente creado con Google SketchUp, que es un software desarrollado por Google que con unas herramientas sencillas permite dibujar en 3D y dar texturas a las caras partiendo de una colección que viene instalada por defecto en el paquete, por ello resulta muy cómodo dibujar los escenarios.

Además dispone de una amplia biblioteca de objetos realizados por la comunidad de usuarios Google que se pueden importar.

4.1.3 Objetos del escenario

Una vez diseñados los objetos 3D mediante poliedros, se incorporan texturas para hacer su visualización más realista y posteriormente son exportados al formato .obj. Para este último paso se ha utilizado Blender.

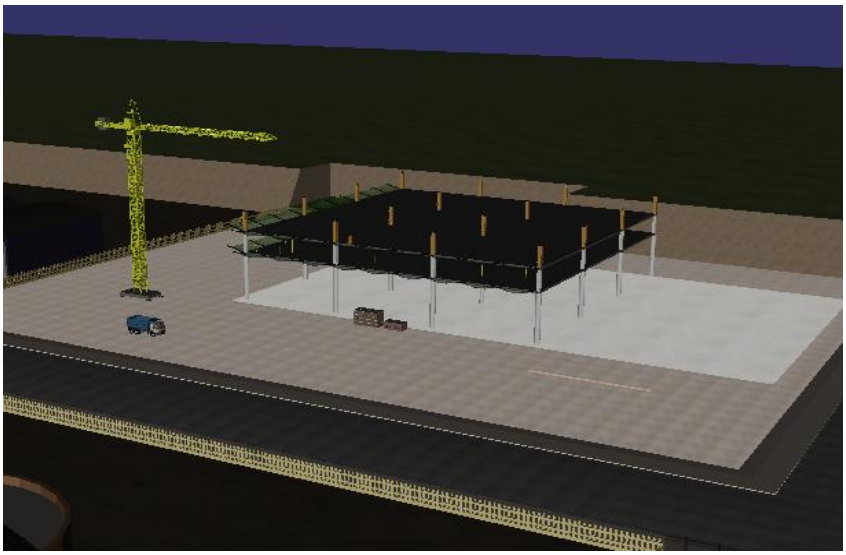

Fig 4.15 Diseño final del escenario

4.2 Salida gráfica en tiempo real

La MBSmodel es una biblioteca de funciones que se encarga de la detección de colisiones para la dinámica, de la representación de los escenarios, de la representación de todas las piezas móviles del sistema y de los gráficos del simulador.

Para ello la MBSmodel utiliza la OpenSceneGraph (OSG), que es otra biblioteca de funciones que permite a las aplicaciones 3D manejar la geometría y el estado de los renders de bajo nivel sin necesidad de lidiar directamente con el hadware de representación (tarjetas

gráficas) y los comandos de bajo nivel necesarios para generar las representaciones. La aplicación 3D maneja los datos de los objetos 3D que tratados con las bibliotecas de la OSG dan lugar a instrucciones para APIs de bajo nivel (la OSG sólo trabaja con OpenGL) que se encargan de dar las instrucciones a la tarjeta gráfica para el correcto renderizado de los objetos.

Con la aplicación 3D se define un escenario para la OSG, ésta traduce el escenario a OpenGL y éste último hace que la tarjeta gráfica lo muestre en pantalla.

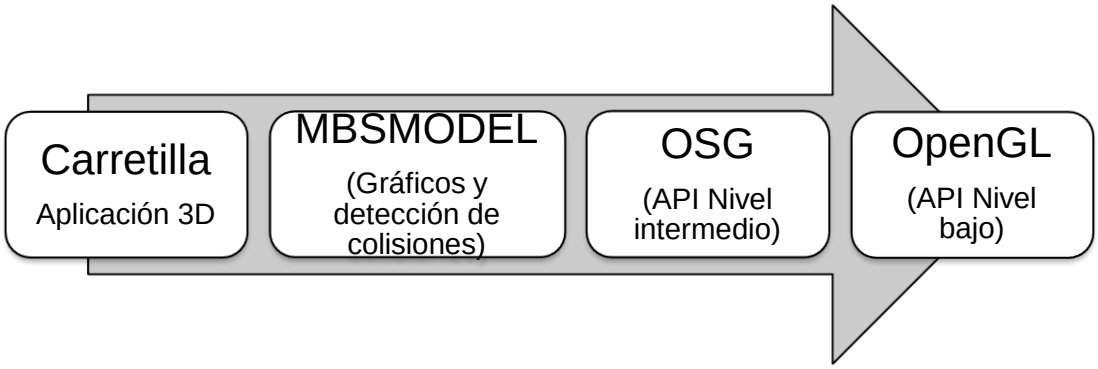

Fig 4.16 Proceso de tratamiento de imágenes 3D

Algunas de las características de la OSG son:

- Organización espacial: Estructurar el escenario en árbol se presta a una intuitiva organización espacial.
- Culling: Reduce la sobrecarga del sistema no procesando la geometría que no aparece en la imagen final renderizada.
- I/O de archivos, permite leer y guardar archivos en disco, que una vez cargados en la memoria, la estructura de datos interna de la OSG permite a la aplicación una fácil manipulación dinámica de los datos 3D.
- Alto nivel adicional, las bibliotecas de la OSG permiten funciones que otras APIs de bajo nivel no soportan, por ejemplo sombras o partículas, optimización de renderizado, soporte para entrada/ salida de ficheros del modelo y un acceso multiplataforma a los dispositivos de entrada y renderizadores de superficies.

El escenario en la OSG es un árbol de datos jerárquico que organiza los datos espaciales del mismo para mejorar la eficiencia del proceso. Está formado por nodos, encabezado por un nodo de nivel superior, el nodo raíz, por debajo de éste, grupos de nodos organizan la geometría y su apariencia. Por último los nodos de menor nivel, nodos hoja, contienen la geometría actual de los objetos de la escena.

El árbol del escenario es:

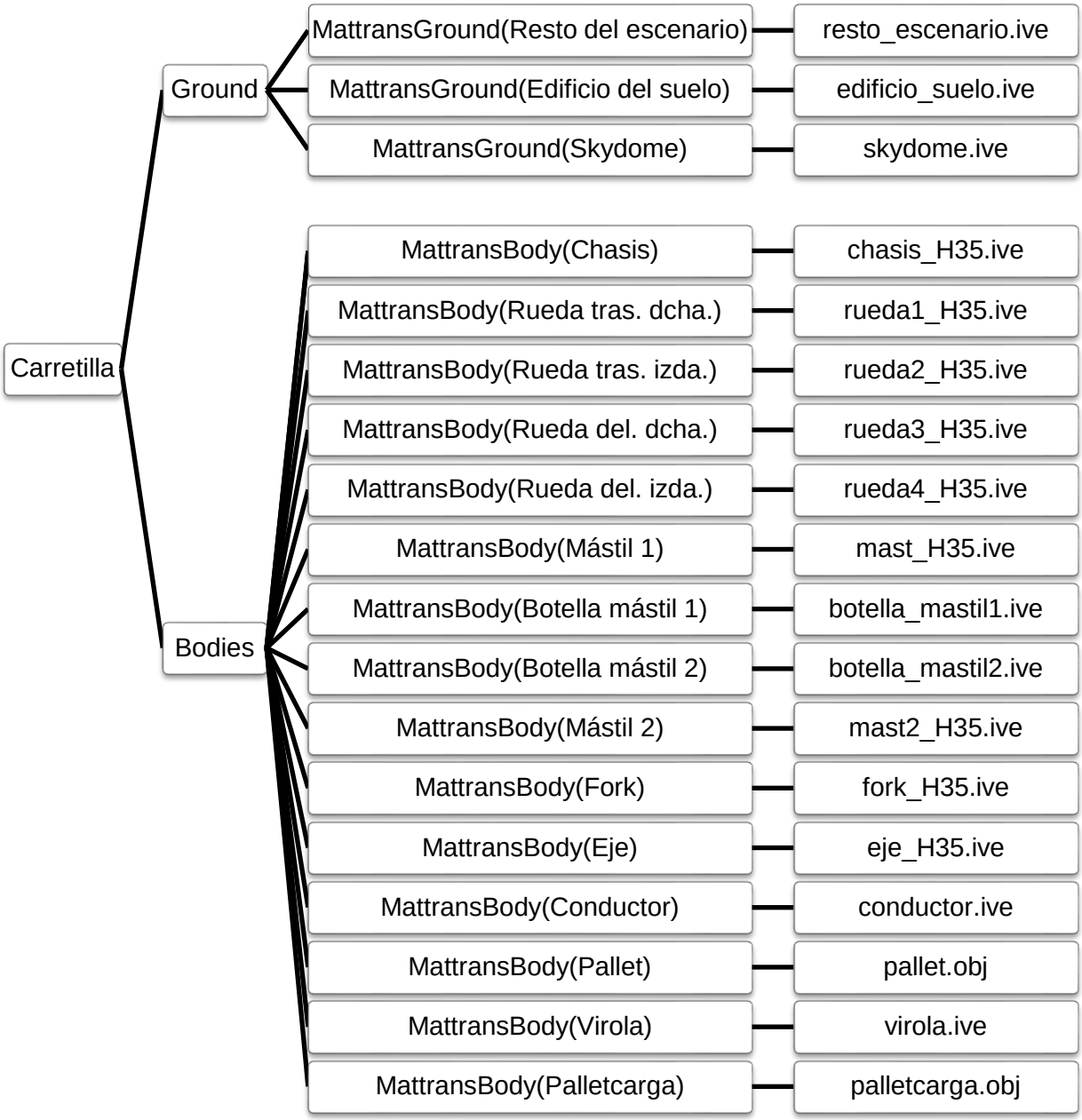

Fig 4.17 Estructura del modelo gráfico-físico de la MBSmodel

El nodo raíz tiene dos grupos, el grupo Ground compuesto por las mallas fijas, correspondiente al entorno y el grupo Bodies compuesto por mallas móviles, correspondiente al resto de objetos. Después de los grupos de mallas fijas y móviles están las matrices de transformación aplicadas para situar a cada uno de los objetos y por último están los archivos .ive y .obj, que son los que tienen la información geométrica del sólido para su renderizado.

$$\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \tag{4.38}$$

Las matrices de transformación son matrices homogéneas de 4x4 que están compuestas por $\mathbf{R}$, que es la matriz de rotación de 3x3, $\mathbf{t}$ que es el vector de translación de

3x1 y la última fila que está compuesta de ceros, menos la componente homogénea que toma el valor 1.

4.2.1 Creación del modelo

Para crear el modelo de la carretilla y facilitar la introducción de la geometría, se dispone de la MBSmodel, que es una biblioteca de funciones desarrollada en el Laboratorio de Ingeniería Mecánica de la Universidad de A Coruña. Ésta se encarga de la comunicación entre la aplicación gráfica o programa principal y la OSG, que como se ha visto en el apartado anterior, controla al subsistema OpenGL (Open Graphics Library), que es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D.

OpenGL consiste en más de 250 funciones diferentes que pueden usarse para dibujar escenas tridimensionales complejas a partir de primitivas geométricas simples, tales como puntos, líneas y triángulos, poder mostrar los gráficos del simulador. Como se ha visto, la MBSmodel usa la OSG, es decir, es de un nivel más alto. Esto provoca que el usuario que controla el simulador, solo maneje la OSG en cuanto lo que se refiere al visor y a las cámaras, puesto que es la MBSmodel la encargada de crear, actualizar y manipular el modelo gráfico.

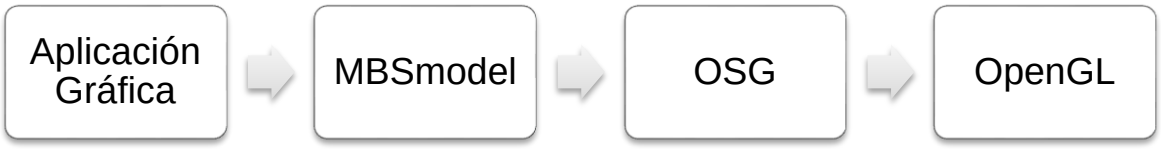

Fig 4.18 Jerarquía de control gráfico

Lo primero es crear un modelo de clase MBSmodel para guardar el modelo. Los datos que almacenará esta clase son:

- Root: grupo raíz del que descienden todos los demás, y que sirve como referencia para OSG como punto de entrada a la geometría.
- GroupGround: grupo de mallas fijas.
- GroupBodies: grupo de mallas móviles.
- MtGround: nodos para matrices de transformación de mallas fijas
- MtBodies: nodos para matrices de transformación de mallas móviles.
- numMattransbodies: número de sólidos móviles.
- numMattransground: número de sólidos fijos.

Y sus funciones son:

- AddGround: añade mallas de Sólidos fijos.
- AddBody: añade mallas de Sólidos móviles.
- MattransBodies: añade matrices de transformación de sólidos móviles.
- MattransGround: añade matrices de transformación de sólidos fijos.
- AddNode: añade nodos gráficos auxiliares (geometría auxiliar o complementaria).

El proceso en la ejecución de la aplicación es:

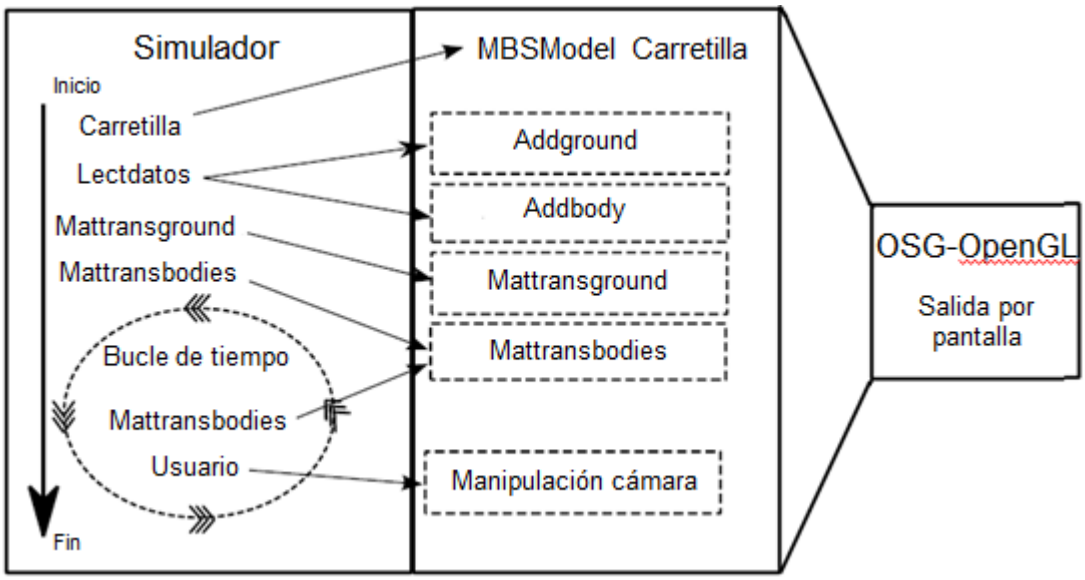

Fig 4.19 Diagrama de llamadas de la parte gráfica del programa

En el inicio del programa, se crea un objeto encargado de llevar la estructura de árbol con una raíz y dos grupos, Ground y Body. Para ello de emplea la clase MBSmodelOPCC que es un modelo gráfico de la MBSmodel con detección de colisiones. Esta clase permite generalizar el cómputo de los parámetros necesarios para el cálculo de los modelos de fuerzas de la dinámica a la MBSLIM puesto que la detección de colisiones puede producirse por múltiples tipologías de contacto y así, la MBSmodel devolverá a la MBSLIM los valores del punto de contacto, la interpenetración o deformación local y la normal al contacto sea cual sea el tipo de éste.

En la siguiente etapa se cargan las mallas de los sólidos fijos y móviles en Carretilla de la siguiente forma: primero la MBSmodel crea un nodo matriz de transformación y un nodo auxiliar hijo del anterior, que contiene la geometría, para cada sólido como se ha visto en la Fig 4.17 . Los nodos estarán en Ground si el sólido es fijo y en Bodies si es móvil. Después carga los datos de la geometría del sólido en el nodo auxiliar, quedando vacío el nodo de matriz de transformación que se inicializa por defecto con la matriz identidad, de tal forma que la posición y orientación por defecto de cada sólido cargado por la MBSmodel coincide con su posición y orientación en ejes locales.

A continuación de la carga del modelo gráfico, el programa principal del simulador (Fig 4.20) llama a la rutina de creación del modelo numérico de MBSLIM en Fortran y posteriormente a las rutinas de resolución de los problemas de posición y velocidad inicial de la MBSLIM. Una vez resueltos estos problemas, la configuración inicial de cada sólido queda correctamente determinada.

Debido a que la posición y orientación por defecto de los objetos gráficos antes citada, no coincide en general con la configuración inicial calculada por el modelo numérico de MBSLIM, el simulador de carretilla hace una primera llamada a la rutina de cálculo de las matrices de transformación de los sólidos que devuelve dichas matrices a partir de la información numérica de la MBSLIM.

En este instante se inicia el visor, que se encarga de representar el escenario con todos los objetos en su posición inicial.

Por último, comienza el bucle de tiempo en el que se intercala el cómputo de la dinámica y la visualización y adquisición de datos de los mandos. La estrategia empleada es la de simular la dinámica hasta el instante actual, y sólo después se representa. De esta manera, puede haber varios pasos de cómputo por cada paso de visualización. En cada instante, la MBSmodel llama al modelo numérico de MBSLIM para obtener las matrices de transformación actualizadas y con Mattransbodies, perteneciente a la MBSmodel se pasan a

la MBSmodel las nuevas matrices de transformación calculadas por la MBSLIM y la OSG puede enviar las instrucciones al OpenGL para el renderizado de la escena. Además desde el bucle de tiempo se llama a la adquisición de datos de los mandos cuyos inputs son pasados también al modelo de la MBSLIM que calcula la dinámica, tal y como se describe en el capítulo siguiente.

Además de las llamadas del programa principal y de la MBSmodel a la MBSLIM, en cada iteración de cada instante de tiempo de la dinámica, hay también llamadas en el sentido inverso. La MBSLIM llama a la MBSmodel cada vez que necesita evaluar una fuerza de contacto ya que la MBSmodel es la única conocedora de la geometría de los sólidos y la encargada de llevar a cabo la detección de contactos y devolver toda la información geométrica que el modelo de fuerzas de la MBSLIM necesita para calcularlas.

4.2.2 Detección de colisiones

Se puede dividir la detección de colisiones en dos pasos: la detección de colisiones propiamente dicha efectuada por la MBSmodel y la evaluación del modelo de fuerza que efectúa la MBSLIM. Se distingue entre los contactos de los distintos tipos de carga con el suelo (tipo esfera-plano) y entre los contactos de los distintos tipos de carga con las uñas de la carretilla (tipo malla-malla).

- **Contacto tipo esfera-plano**

Este tipo de contacto es el que se implementa en las colisiones de los distintos tipos de carga con el suelo. Para obtener este tipo de contacto se han definido, en coordenadas locales del sólido, unas esferas de un radio determinado situadas en puntos estratégicos del modelo. Estas esferas marcan los puntos de colisión del sólido con el suelo, por lo que si cualquiera de las esferas definidas interfiere con cualquier límite del terreno, aparece una fuerza de contacto que impide la penetración en el mismo.

Así mismo, los puntos donde se han establecido los centros de las esferas con los que se definen las cargas, que se pueden ver en el Anexo I, en el fichero lectdatos.f90. El radio establecido para las esferas dependerá de la geometría, en el caso que nos atañe, a las esferas que se establecen en ambos pallets se les asigna un radio de 50mm para que estas sean tangentes a los bordes exteriores de los apoyos, como se puede apreciar en las siguientes imágenes.

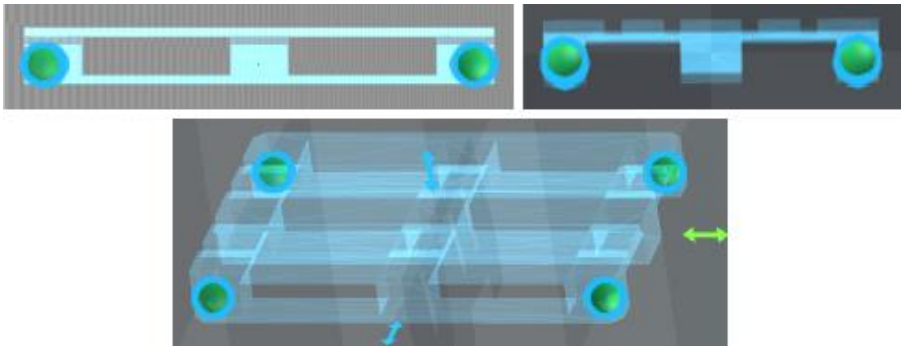

Fig 4.20 Situación de las esferas de contacto en el pallet vacío

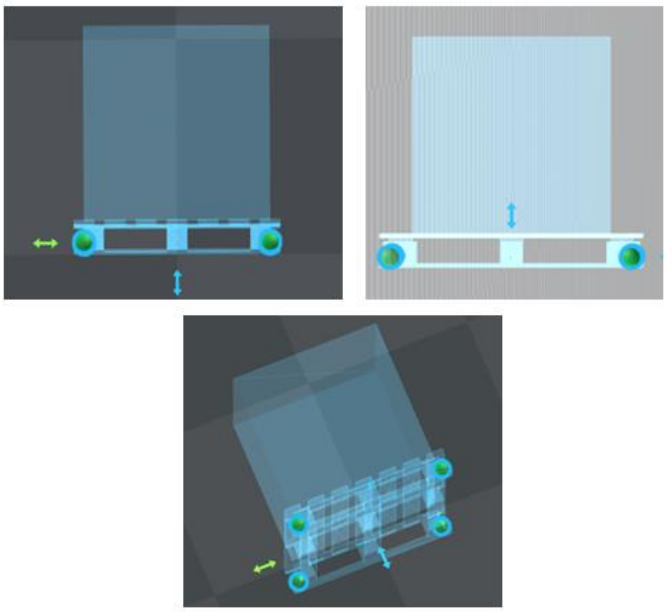

Fig 4.21 Situación de las esferas de contacto en el pallet cargado

En la virola, el radio de las esferas que se han definido también es de 50 mm aunque la pared de la misma es de menor espesor. Esto sucede puesto que si el diámetro de las esferas fuese del mismo espesor que la pared, podría dar problemas de penetración en el suelo y poner un diámetro mayor, no nos influirá si las esferas son tangentes al exterior de la pared. Se sitúan 4 esferas formando un cuadrado como se ve en la siguiente imagen.

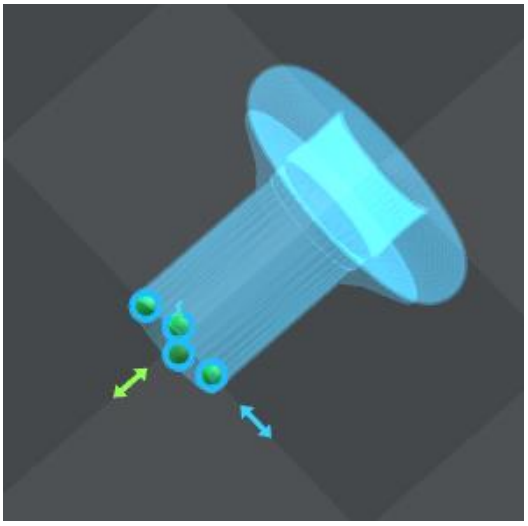

Fig 4.22 Situación de las esferas de contacto en la virola

- **Contacto tipo malla-malla**

Es el tipo de contacto es el que se implementa en las colisiones entre dos cuerpos. Este es el tipo de contacto implementado entre las uñas de la horquilla y los distintos tipos de carga. Además para obtener resultados de posibles vuelcos de la carretilla, se han implementado también este tipo de colisiones entre la horquilla y el chasis contra el suelo.

4.2.3 Visor

El visor es el encargado de representar en cada en instante el escenario del simulador, permite ajustar las luces con los vectores Ambient (Ambiente), Diffuse (Difuso), Specular (Especular) y Position (Posición).

El vector Position define la posición del foco de luz, los otros tres Ambient, Diffuse y Specular afectan a la forma de reflejarse la luz en los objetos y la cámara.

4.2.4 Cámara

La cámara define la proyección de la perspectiva, y calcula qué objetos entran en la escena, de tal manera que sabe qué objetos son visibles, reduciendo la carga de trabajo del hardware de visualización (tarjeta gráfica).

Las cámaras de la OSG se definen mediante tres vectores contenidos en el campo makeLookAt de un objeto de la clase osg::Matrixd: el primero define la posición de la cámara, el segundo, la posición del punto donde está enfocada y el tercero, la dirección de la vertical.

Se ha implementado una única cámara de clase trackball del tipo osgGA::TrackballManipulator. Esta inicializa la cámara en un punto que se puede desplazar con los movimientos del ratón. Siendo sus controles:

- Click izquierdo + desplazamiento: zoom.
- Click derecho + desplazamiento: desplazamiento alrededor del centro definido en el campo makeLookAt.

5 LECTURA DE INPUTS DE USUARIO

La lectura de inputs de usuario se realiza mediante Simple DirectMedia Layer (SDL), que es un conjunto de bibliotecas desarrolladas en lenguaje C que proporcionan un acceso sencillo al audio, teclado, ratón, joysticks y a la tarjeta gráfica vía OpenGL.

De ahí, se ha utilizado la parte de la SDL correspondiente al joystick, con la que se ha generado el control de la carretilla elevadora en tiempo real. Como se verá en el siguiente apartado 6, para definir los movimientos de la carretilla, los mandos utilizados son, un joystick y un volante con sus pedales, cuyos movimientos se corresponden con los de la carretilla real que ha sido simulada.

En el simulador, se tienen una serie de funciones programadas para los distintos tipos de acciones que se pueden hacer con los mandos, ver Anexo I, fichero MandosSDL:

1. Accionar un eje en sentido positivo o negativo. La función controlEjes toma el valor del estado del eje en cada instante, la SDL tiene un rango de valores que va desde -215 hasta 215, se quiere que el valor permanezca entre -1 y 1, por esto, se divide el valor devuelto por la función SDL_JoystickGetAxis(j, eje) entre 32768.

2. Pulsar un botón para alguna función específica. La función ControlBotón recoge el estado del botón, si está pulsado SDL_JoystickGetButton (j, n) toma el valor 1, si no está pulsado toma el valor 0.

3. Leer una pareja de botones. Cuando se tiene una pareja de botones, uno acciona el sentido positivo del GDL y otro el negativo. La función ControlBotones recoge el estado de los botones, llamando l al botón positivo y h al negativo, si está pulsado l, la función devuelve un 1 y si el que está pulsado es h devuelve un -1.

Las acciones de los mandos son recogidas en cada instante en un vector $\mathbf{h}$ de dimensión 2, que representa el movimiento de cada eje del joystick correspondiendo con la inclinación del mástil y la elevación o descenso del portahorquillas respectivamente, y dos variables “d” y “t” que representan el movimiento de dirección del volante y el movimiento de la transmisión en los pedales respectivamente.

Tanto el vector $\mathbf{h}$, como las variables “d” y “t” que han sido nombradas anteriormente, se llaman desde el main.cpp en la función lectjoysticks. Desde el main.cpp se envían a constantes_modelo.f90 a una subrutina llamada joysticks que es de donde las recoge la función guiada.f90, que es donde se manipulan para representar el movimiento en el simulador (véase Anexo I).

En esta subrutina guiada.f90, los valores del vector $\mathbf{h}$ representados por el vector ACTUAD, se interpretan como coeficientes que multiplican a la aceleración máxima de cada actuador, es decir, la aceleración de cada actuador en cada instante se calcula como $h_i \times a_{max\_actuador\ i}$, y partiendo de los datos del paso de tiempo anterior, se calcula su posición y velocidad integrando con el método de numérico Euler Adelante. Además, el bucle que realiza estos cálculos, tiene implementadas unas limitaciones de estos actuadores tanto por fuerza máxima alcanzada por las botellas hidráulicas, por velocidad máxima alcanzada y por límites de carrera de los actuadores superados.

Los valores de la variable “t” representados por ACTUAD_TRANSM, se interpretan, al igual que hemos visto con los del vector $\mathbf{h}$, como un coeficiente que multiplica a la aceleración máxima del actuador. Las únicas diferencias con los cálculos del movimiento representado por el vector $\mathbf{h}$, son:

- La inclusión de una deceleración implementada si la variable ACTUAD_TRANSM en el instante actual, es menor que esta misma variable en el instante inmediatamente anterior, ya que como se ha comentado en el apartado 1.3, en el modelo de carretilla elevadora elegido, el freno de la misma se efectúa según vamos soltando el pedal de aceleración ya sea en el sentido de la marcha o en el opuesto. Este fenómeno se explica como un modelo de fuerza en el apartado 2.4.6.2.
- La eliminación de la limitación por límites de carrera superado ya que en este caso no tienen sentido.

Sin embargo, los valores de la variable “d” representados por ACTUAD_DIREC, son interpretados como un coeficiente que se multiplica, a diferencia de los anteriores, por el ángulo máximo de giro, para establecer el ángulo de giro de las ruedas directrices.

Los giros de ambas ruedas no son los mismos durante las curvas, como corresponde a la condición de Ackerman de no deslizamiento. Pero dada la dificultad de medir con precisión la geometría de las barras de dirección, se ha supuesto un cumplimiento perfecto de ésta condición:

$$\tan\delta_0 = \frac{L}{\left(R + \frac{t}{2}\right)} \tag{5.1}$$

$$\tan\delta_i = \frac{L}{\left(R - \frac{t}{2}\right)} \tag{5.2}$$

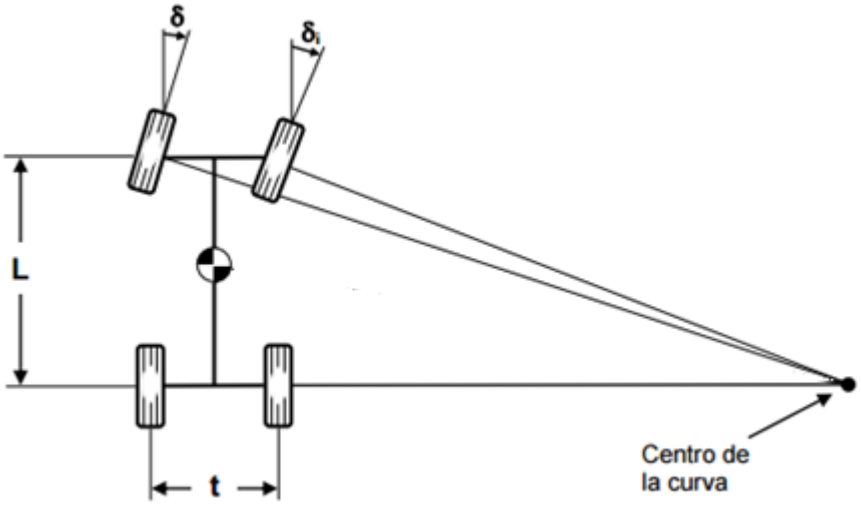

Fig 5.1 Geometría de giro de un vehículo

Siendo *R,* la distancia desde el centro de la curva hasta el centro del eje trasero.

Los ángulos δ_0 y δ_i de las ruedas se calculan para que los ejes de giro de las mismas se corten sobre el mismo punto del eje delantero de la máquina, siendo ese punto el centro instantáneo de rotación (CIR) del movimiento plano del chasis.

6 DESCRIPCIÓN DEL SIMULADOR DE LA CARRETILLA

El entorno del simulador se basa en un polígono industrial en un día despejado. En él, se pueden distinguir diferentes parcelas con sus correspondientes naves industriales.

Centrándose en la parcela que nos ocupa, se puede ver la carretilla elevadora y los diferentes tipos de cargas que se utilizan en la simulación, delante de una nave industrial que se encuentra en construcción. Además de esto, dentro de la parcela también se puede destacar que hay un camión volquete y una grúa torre.

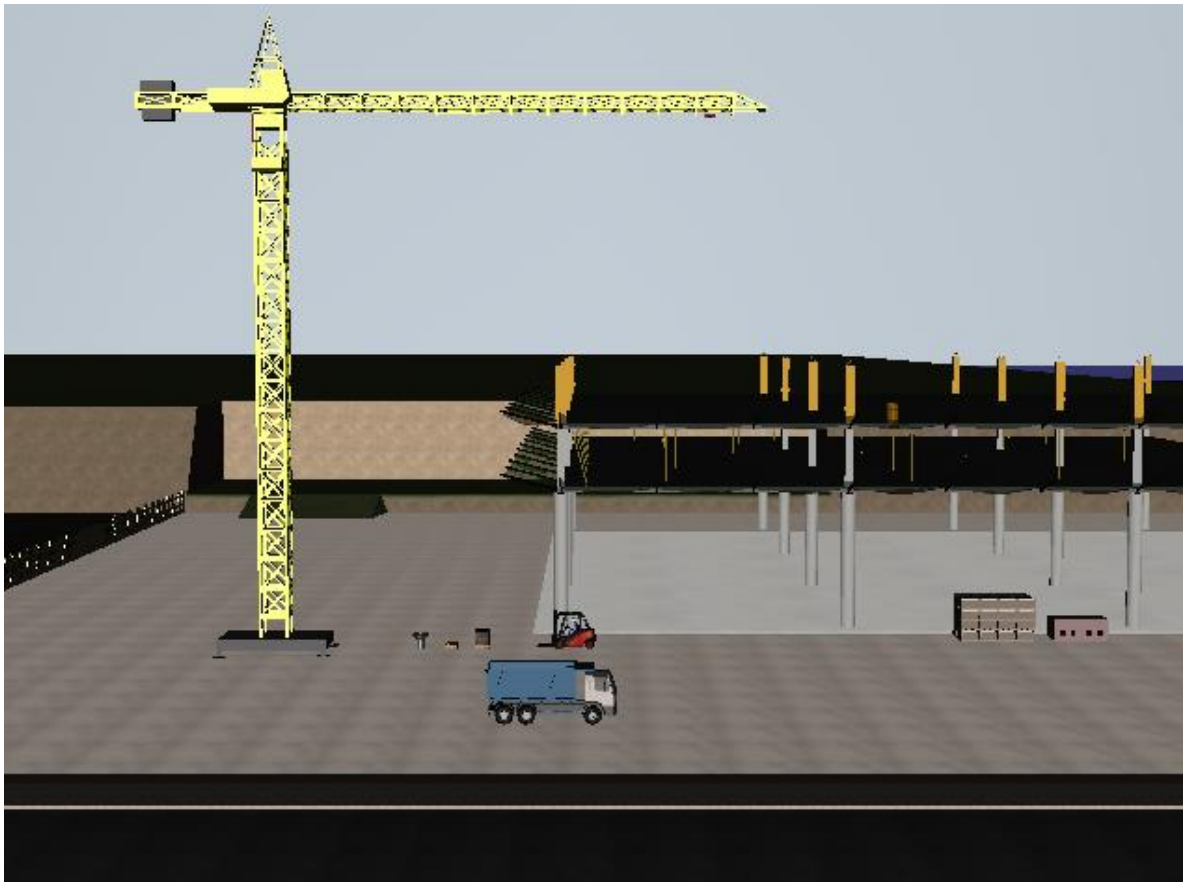

Fig 6.1 Panorámica del entorno de nuestro simulador

En el simulador se pueden controlar tanto los movimientos de inclinación del mástil y elevación y descenso de la horquilla, como los movimientos de dirección y transmisión de la carretilla. Durante la simulación, existe la posibilidad de visualizar las fuerzas que ejercen tanto en la carretilla como en los distintos tipos de carga.

Los controles, se pueden dividir en dos partes. El control del entorno del simulador, como el inicio de la reproducción, la velocidad de simulación y el control de las cámaras, que se realizan con el teclado y el ratón del ordenador. Y el control de movimientos de la carretilla elevadora, realizados con los mandos simulando los controles de la carretilla real (ver Anexo II).

Respeto al control del entorno del simulador, no tiene demasiado sentido aplicarlo en un simulador definitivo ya que utilidades como cambiar la velocidad de simulación o rebobinar no se utilizarían, pero puesto que estas son funcionalidades de la MBSmodel, se han

aprovechado para poder hacer pruebas de maniobras y poder verificar ciertos aspectos de la simulación con más detalle. Los controles de cámara se realizan mediante el ratón como se ha explicado en el apartado 4.2.4. El resto de controles del entorno se realizan mediante el teclado. Se utilizarán la flecha hacia la derecha "→" para comenzar con la simulación, la flecha hacia la izquierda "←" para rebobinar la reproducción y las flechas hacia abajo "↓" y hacia arriba "↑", se utilizarán para disminuir o aumentar la velocidad de reproducción respectivamente. Con el teclado también se pueden ocultar o mostrar las fuerzas que se ejercen sobre la carretilla y sobre los distintos tipos de carga con la letra "a" e incluso volver al enfoque principal de la cámara sobre la carretilla con la tecla espaciadora.

Respeto al control de movimientos de la carretilla elevadora, se ha utilizado un joystick y un volante con sus pedales simulando a los controles reales. En primer lugar, agrupamos los 5 GDL guiados en un vector. Con el joystick, ya que se dispone de dos ejes, se realizan los movimientos de los GDL 3 y 4, correspondientes a la inclinación del mástil y el descenso o la elevación de la horquilla respectivamente. Para la inclinación del mástil hacia delante y hacia atrás, de debe inclinar el joystick hacia delante y hacia atrás respectivamente. Para la elevación y descenso de la horquilla, se debe inclinar el joystick hacia la derecha e izquierda respectivamente. Es decir, ↑ inclinación mástil hacia delante, ↓ inclinación mástil hacia atrás, → elevación horquilla y ← descenso de horquilla.

Estos controles corresponden con los del modelo de carretilla elevadora que hemos simulado (ver Anexo II).

Fig 6.2 Joystick, inclinación del mástil hacia delante

Fig 6.3 Joystick, inclinación del mástil hacia atrás

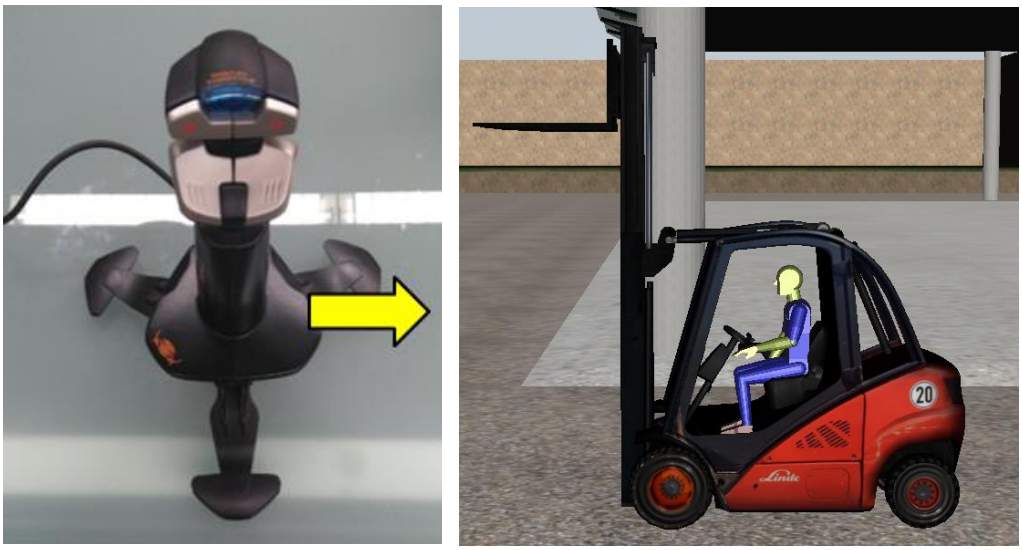

Fig 6.4 Joystick, elevación de horquilla

Fig 6.5 Joystick, descenso de horquilla

Con el volante se controlan los movimientos de los GDL 1 y 2, correspondientes a la dirección de cada una de las ruedas traseras. Aunque al volante solo corresponde a un eje, se controlan dos GDL ya que realmente lo se puede interpretar como un único GDL puesto que como se ha visto anteriormente en el apartado 5, los ángulos de giro están relacionados por la condición de Ackerman.

Fig 6.6 Volante, control de dirección

Y por último, con los pedales se controla el movimiento del GDL 5, correspondiente a la transmisión de la carretilla. Con el pedal derecho del acelerador, color verde en la Fig 6.7, se inicia la marcha hacia delante de la carretilla y a medida que se levanta el pie de este, la carretilla frena de manera proporcional a lo que se vaya soltando el pedal, llegando al reposo cuando el pedal está suelto completamente. Con el pedal izquierdo, color rojo en la Fig 6.7, se inicia la marcha atrás de la carretilla e igual que en el caso anterior, la aceleración de la marcha disminuye según se va levantando soltando el pedal. Este fenómeno se ha descrito con más detalle en el apartado 2.4.6.2.

Fig 6.7 Pedales, control de velocidad

Lo que permite el guiado de estos movimientos, es la posibilitación de realización de maniobras que consistan en el transporte de los diferentes tipos de carga.

Fig 6.8 Transporte de la virola

Por último, se ha implantado la funcionalidad de reproducir la maniobra realizada anteriormente para poder ajustar parámetros sin la necesidad de simular una maniobra nueva cada vez, para ello se ha creado en la función guiada.f90 una variable llamada cargaGuiado, que cuando su valor corresponde con .false., el usuario controla los mandos

para el guiado de la carretilla y cuando corresponde con .true., se cargan los datos de la maniobra anterior guardada en el archivo maniobras.csv y la reproduce.

7 CONCLUSIONES

Se puede decir que el objetivo principal de este trabajo se ha logrado, puesto que se había planteado desarrollar un simulador de la carretilla elevadora que se pudiese conducir en tiempo real con unos mandos que replicasen a los reales de la máquina y se ha conseguido.

Otro objetivo cumplido es el manejo sencillo y semejante al modelo real con una interfaz agradable, lo que puede llevar al usuario a familiarizarse con los controles de la carretilla real y sus movimientos.

Como se ha expresado al principio de la memoria, entre los factores que motivaron este trabajo estaba la petición de la simulación de una maniobra de vuelco real de una de máquina con resultado fatal para la vida del operario con objeto de ser empleado como prueba pericial en un juicio. Ahora con este simulador se pueden realizar multitud de maniobras diferentes observando movimientos críticos de la máquina, limitaciones, etc. que pueden servir tanto para evitar accidentes, como para incluir limitaciones en los movimientos de la máquina e incluso para mejorar sus características.

8 BIBLIOGRAFÍA

Blender. *Blender.* https://www.blender.org/

Cmake. *Cmake*. https://cmake.org/

Daniel Dopico Dopico, Alberto Luaces Fernández, Urbano Lugrís Armesto, Javier Cuadrado Aranda, Mariano Saura Sánchez, Francisco Javier González Varela, Emilio Sanjurjo Maroño, Roland Pastorino. (,22/06/2016) *"Multibody Systems en Laboratorio de Ingeniería Mecánica (MBSLIM)"*.
Registro de la propiedad intelectual, C-240-2016. Software para la simulación dinámica de sistemas multicuerpo.

David Vilela Freire. (2011-10). *Simulador de grúa Panamax para movimiento de contenedores en puerto* (Proyecto fin de carrera), Ferrol.

Dopico, D.; Luaces, A.; Gonzalez, M. & Cuadrado, J. (2011). *Dealing with multiple contacts in a human-in-the-loop application Multibody System Dynamics.*

Emilio Sanjurjo Maroño. (2011-07). *Modelo multicuerpo de automóvil para su aplicación en técnicas de estimación de estados* (Proyecto fin de carrera), Ferrol.

Forklift international. (2017-98). *Forklift international.* http://www.forklift-international.com/es/

Hans Cristian Muller Santa Cruz. (2007). *Programando en Fortran.*

Ing. Arturo J. López García. (2004). *Guía de Programación en Fortran 90/95.* http://hep.fcfm.buap.mx/cursos/2012/FCII/fortran95.pdf

J. Garcia de Jalon, E. Bayo. (1994). *Kinematic and Dynamic Simulation of Multibody Systems,* Springer-Verlag.

Jeanne C. Adams, Walter S. Brainerd, Jeanne T. Martin, Brian T. Smith, Jerrold L. Wagener. (1992). *Fortran 90 Handbook, Complete ANSI / ISO Reference.* Intertext Publications McGraw-Hill Book Company

Laboratorio de Ingeniería Mecánica. *Laboratorio de Ingeniería Mecánica*. http://lim.ii.udc.es/

Linde. (2014). *Manual original carretilla diésel Linde H25D-02, H30D-02 y H35D-02.Serie 393-02.*

Linde. *Linde.* http://www.linde-mh.es

Pedro Cobo Beltrán. (2011-10). *Simulador de grúa hidráulica para aplicaciones marinas.* (Proyecto fin de carrera), Ferrol.

Peter Class, Pello Xabier Altadill Izura. (2004). *Tutorial de C++: o el diario de Peter Class.* http://es.tldp.org/Manuales-LuCAS/doc-tutorial-c++/doc-tutorial-c++.pdf

Rill, G. (2009). *"Dynamic Tire Forces With Smooth Transition to Stand–Still,"* 7th EUROMECH Solid Mechanics Conference, Lisbon, Portugal. [33] Hirschberg, W., Rill, G., and Weinfurter, H., 2007, "Tire Model TMeasy," Veh. Syst. Dyn.

Roland Pastorino. (2012). *Experimental validation of a multibody model for a vehicle prototype and its application to state observers* (Tesis Doctoral). Ferrol.

Siemens. (2017). *Solid Edge* https://www.plm.automation.siemens.com/es_es/products/solid-edge/

Visual Studio. (2017). *Visual Studio*. https://www.visualstudio.com

9 ANEXOS

Anexo I: Código

Anexo II: Manual original carretilla diésel Linde H25D-02, H30D-02 y H35D-02.Serie 393-02.