



**UNIVERSIDAD POLITÉCNICA DE MADRID**

**ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA  
AERONÁUTICA Y DEL ESPACIO**

**DESARROLLO DEL MÓDULO DE CÁLCULO PARA EL  
SISTEMA DE CONTROL DE ACTITUD DE UN SATÉLITE**

**TUTOR: SEBASTIÁN FRANCHINI**

**AUTORA: ANDREA MACARULLA RODRÍGUEZ**

**CIENCIAS Y TECNOLOGÍAS AEROESPACIALES**

**2013/2014**



# Resumen

En el curso 2013/2014 cuatro alumnos hemos realizado las prácticas en las instalaciones del IDR del campus de Montegancedo. Nuestra misión fue poner a prueba la base de datos diseñada en este instituto haciendo cada uno de nosotros un módulo de cálculo para un satélite que utilizase dicha base de datos.

El presente trabajo se divide en tres partes.

En la primera se describe lo que es una instalación y una base de datos de ingeniería concurrente. Se explica como el módulo diseñado en Matlab se conecta a esta base de datos para extraer e insertar información de una forma coherente con el conjunto de módulos.

En la segunda parte se describe el módulo realizado en Matlab para el control de actitud de un satélite. Este módulo consta de tres interfaces de usuario consecutivos y emplea las fórmulas del cálculo de perturbaciones y de diseño de componentes.

En la tercera parte, ya independiente de las otras dos, se explica el desarrollo del modelo matemático que simula el movimiento de traslación de un satélite en una órbita estacionaria y el diseño del sistema de control para conseguir que un eje del satélite apunte a una dirección dada sin importar que éste gire sobre este eje. Así mismo se hace un diseño de la simulación de las perturbaciones, y se hace una evaluación de la sintonización conseguida del control PID utilizado y la estrategia escogida para implantar dicho control.



# Abstract

In the 2013/2014 year four students have done internships in campus Montegancedo IDR facilities . Our task was to test the database in this institute. Each of us designed a satellite module that used this database.

This project is divided in three parts.

The first describes what a concurrent engineering facility and database are. It explains how the module designed in Matlab connects to this database to extract and insert information in a coherent method with all modules.

In the second part the Matlab module made for the attitude control of a satellite is described. This module consists of three main consecutive user interfaces and their auxiliaries. This module also has the formula used for calculating disturbances and component design.

In the third part, which is independent of the other two, the development of the mathematical model that simulates the translational motion of a satellite in a stationary orbit is explained. The control system design for a satellite whose z axis is pointing to a direction regardless of the rotation velocity around this axis is also described. A design simulation of disturbances is performed too, and an assessment of the achieved tuning PID control used and chosen to implement this control strategy is made.



# Índice general

<b>I</b>	<b>CDF y ADCS</b>	<b>21</b>
<b>1.</b>	<b>Introducción</b>	<b>23</b>
1.1.	Concurrent Design Facility . . . . .	23
1.2.	Motivación y objetivos . . . . .	24
1.2.1.	Motivación . . . . .	24
1.2.2.	Objetivos . . . . .	25
1.3.	Sistema de control de actitud . . . . .	25
<b>2.</b>	<b>Bases de datos</b>	<b>27</b>
2.1.	Introducción . . . . .	27
2.2.	Creación . . . . .	28
2.2.1.	CDF . . . . .	28
2.2.2.	Base de datos particular del módulo . . . . .	28
2.3.	Issue/versión . . . . .	29
2.4.	Módulos de conexión . . . . .	31
2.4.1.	CDF . . . . .	31
2.4.2.	Base de datos particular del módulo . . . . .	31
2.5.	Programación CDF . . . . .	32
2.5.1.	Import . . . . .	32
2.5.2.	Export . . . . .	35
<b>3.</b>	<b>Conclusiones de la Parte I</b>	<b>41</b>
3.1.	Cumplimiento de objetivos . . . . .	41
3.2.	Trabajos futuros . . . . .	41

<b>II</b>	<b>Diseño de prefase-A</b>	<b>43</b>
<b>4.</b>	<b>Perturbaciones</b>	<b>45</b>
4.1.	Concepto teórico . . . . .	45
4.1.1.	Perturbación solar . . . . .	46
4.1.2.	Perturbación debido al campo magnético . . . . .	46
4.1.3.	Perturbación aerodinámica . . . . .	47
4.1.4.	Perturbación por gradiente de gravedad . . . . .	48
4.2.	Programación . . . . .	49
4.2.1.	Función recupera_variables . . . . .	50
4.2.2.	Función dato_r . . . . .	50
4.2.3.	Función dato_w . . . . .	50
4.2.4.	Botones del cuadro de diálogo . . . . .	53
<b>5.</b>	<b>Actuadores sin combustible</b>	<b>55</b>
5.1.	Conceptos teóricos . . . . .	55
5.1.1.	Ruedas de inercia . . . . .	55
5.1.2.	Ruedas de reacción . . . . .	56
5.1.3.	Pares magnéticos . . . . .	57
5.2.	Programación . . . . .	58
5.2.1.	Interfaz de datos . . . . .	58
5.2.2.	Operaciones con el catálogo . . . . .	61
5.2.3.	Selección y situación . . . . .	67
5.2.4.	Cálculos . . . . .	76
<b>6.</b>	<b>Actuadores con combustible</b>	<b>79</b>
6.1.	Conceptos teóricos . . . . .	79
6.2.	Programación . . . . .	82
6.2.1.	Interfaz de datos . . . . .	82
6.2.2.	Selección y situación de propulsores . . . . .	83
6.2.3.	Cálculos . . . . .	87
6.2.4.	Grabar en BD . . . . .	88
<b>7.</b>	<b>Sensores</b>	<b>91</b>
7.1.	Conceptos teóricos . . . . .	91
7.1.1.	Sensor de Sol . . . . .	91
7.1.2.	Sensor de estrellas . . . . .	94
7.1.3.	Sensor horizonte . . . . .	95



7.1.4. Giróscopos . . . . .	97
7.1.5. Magnetómetros . . . . .	99
7.2. Programación . . . . .	101
7.2.1. Selección y situación de sensores . . . . .	101
7.2.2. Cálculos . . . . .	102
<b>8. Conclusiones de la Parte II</b>	<b>105</b>
8.1. Cumplimiento de objetivos . . . . .	105
8.2. Trabajos futuros . . . . .	106
<b>III Simulación</b>	<b>107</b>
<b>9. Sistemas de referencia</b>	<b>109</b>
9.1. Introducción . . . . .	109
9.2. Sistemas de referencia . . . . .	109
9.2.1. Ejes inerciales . . . . .	109
9.2.2. Ejes órbita . . . . .	110
9.2.3. Ejes trayectoria . . . . .	110
9.2.4. Ejes cuerpo . . . . .	111
9.3. Matrices de rotación . . . . .	113
9.4. Cuaterniones y rotación espacial . . . . .	115
9.4.1. Introducción . . . . .	115
9.4.2. Rotación . . . . .	117
9.4.3. Operaciones con cuaterniones . . . . .	118
<b>10. Modelo dinámico</b>	<b>119</b>
10.1. Conceptos teóricos . . . . .	119
10.2. Modelo Simulink de la dinámica del satélite . . . . .	120
10.2.1. Descripción del subsistema de cálculo de velocidad angular . . . . .	120
10.2.2. Comprobación de modelo de cálculo de velocidad angular	121
10.2.3. Descripción del subsistema de cálculo de actitud . . . . .	123
10.2.4. Integración de subsistema cálculo de velocidad angular y cálculo de actitud . . . . .	124
<b>11. Simulación de órbitas</b>	<b>127</b>
11.1. Conceptos teóricos . . . . .	127

11.1.1. Órbita en su plano . . . . .	127
11.1.2. Paso de órbita a sistema inercial . . . . .	128
11.1.3. Eclipse . . . . .	129
11.2. Modelo Simulink de cálculo de órbita de satélite . . . . .	130
11.2.1. Modelo Simulink de cálculo de órbita en ejes órbita . .	130
11.2.2. Modelo Simulink de transferencia de ejes órbita a ejes inerciales . . . . .	131
11.2.3. Integración de los bloques de cálculo de órbita y paso a coordenadas inerciales . . . . .	132
11.2.4. Comprobación de la órbita del satélite en ejes órbita .	133
11.3. Modelo Simulink de cálculo de órbita solar . . . . .	134
11.3.1. Cálculo de la órbita del Sol . . . . .	134
11.4. Cálculo de eclipse . . . . .	134
11.4.1. Distribución de tiempos de eclipse . . . . .	135
<b>12.Simulación de perturbaciones</b>	<b>147</b>
12.1. Introducción . . . . .	147
12.2. Perturbación aerodinámica . . . . .	147
12.2.1. Conceptos teóricos . . . . .	147
12.2.2. Modelo Simulink . . . . .	148
12.2.3. Módulo área proyectada . . . . .	149
12.2.4. Densidad del aire . . . . .	150
12.2.5. Simulación . . . . .	150
12.3. Perturbación solar . . . . .	151
12.3.1. Conceptos teóricos . . . . .	151
12.3.2. Modelo Simulink . . . . .	151
12.3.3. Simulación . . . . .	152
12.4. Perturbación magnética . . . . .	153
12.4.1. Conceptos teóricos . . . . .	153
12.4.2. Modelo Simulink . . . . .	153
12.4.3. Cálculo del campo magnético terrestre . . . . .	154
12.4.4. Simulación . . . . .	155
12.5. Perturbación por gradiente de gravedad . . . . .	155
12.5.1. Conceptos teóricos . . . . .	155
12.5.2. Modelo Simulink para el control de actitud . . . . .	155
12.5.3. Simulación . . . . .	156
12.6. Integración de los pares de perturbación al modelo global . .	156

<b>13. Control de actitud</b>	<b>165</b>
13.1. Introducción . . . . .	165
13.2. Subsistema de control . . . . .	165
13.2.1. Estrategia seguida . . . . .	165
13.2.2. Conceptos teóricos . . . . .	167
13.2.3. Modelo Simulink . . . . .	170
13.2.4. Sintonización PID . . . . .	171
<b>14. Ejecución del modelo</b>	<b>185</b>
14.1. Introducción . . . . .	185
14.2. Establecimiento de consigna . . . . .	185
14.3. Comportamiento dinámico . . . . .	186
14.3.1. Orientación hacia una dirección fija en ejes inerciales . . . . .	187
14.4. Acumulación de momento . . . . .	190
14.5. Variaciones del comportamiento del modelo dinámico . . . . .	191
14.5.1. Escenario 1: $J_{xy} \neq 0$ . . . . .	191
14.5.2. Escenario 2: $J_{yz} \neq 0$ . . . . .	193
14.5.3. Escenario 3: $J_{xz} \neq 0$ . . . . .	195
14.5.4. Escenario 4: $J_{xz} \neq 0$ y $J_{yz} \neq 0$ . . . . .	198
14.5.5. Escenario 5: $J_{yz} \neq 0$ con perturbaciones . . . . .	200
14.5.6. Escenario 6: $J_{yz} \neq 0$ y $J_{xz} \neq 0$ con perturbaciones . . . . .	203
<b>15. Conclusiones de la Parte III</b>	<b>205</b>
15.1. Cumplimiento de objetivos . . . . .	205
15.2. Comentarios . . . . .	205
15.3. Trabajos futuros . . . . .	207



# Índice de figuras

1.1. Sala CDF del IDR en el campus de Montegancedo . . . . .	24
2.1. Diseño de base de datos en CDF . . . . .	28
2.2. Algoritmo para la comprobación de coherencia del <i>Issue</i> y <i>version</i> . . . . .	32
2.3. Configuración de driver de conexión . . . . .	33
2.4. Comprobación de conexión a la base de datos con la toolbox de Matlab . . . . .	34
2.5. Configuración ODBC para la base de datos particular . . . . .	35
4.1. Interfaz gráfica de perturbaciones . . . . .	49
4.2. Mensaje de error . . . . .	53
5.1. Rueda de inercia RSI45 . . . . .	56
5.2. Pares magnéticos instalados en el satélite LUSEX . . . . .	58
5.3. Interfaz de actuadores . . . . .	61
5.4. Catálogo de ruedas . . . . .	62
5.5. Contenido de la tabla de ruedas elegidas . . . . .	69
5.6. Cuadro de diálogo para situar una rueda . . . . .	78
6.1. Maniobras de momento cero y momento continuo . . . . .	80
6.2. Interfaz de cálculo de propulsores . . . . .	83
6.3. Interfaz de colocación de propulsores. Elección de la cara sobre la que va a situarse el sensor. . . . .	87
6.4. Interfaz de propulsores. Situación sobre la cara. . . . .	89
6.5. Interfaz de colocación de propulsores. Orientación del propulsor. 90	
7.1. Esquema de sensor de Sol de rejilla. Extraído de [10] . . . . .	92
7.2. Esquema de sensor de sol de dos rejillas. Extraído de [10] . . . . .	93

7.3. Definición de los ángulos del sensor de Sol . . . . .	94
7.4. Imagen del sensor de estrellas SED26 de la empresa Sodern . .	95
7.5. Imagen sensor horizonte IRES-C de la ESA . . . . .	98
7.6. Imagen de un giróscopo . . . . .	99
7.7. Magnetómetro instalado en la nave Pioneer 10 y 11 . . . . .	101
7.8. Catálogo de sensores . . . . .	102
7.9. Interfaz para colocar un sensor. Selección de cara . . . . .	103
7.10. Interfaz para posicionar y orientar un sensor. . . . .	104
9.1. Sistema de referencia inercial heliocéntrico . . . . .	111
9.2. Sistema de referencia inercial geocéntrico . . . . .	112
9.3. Sistema de referencia no inercial asociado al plano de la órbita	113
9.4. Sistema de referencia no inercial en ejes cuerpo . . . . .	114
9.5. Ángulos de rotación de Euler zxz . . . . .	116
9.6. Representación de rotación alrededor de un eje . . . . .	117
10.1. Subsistema de respuesta dinámica . . . . .	121
10.2. Resultado de la simulación de prueba . . . . .	124
10.3. Subsistema que calcula el cuaternión que determina la actitud	125
10.4. Integración del sistema dinámico y cálculo de actitud . . . . .	125
11.1. Parámetros de la elipse . . . . .	128
11.2. Ángulos de la elipse . . . . .	136
11.3. Esquema de las condiciones de eclipse . . . . .	137
11.4. Cuadro de diálogo para introducir los parámetros de la órbita	137
11.5. Subsistema para el cálculo de órbita en ejes órbita . . . . .	138
11.6. Subsistema de conversión de ejes órbita a ejes inerciales . . . . .	138
11.7. Módulo de la librería Simulink para conversión de ejes inerciales a ejes órbita . . . . .	139
11.8. Cálculo de la órbita en ejes inerciales . . . . .	139
11.9. Cálculo de orbitas y eclipse del satélite y Sol . . . . .	140
11.10. Dibujo de la órbita con la Tierra en el foco . . . . .	141
11.11. Comprobación de que la energía mecánica permanece constante	142
11.12. Descomposición del cálculo de la órbita del Sol . . . . .	142
11.13. Parámetros de la órbita solar . . . . .	143
11.14. Órbita del Sol en ejes órbita . . . . .	143
11.15. Conversión del radio vector del Sol a coordenadas inerciales . .	144
11.16. Bloque Simulink para la determinación de eclipse . . . . .	144

11.17	Periodo de eclipse el día 21 de Marzo (equinoccio de primavera)	145
12.1.	Subsistema para el cálculo del par de perturbación aerodinámico	148
12.2.	Subsistema que calcula el área proyectada . . . . .	150
12.3.	Cálculo de la densidad del aire en función de la altura . . . . .	157
12.4.	Perturbación aerodinámica en módulo . . . . .	159
12.5.	Subsistema para el cálculo del par Solar . . . . .	159
12.6.	Perturbación Solar en módulo . . . . .	160
12.7.	Representación del dipolo terrestre . . . . .	160
12.8.	Subsistema que calcula el par magnético . . . . .	161
12.9.	Subsistema del cálculo del campo magnético terrestre . . . . .	161
12.10	Perturbación debida al campo magnético terrestre en módulo	162
12.11	Subsistema que calcula el par de gradiente de gravedad . . . .	162
12.12	Cálculo del vector unitario y el módulo al cubo . . . . .	163
12.13	Perturbación debida al gradiente de gravedad en módulo . . .	163
12.14	Integración de los cuatro pares de perturbación al modelo global	164
13.1.	Integración de los subsistemas de cálculo de pares, modelo dinámico, cálculo de órbitas y control . . . . .	166
13.2.	Determinación del error . . . . .	167
13.3.	Parámetros del bloque de control . . . . .	168
13.4.	Subsistema para el control de actitud . . . . .	170
13.5.	Orientación del satélite con control PID para el método de Ziegler-Nichols . . . . .	173
13.6.	Orientación del sistema en bucle abierto . . . . .	174
13.7.	Orientación del sistema con control proporcional . . . . .	175
13.8.	Orientación del sistema con control proporcional derivativo . .	176
13.9.	Velocidad angular con control proporcional derivativo . . . . .	177
13.10	Error estacionario con control PD . . . . .	178
13.11	Orientación del sistema con control proporcional integral de- rivativo . . . . .	179
13.12	Velocidad angular con control proporcional integral derivativo	180
13.13	Error estacionario con control PID . . . . .	181
13.14	Orientación del sistema con control proporcional derivativo . .	182
13.15	Velocidad angular con control proporcional derivativo . . . . .	183
14.1.	Subsistema para la asignación de consigna . . . . .	185
14.2.	Velocidad angular $\omega$ en los tres ejes . . . . .	187

14.3. Orientación del satélite, convergencia XY . . . . .	188
14.4. Detalle de la figura 14.3 . . . . .	189
14.5. Par de control en los tres ejes . . . . .	190
14.6. Detalle de la figura 14.5 . . . . .	191
14.7. Momento acumulado en las tres direcciones. . . . .	192
14.8. Convergencia del apuntamiento cuando $J_{xy} \neq 0$ . . . . .	192
14.9. Detalle ampliado del par de control con $J_{xy} \neq 0$ una vez esta- bilizado . . . . .	193
14.10 Convergencia del apuntamiento cuando $J_{yz} \neq 0$ . . . . .	194
14.11 Momento acumulado cuando $J_{yz} \neq 0$ . . . . .	195
14.12 Convergencia de apuntamiento cuando $J_{xz} \neq 0$ . . . . .	196
14.13 Pares de control para el caso de $J_{xz} \neq 0$ . . . . .	197
14.14 Momentos acumulados cuando $J_{xz} \neq 0$ . . . . .	198
14.15 Convergencia cuando $J_{xz} \neq 0$ y $J_{yz} \neq 0$ . . . . .	199
14.16 Velocidad angular cuando $J_{xz} \neq 0$ y $J_{yz} \neq 0$ . . . . .	199
14.17 Momento acumulado con $J_{xz} \neq 0$ y $J_{yz} \neq 0$ . . . . .	200
14.18 Convergencia cuando $J_{yz} \neq 0$ con perturbaciones . . . . .	201
14.19 Velocidad angular cuando $J_{yz} \neq 0$ con perturbaciones . . . . .	202
14.20 Pares de control cuando $J_{yz} \neq 0$ con perturbaciones . . . . .	202
14.21 Convergencia cuando $J_{xz} \neq 0$ y $J_{yz} \neq 0$ con perturbaciones . . .	203
14.22 Pares de control cuando $J_{xz} \neq 0$ y $J_{yz} \neq 0$ con perturbaciones	204



# Listados de código

2.1.	Función que importa los datos de la base de datos concurrente	36
2.2.	Función de mapeo entre variables . . . . .	38
4.1.	Función recupera_variable . . . . .	51
4.2.	Función dato_r . . . . .	52
4.3.	Función dato_w . . . . .	52
4.4.	Traspaso de variables de un interfaz a otro . . . . .	54
5.1.	Inicialización del cuadro de diálogo de actuadores . . . . .	59
5.2.	Función para cargar variables en pantalla . . . . .	60
5.3.	Código para seleccionar un registro dentro de una tabla . . . .	64
5.4.	Crear un registro a partir de otros existentes . . . . .	65
5.5.	Código para hacer la copia de un registro . . . . .	66
5.6.	Código para borrar un registro . . . . .	68
5.7.	Recogida de datos de un cuadro de diálogo en su llamada . . .	71
5.8.	Verificación de que el dato introducido está dentro de límites .	72
5.9.	Cumplimentar las coordenadas xy gráficamente con el botón posición y cuadrícula . . . . .	74
5.10.	Llamada a la rutina de colocación de la rueda e inserción en la base de datos . . . . .	75
5.11.	Función para calcular masas, centro de gravedad y tensor de inercia de los elementos colocados . . . . .	77
6.1.	Codificación de las caras del satélite en la base de datos . . . .	85
6.2.	Representación de la cara del satélite elegida . . . . .	86



# Índice de cuadros

6.1. Codificación de las caras del satélite . . . . .	88
9.1. Sistemas de referencia inerciales . . . . .	110
10.1. Valores del ejemplo numérico para comprobación . . . . .	123
12.1. Tabla de valores de densidad del aire . . . . .	158
13.1. Efectos al incrementar cada parámetro por separado [8] . . . .	169
13.2. Tabla de valores para sintonizar un PID según el método Ziegler-Nichols . . . . .	170



# Parte I

## CDF y ADCS



# Capítulo 1

## Introducción

### 1.1. Concurrent Design Facility

“Concurrent Design Facility” o Instalación de Diseño Concurrente es un conjunto de ordenadores y dispositivos multimedia interconectados para que diseñadores de diferentes disciplinas puedan trabajar simultáneamente en un proyecto complejo intercambiando programas y datos. De este modo se puede llegar a una solución completa aproximándose de forma continua en la que todos los ingenieros han participado resolviendo los problemas y llegando a una solución de compromiso. Esta sala está equipada con dispositivos de comunicación de modo que puedan participar en las reuniones personas que no se encuentren en la sala.

“THE CONCURRENT DESIGN FACILITY (CDF) IS A STATE-OF-THE-ART FACILITY EQUIPPED WITH A NETWORK OF COMPUTERS, MULTIMEDIA DEVICES AND SOFTWARE TOOLS, WHICH ALLOWS A TEAM OF EXPERTS FROM SEVERAL DISCIPLINES TO APPLY THE CONCURRENT ENGINEERING METHOD TO THE DESIGN OF FUTURE SPACE MISSIONS. IT FACILITATES A FAST AND EFFECTIVE INTERACTION OF ALL DISCIPLINES INVOLVED, ENSURING CONSISTENT AND HIGH-QUALITY RESULTS IN A MUCH SHORTER TIME.”

[5]

La sede de la Agencia Espacial Europea (ESA) se encuentra en Noordijk, Países Bajos. El presente trabajo se ha desarrollado en el Instituto de Microgravedad “Ignacio da Riva”, situado en el campus de Montegancedo en Pozuelo de Alarcón. Dicho instituto dispone de una instalación de este tipo

para fines educativos, gracias a un acuerdo con ESA. Para más información sobre esta sede, consultar la bibliografía. [6]



Figura 1.1: Sala CDF del IDR en el campus de Montegancedo

## 1.2. Motivación y objetivos

### 1.2.1. Motivación

El objetivo principal del presente trabajo es el desarrollo de un módulo de cálculo del control de dinámica y actitud de un satélite o ADCS en sus siglas en inglés. Este módulo pertenece a un proyecto de ingeniería concurrente (“Concurrent Engineering”) para desarrollar la pre-fase A de un satélite artificial. La ingeniería concurrente consiste en el trabajo en equipo multidisciplinar en tiempo real para realizar de forma ágil lo que de otro modo sería un proceso que duraría varios meses. Los diseños de los diferentes sistemas se hacen simultáneamente con concordancia e intercambio de datos. La prefase A o fase 0 en el ciclo de vida de un proyecto consiste en la formulación de un programa e identificación de misiones potenciales, apoyadas por estudios avanzados, compuesto por análisis, simulaciones y ciertas investigaciones o desarrollos preliminares. Las principales metas son formular los objetivos de misión, los requerimientos a nivel de sistemas, concepto preliminar de operaciones, costos estimados, planificación estimada, e identificar



las tecnologías que deben desarrollarse. Este módulo interactúa con módulos similares (planta de potencia, órbita, control térmico, estructuras, etc.). La conexión entre módulos se lleva a cabo mediante una base de datos común donde cada módulo es propietario de ciertas variables y puede solicitar acceso de las variables del resto de módulos. El conjunto de módulos sirve para poder ejecutar la pre-fase A del diseño del satélite de forma concurrente. [9]. Para dar un paso más en el diseño del módulo ADCS, se ha incluido un modelo Simulink que permite observar el comportamiento del control a lo largo del tiempo frente a las perturbaciones.

### 1.2.2. Objetivos

El IDR estableció un acuerdo con la ESA por el que se les proporcionó software para la ingeniería concurrente pero dicho software no era lo suficientemente flexible. Por lo tanto el objetivo de las prácticas y por tanto el del proyecto es desarrollar un software equivalente pero más actualizado.

Los objetivos del presente proyecto son los siguientes:

- Conexión de un programa externo (Matlab) a la base de datos del CDF
- Diseño de una interfaz de usuario para el diseño de la prefase-A en la herramienta Guide de Matlab
- Desarrollo de aplicaciones para manipulación de datos con la herramienta Matlab
- Simular las cuatro perturbaciones principales que afectan a la actitud
- Sintonizar un control PID utilizando el modelo sin perturbaciones y comprobar que es eficiente cuando las hay.

## 1.3. Sistema de control de actitud

El sistema de control de actitud de un satélite es aquel que se encarga de mantener la orientación del satélite hacia la dirección deseada en cada momento. Según el tipo de misión el satélite puede apuntar la mayor parte del tiempo a un punto fijo de la esfera celeste (por ejemplo el telescopio Hubble), a un punto fijo de la superficie terrestre (un satélite de comunicaciones), seguir siempre la vertical local (observación meteorológica), entre otros

casos. Además de mantener una orientación predeterminada puede que el satélite necesite realizar maniobras periódicas. Por ejemplo cambio de objetivo, transmisión de datos a Tierra, maniobras de control térmico, etc. Por último, para el correcto desarrollo de la misión y para mantener la precisión requerida hace falta micromaniobras de ajuste fino con el fin de corregir las pequeñas desviaciones causadas por las perturbaciones. Para determinar la actitud, el subsistema cuenta con sensores de dirección. Estos sensores pueden ser solares, de horizonte (o infrarrojos), estrellas, giróscopos y magnetómetros. Dichos sensores son responsables de recoger las medidas necesarias para poder calcular la orientación real del satélite. Para conseguir la actitud deseada, se necesitan actuadores que generan un par que saca al satélite del equilibrio y le hace girar alrededor de su centro de masas. Los actuadores pueden ser propulsores (microcohetes), estabilización por spin (el propio satélite es un giróscopo), estabilización por gradiente de gravedad, pares magnéticos, ruedas de inercia y ruedas de reacción. La lógica de control es el conjunto de algoritmos que activa los actuadores para conseguir la actitud deseada en función de la lectura de los sensores. Con la lectura de sensores se mide la orientación actual, se compara con la orientación deseada y en función de esta diferencia y su variación en el tiempo (teniendo en cuenta los parámetros inerciales del satélite), se activan los actuadores adecuados, con la potencia adecuada durante el tiempo adecuado. Al conjunto de sensores, actuadores y algoritmos se le denomina sistema de “Guía, Navegación y Control”.

# Capítulo 2

## Bases de datos

### 2.1. Introducción

La base de datos con la que se trabajará en este proyecto de ingeniería concurrente es la misma para todos los subsistemas. En ella se guardan las variables necesarias. Cada variable tiene un sistema propietario y sólo él puede cambiar su valor. El resto de sistemas puede leer su valor previa autorización y utilizarlo para los cálculos propios. El presente proyecto es propietario de las variables concernientes al control de actitud. Por ejemplo la masa de los actuadores y sensores, momento de inercia del sistema, o rangos térmicos de los componentes. Además de la base de datos citada, este subsistema utiliza una base de datos independiente donde se guardan catálogos de componentes (ruedas, propulsores y sensores) y la selección del operario así como su ubicación y orientación sobre el satélite. Con esta segunda base de datos se calculan los parámetros que se pasarán al resto de los sistemas. Para el desarrollo del programa la segunda base de datos se ha instalado sobre un sistema de Microsoft Access, pero teniendo en cuenta durante todo el proyecto que esta base de datos se pueda trasladar a la misma plataforma que la base de datos principal. La conexión a ambas bases de datos se realiza utilizando los estándares ODBC (Open DataBase Connectivity). Todos los programas se encuentran en el CD que acompaña a este documento.

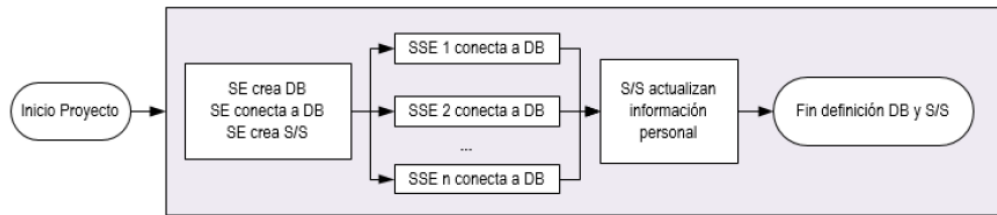


Figura 2.1: Diseño de base de datos en CDF

## 2.2. Creación

### 2.2.1. CDF

La base de datos de variables del satélite está en una plataforma MySQL y su funcionamiento interno está diseñado por personal del IDR. El grupo de alumnos en prácticas ha definido las variables necesarias y cada participante solicita las variables que necesita de los demás sistemas y es propietario de las que él es responsable. La forma de crear una base de datos se muestra en la figura 2.1. La documentación se extrajo de la referencia [3].

Cada usuario pide las variables que necesita a otros subsistemas, con unidades. El subsystem engineer se encarga de coordinar el proceso y da el visto bueno a las peticiones. Una vez que el SE aprueba las peticiones, éstas llegan al subsistema correspondiente, que las puede aceptar o rechazar. Si son aceptadas, pasan a ser visibles por los subsistemas, el solicitante (input) y el propietario (output). También será visible por los demás usuarios por si necesitan pedirla. Una vez se cierra la creación de la base de datos, se considera *Issue* 1 dispuesta para trabajar. Si se desea añadir a posteriori más variables, se creará una nueva *Issue*. La explicación de la *Issue* y version se realizará en el apartado siguiente.

### 2.2.2. Base de datos particular del módulo

La base de datos particular se ha hecho en Access y tan sólo contiene seis tablas.

- Catálogo de componentes. Cada tabla tiene los datos importantes para seleccionar el componente deseado.

- Sensores
  - Propulsores
  - Ruedas
- Componentes elegidos y situados sobre en el satélite. Cada tabla tiene los datos importantes para los output del satélite (coordenadas, peso, etc.).
- S\_colocado
  - P\_colocado
  - R\_colocada

Estas mismas tablas podrían crearse en otra base de datos (por ejemplo la propia del IDR) y los programas seguirían funcionando igualmente una vez configurado en el ordenador el conector ODBC correspondiente, como se indica en 2.4.2.

## 2.3. Issue/versión

La parte accesible de la base de datos de ingeniería concurrente para cada uno de los subsistemas consta de una tabla y una vista. La tabla se llama *var\_record* y sus registros son las variables que utilizan los módulos de cálculo [3]. Así los campos funcionales más significativos son:

**varName** Nombre de variable

**varValue** Valor de la variable

**varUnits** Unidades de la variable

**ssName** Propietario de la variable

**varIssue** Issue de la variable

**varVersion** Version de la variable

Obsérvese que no hay tipo input/output sino subsistemas propietarios. Para que esta base de datos pueda ser compartida de una forma ordenada existen dos campos fundamentales. *Issue* y *Version*, definidas dentro de la tabla como *varIssue* y *varVersion*. Se hace un cambio de Issue cuando hay un cambio sustancial en la base de datos. Como puede ser añadir un subsistema o una variable. O se decide congelar una configuración determinada. Se hace un cambio de *Version* cuando se modifica el valor de la variable. Cada variable tiene su *version*)

Nótese que se crea una nueva issue desde el administrador de la base de datos principal, mientras que el cambio de versión lo llevará a cabo cada uno de los módulos cuando guarde sus resultados. Cuando se comienza el diseño de un satélite se le asigna una *Database*. En el presente proyecto la *Database* se llama *cdf\_01*. Los usuarios de esta Database son los subsistemas (incluido el Subsystem engineer SE). Esta estructura la crea el administrador de la base de datos. Los usuarios (responsables de subsistemas) crean las variables que necesitan con los campos funcionales mencionados anteriormente. Todas las variables tendrán el mismo valor de *varIssue* y *varVersion*, empezando por uno y cero respectivamente. Una vez se hayan definido todas las variables y estén aprobadas por todos los subsistemas se procede al diseño del satélite. Para realizar sus cálculos cada módulo debe leer el conjunto de variables que le atañen correspondientes a la última versión existente de cada variable de la Issue en curso. Esta versión se le llama *varUsedVersion*. Si un módulo al realizar sus cálculos como resultado debe modificar el valor de una de sus variables la guardara con el siguiente número de versión. Como varios subsistemas pueden estar trabajando simultáneamente, es posible que uno grave antes que otro, creando una nueva versión. Por lo tanto antes de grabar hay que volver a leer la base de datos para verificar que la última versión y valor existente de cada variable ajena (*varPubVersion* y *varPubValue* de la variable tipo input) coincide con la versión que se leyó antes de arrancar el módulo (*varUsedVersion* y *varUsedValue*). Pueden darse varios casos de incoherencia:

- Para la base de datos:
  - La Issue actual no coincide con la Issue usada. Es un error grave en la base de datos.
- Para las variables de otros subsistemas (input):

- Si la versión coincide pero el valor no. Hay un error en la base de datos. Probablemente un subsistema no ha programado correctamente su módulo de exportación de variables.
- La versión no coincide pero el valor sí. También es una incoherencia pero no provoca resultados incoherentes.
- No coinciden ni la versión ni el valor de una variable. En este caso si guardásemos los cálculos realizados podrían estar desacoplados con el resto de los subsistemas. Habría que recalcular con los nuevos valores antes de grabar, y volver a verificar coherencia para poder guardar.
- Todas las variables input coinciden en valor y versión con las leídas al principio del cálculo. Datos coherentes, se pueden grabar nuestros resultados aumentando el número de versión de aquellas variables que hayan cambiado.

En el presente presente proyecto el programa que se encarga de grabar en la base de datos haciendo estas comprobaciones se llama *Grabar\_BD*. El programa se encuentra en el CD adicional.

## 2.4. Módulos de conexión

### 2.4.1. CDF

Para que el módulo desarrollado en Matlab tenga acceso a las bases de datos se han instalado los drivers ODBC del motor empleado (MySQL). La configuración para este conector es la que se muestra en la figura 2.3.

Desde Matlab se comprueba que este conector funciona ejecutando la toolbox Querybuilder. Esta toolbox permite acceder al contenido de la base de datos, como se muestra en la figura 2.4.

Realmente el módulo de cálculo no emplea el toolbox Querybuilder, sino las funciones individuales de manejo de bases de datos propias de Matlab.

### 2.4.2. Base de datos particular del módulo

Además de la base de datos de la ingeniería concurrente se dispone de otra pequeña base de datos donde se guardan los catálogos de los componentes y los elementos seleccionados para el satélite en cuestión así como sus puntos

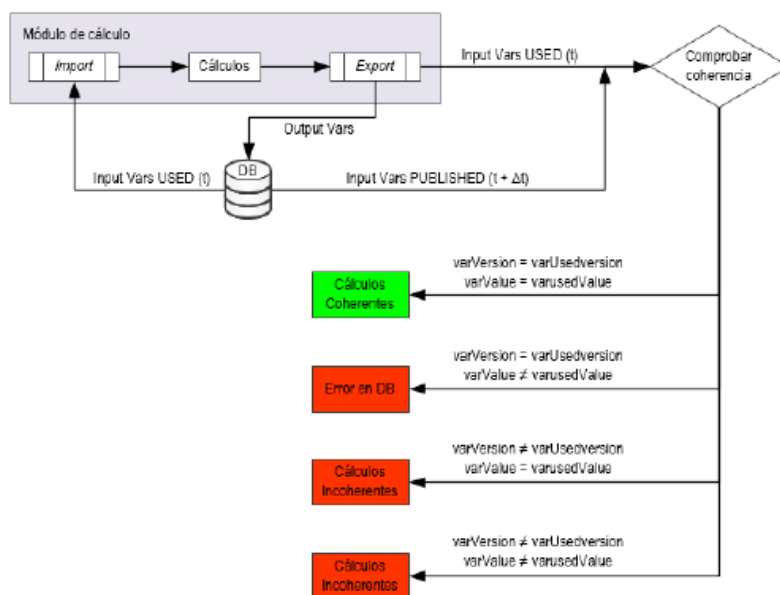


Figura 2.2: Algoritmo para la comprobación de coherencia del *Issue* y *version*

de colocación y orientación. Esta base de datos es un fichero Access con su driver ODBC instalado, pero podría ser cualquier otro motor (Oracle, SQL Server, MySQL, etc.) , siendo la configuración la que se muestra en la figura 2.5.

## 2.5. Programación CDF

### 2.5.1. Import

Para tener acceso a las variables del módulo de ingeniería concurrente se ha escrito una función que se denomina *Import\_DB\_3*. Esta función es llamada por el primer interfaz para cargar las variables disponibles. Esta función se conecta utilizando el driver ODBC utilizando la instrucción:

```
cn=database('IDR','','')
```



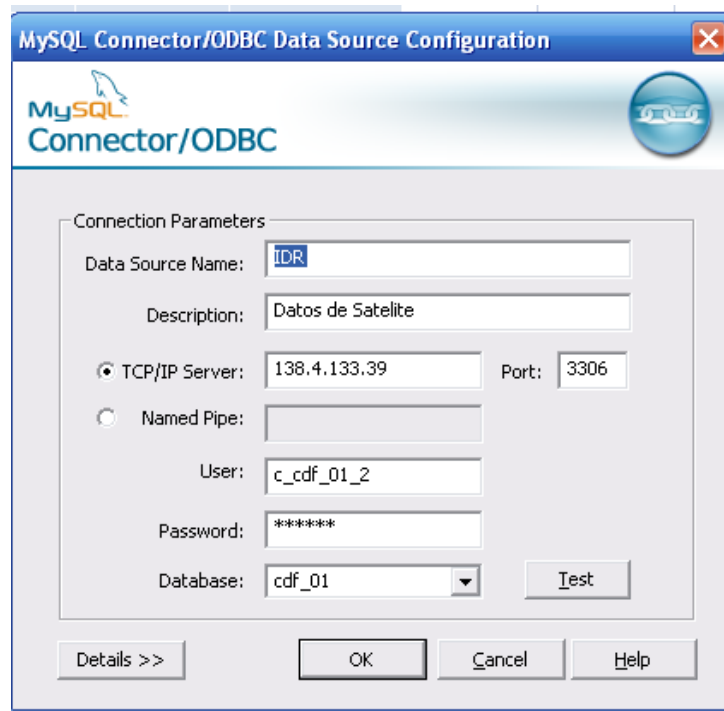


Figura 2.3: Configuración de driver de conexión

Se recuperan todas las variables disponibles para el módulo de actitud (llamado andrea) con las instrucciones:

```
curs=exec(cn,'select * from andrea')
curs=fetch(curs)
datos=curs.data
```

La variable *datos* se ha creado como un *cell\_array*. Cada fila de este *cell\_array* corresponde a una variable para el módulo. A continuación, se crean dos estructuras de datos:

- La estructura *s1* tiene como campos hijos los nombres de las variables y éstos contienen su valor numérico.

Por ejemplo, una variable queda almacenada como

```
s1.Velocity = 7.38
```

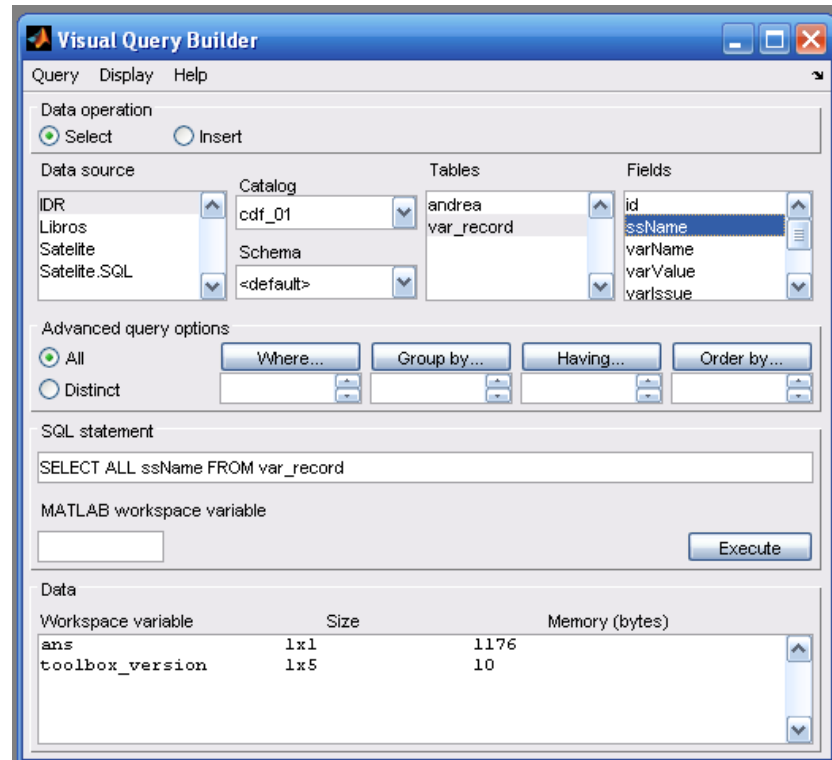


Figura 2.4: Comprobación de conexión a la base de datos con la toolbox de Matlab

- La estructura s2 tiene dos ramas, correspondientes al tipo de variable respecto al módulo.
  - Input
  - Output

Cada una de estas ramas tiene como hijos los nombres de las variables correspondientes, y cada una de estas variables tiene como hijos

- Valor
- Issue
- Version

Por ejemplo, la misma variable queda almacenada como

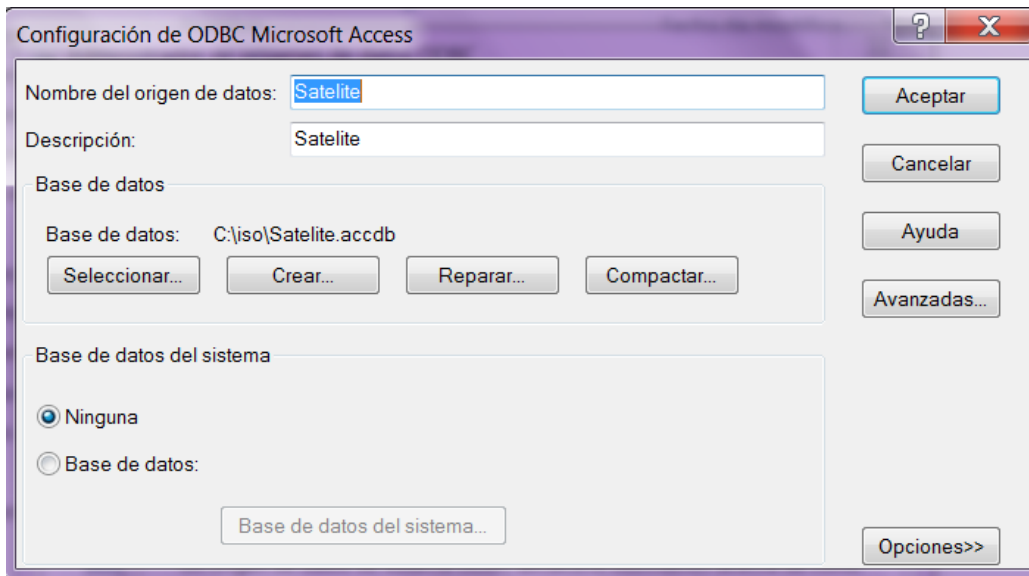


Figura 2.5: Configuración ODBC para la base de datos particular

```
s2.input.Velocity.Valor=7.38
s2.input.Velocity.Issue=1
s2.input.Velocity.Version=0
```

La función completa se muestra en el listado 2.1. La función devuelve las dos estructuras *s1* y *s2*.

### 2.5.2. Export

Para grabar los datos al final del diseño hay que seguir el diagrama de flujo explicado en 2.3. Como se explica en el apartado de programación de los interfaces 5.2.1, las variables se pasan de un programa a otro como hijos de la estructura *handles*. De esta manera se pasan tres estructuras:

- s** Corresponde a la estructura *s1* de la función *Import\_DB*. Contiene los nombres de las variables y sus valores.
- sc** Corresponde a la estructura *s2* de la función *Import\_DB*. Contiene el tipo de variable, el nombre, el valor, la *Issue* y la *Version*.
- p** Contiene las variables empleadas en el programa. Algunas son las mismas de *s*, con el nombre cambiado y otras son nuevas. Por ejemplo

Listing 2.1: Función que importa los datos de la base de datos concurrente

```

1 function [s1 s2] = ImportDB_3()
2 %Esta funcion trae de la base de datos las variables.
3 %Devuelve dos estructuras,
4 % s1 con todos los datos y
5 % s2 con la misma informacion, pero separadas
6 %por 'input' y 'output' con los valores de Issue y version
7
8 %quitar estas instrucciones cuando hay conexion de verdad
9 %load('datos_satelite.mat','s1');return;
10
11 cn = database('IDR','','');
12 curs = exec(cn,'select * from andrea');
13 curs = fetch(curs);
14
15 datos = curs.Data;
16
17 [r c] = size(datos);
18 close(curs);
19 close(cn);
20
21 for i=1:r
22     tipo = datos{i,2}; %input /output
23
24     variable = datos{i,4} ;
25     valor = str2num(datos{i,5});
26     Issue = datos{i,6};
27     Version = datos{i,7};
28
29     s1.(variable) = valor;
30     %
31     s2.(tipo).(variable).Valor = valor ;
32     s2.(tipo).(variable).Issue = Issue ;
33     s2.(tipo).(variable).Version = Version;
34 end
35
36 %Variables a la espera de asignación por el SE
37 tipo = 'input';variable = 'HeightCube';
38 valor = .6;Issue = 1;Version = 0;
39
40 s1.(variable) = valor;
41 s2.(tipo).(variable).Valor = valor ;
42 s2.(tipo).(variable).Issue = Issue ;
43 s2.(tipo).(variable).Version = Version;
44
45 %Guardo los datos para cuando vaya a ejecutar
46 %sin conexion a la base dedatos.
47 save('datos_satelite.mat','s1');
48
49 end

```

```
s.velocity = p.v
```

A lo largo del programa las variables de esta estructura cambian.

En el último interfaz se ha habilitado un botón para grabar. Este botón llama a la función *Grabar\_DB* que se encarga de verificar que se cumplan las condiciones de compatibilidad y de grabar en la base de datos si se cumplen.

Esta función recibe las estructuras *p* y *sc*. Lo primero que hace es conectarse a la base de datos y obtener la estructura correspondiente a las variables en el momento actual. De las dos estructuras que devuelve la función *Import\_DB* sólo se queda con la 'completa' y la denomina *scnew*.

De la estructura *sc* se extraen los nombres de las variables tipo *input*. Se comprueba para cada una que en las estructuras *sc* y *scnew* tienen los mismos valores de *Issue*, *Version* y *Valor*. En caso de que haya una discrepancia, se da mensaje de error y se detiene el proceso de grabación. Con esto se ha comprobado que durante la ejecución del módulo los dueños de las variables ajenas no las hayan cambiado.

En un segundo paso, se comprueba que durante el diseño no se ha modificado una variable de la estructura *p* que esté vinculada a una variable de la estructura *s* y que sea *input*. Para ello, para cada variable *input* de la estructura *s* se busca su equivalente de la estructura *p*. Este mapeo se realiza con la función *NombreV*, que se muestra en el listado 2.2. Se comprueba que los valores son iguales y si no es así se da el mensaje de error y se detiene el proceso.

Por último se busca en la estructura *sc* las variables *output*. Para cada una de ellas se busca su equivalente en la estructura *p* con la misma función *NombreV*. Si los valores son diferentes, hay que grabar el nuevo valor. Si son iguales, no se hace nada con esta variable. Para ello, el programa realiza los siguientes pasos (Para el caso de *Issue* 1, *version* 0 y variable *Velocity*):

1. Se conecta a la base de datos y extrae los nombres de los campos de la tabla *var\_record* (este paso solo se ejecuta una vez para todas las variables).

```
cn = database('IDR','','')
colnames=columns(cn','','','var_record')
```

2. Extrae el registro correspondiente a esta *Issue*, *Version* y variable :

Listing 2.2: Función de mapeo entre variables

```

1 function [ NombreP ] = NombreV( NombreBD )
2 %Con esta función se traduce el nombre de una variable en
3 %base de datos con el nombre utilizado en este programa
4
5 switch NombreBD
6     case 'OrbitAltitudeCircular'
7         NombreP = 'h';
8     case 'MaximumDeviation'
9         NombreP = 'theta_m';
10    case 'Velocity'
11        NombreP = 'v';
12    case 'ResidualDipole'
13        NombreP = 'D1';
14    case 'Iz'
15        NombreP = 'MIz';
16    case 'Imin'
17        NombreP = 'MIy';
18    case 'SuperficieFrontal'
19        NombreP = 'A';
20    case 'Cd'
21        NombreP = 'Cda';
22    case 'GravityCenter'
23        %NombreP = 'cg';
24        NombreP = 'dummy';
25
26    case 'PointingAccuracy'
27        NombreP = 'theta_a';
28    case 'Period'
29        NombreP = 'P';
30    case 'Burntime'
31        NombreP = 'tb';
32    case 'Isp'
33        NombreP = 'Isp_v';
34
35    case 'ActitudeControlWeight'
36        NombreP = 'masa';
37    otherwise
38        %Variable no modificable en el programa
39        NombreP = 'dummy';
40 end
41
42
43 end

```

```
st=['select * from var_record where varIssue=1...  
    and varVersion=0 and varName="Velocity" ']  
curs=exec(cn,st)  
curs=fetch(curs,1)  
nuevo=curs.Data  
close(curs)
```

3. Cambia el valor viejo por el valor nuevo

```
nuevo{varValue} = 8.1
```

4. Cambia el valor de la versión

```
nuevo{varVersion} = 0+1
```

5. Inserta el nuevo registro

```
insert(cn,'var_record', colnames(2:end),nuevo(2:end))
```

Nuevo es un *cell\_array* con los datos de la variable que se quiere grabar.

6. Cuando se procesan todas las variables se cierra la conexión a la base de datos

```
close(cn)
```





## Capítulo 3

# Conclusiones de la Parte I

### 3.1. Cumplimiento de objetivos

Esta parte del proyecto tenía como objetivos la conexión de un programa externo (Matlab) a la base de datos del CDF de forma coherente con el resto de los módulos, respetando la integridad de los datos. Este objetivo se ha conseguido satisfactoriamente y sin mayor complicación aprovechando las ventajas de ODBC. Los estándares ODBC son accesibles desde todas las plataformas y para todos los motores de base de datos. La programación usando estos estándares es sencilla utilizando un lenguaje unificado muy fácil de implementar. Los programas se pueden consultar en el anexo I.

Además se ha diseñado una base de datos particular para guardar los catálogos y para realizar los cálculos. Esta base de datos está montada sobre Microsoft Access pero la forma de programación es idéntica a la anterior. El código de estos programas se encuentra en el CD del proyecto.

### 3.2. Trabajos futuros

La base de datos particular se puede ampliar añadiendo más información a las tablas existentes y añadiendo otros catálogos, como puede ser magnetopares. Una mejora evidente sería incorporar las tablas de la base de datos particular a la base de datos general, ahorrando así mantenimiento y compactando el sistema.



# Parte II

## Diseño de prefase-A



# Capítulo 4

## Perturbaciones

### 4.1. Concepto teórico

Uno de los primeros pasos para el diseño del sistema de control de actitud del satélite será el estudio de fuerzas perturbadoras y sus efectos sobre el movimiento de la órbita. Se tratarán aquellas perturbaciones que se manifiesten en variaciones seculares lineales (variaciones de corto periodo). Su periodo será menor que el de la órbita, de modo que afecte de forma continua a los parámetros orbitales (incrementándolos o disminuyéndolos) [11]. A continuación se analizarán las siguientes perturbaciones: solar, aerodinámica, gradiente de gravedad y magnética. El modelo que se ha escogido para estimar estas perturbaciones se encuentra en la referencia [16].

Puesto que este capítulo sirve para dimensionar los actuadores del satélite, no se necesita calcular con exactitud cada uno de ellos sino que se hace una estimación por separado y se aplica el criterio conservativo de sumar todos los máximos de pares para obtener un par total, aún sabiendo que no se dan simultáneamente ni en la misma dirección.

$$T_D = T_{sp} + T_m + T_a + T_g. \quad (4.1)$$

Donde los pares son los calculados en las ecuaciones (4.2), (4.3), (4.4) y (4.5) respectivamente.

### 4.1.1. Perturbación solar

Esta perturbación se debe a la colisión de fotones sobre la superficie del satélite. Es una fuerza que lleva la dirección del haz solar. Al cabo de una vuelta completa del satélite el trabajo realizado es nulo. Sin embargo, si el centro de la superficie proyectada del satélite no está alineado siempre con el centro de masas se genera un par, que deberá ser compensado por el sistema de actitud. En este apartado se estima el par que genera esta presión solar.

■ Input:

- Datos de órbita

$P_{\text{sun}}$  Presión solar

- Datos del satélite

$A_s$  Geometría del satélite

$q$  Reflexión de la superficie (se encuentra entre 0 y 1)

$c_g$  Posición del centro de gravedad

$c_{ps}$  Posición del centro de presión solar

$i$  Ángulo de incidencia solar

- Otros datos

$c$  velocidad de la luz

■ Output:

$T_{sp}$  Par Solar

$$T_{sp} = \frac{P_{sun}}{c} A_s (1 + q) \cos(i) * (c_{ps} - c_g). \quad (4.2)$$

### 4.1.2. Perturbación debido al campo magnético

El campo magnético que rodea a la Tierra también genera pares si el satélite presenta algún dipolo. Los circuitos eléctricos pueden hacer que este dipolo exista con lo cual el satélite se ve interferido. Un buen aislamiento de los cables ayuda a minimizar esta interferencia.

- Input:

- Datos de órbita
  - h** Altitud de la órbita
- Datos del módulo de potencia
  - D** Dipolo magnético residual del satélite
- Otros datos
  - M** campo magnético característico de la Tierra
  - R** radio terrestre

- Output

**T<sub>m</sub>** Par magnético

$$T_m = D \frac{2M}{(R + h)^3}. \quad (4.3)$$

### 4.1.3. Perturbación aerodinámica

La atmósfera crea fricción sobre todo cuerpo que se mueva en su seno. Aunque la densidad sea muy baja a grandes alturas, la alta velocidad en la que se mueven los satélites hace que este efecto no sea despreciable.

- Input:

- Datos de órbita
  - $\rho$**  Altitud de órbita
  - v** Velocidad del satélite
- Datos del satélite
  - A** Máxima área proyectada
  - c<sub>g</sub>** Posición del centro de gravedad
  - c<sub>pa</sub>** Centro de presiones aerodinámico
  - c<sub>D</sub>** Coeficiente de resistencia aerodinámica

■ Output

$T_a$  Par aerodinámico

$$T_a = \frac{1}{2} \rho A v^2 C_D * (C_{pa} - c_g). \quad (4.4)$$

#### 4.1.4. Perturbación por gradiente de gravedad

Cuando se realizan cálculos de fuerzas y momentos cerca de la superficie terrestre, se considera el valor de la gravedad terrestre  $g$  constante para todos los puntos. Se sabe que la fuerza gravitatoria es inversamente proporcional al cuadrado de la distancia. La diferencia de distancias al centro de la Tierra de un punto a otro se considera despreciable frente a la distancia total, entonces el peso no genera pares. Sin embargo, en un satélite unos pares muy pequeños aplicados el tiempo suficiente pueden desviar la orientación del mismo. Por ello se calcula el par generado por la diferencia de gravedades en los diferentes puntos del satélite. Dicho par se emplea en los satélites cuya orientación se desea que sea apuntando al nadir terrestre para estabilizarlos. Si se desea un satélite autoestabilizado por gradiente de gravedad y que apunte a nadir, conviene que su momento respecto al eje  $z$  sea mucho más pequeño que los demás (satélite con forma de lapicero).

■ Input:

- Datos de órbita

$h$  Altitud de órbita

- Datos del satélite

$I_z$  Menor momento de inercia de los tres principales

$I_y$  Mayor momento de inercia de los tres principales

- Otros datos

$\mu$  Constante gravitacional terrestre ( $GM_T$ )

$R$  Radio de la Tierra

$\theta$  Ángulo de separación de la vertical

■ Output



$T_g$  Par gradiente de gravedad

$$T_g = \frac{3\mu}{2(R+h)^3} |I_z - I_y| \sin(2\theta). \quad (4.5)$$

## 4.2. Programación

Para realizar la estimación del conjunto de perturbaciones concerniente a este satélite se ha desarrollado una interfaz gráfica con el toolbox GUIDE de Matlab. (Figura 4.1)

Figura 4.1: Interfaz gráfica de perturbaciones

Con este interfaz se han insertado paneles con datos (datos órbita y datos satélite) y un panel con resultados (perturbaciones).

Lo primero que hace el interfaz al cargarse es conectarse a la base de datos de ingeniería concurrente y traerse las variables con sus valores y su número de *Issue* y *version*. Añadimos todas las variables a la estructura de *handles* del cuadro de diálogo para que estén disponibles en cualquier momento. En *handles.sc* se encuentran todas las variables con sus valores, Issues y versions. En la *handles.s* se encuentran todas las variables y sus valores.

### 4.2.1. Función `recupera_variables`

La función `recupera_variables` (Listado 4.1) se encarga de crear una estructura de variables independiente de la que se lee de la base de datos y con algunas variables mapeadas. En esta estructura se puede modificar el valor de todas las variables desde el interfaz. Esa estructura se añade también a la estructura de `handles` para que esté disponible en todo momento (`handles.p`). De este modo se deja libre al usuario que modifique incluso valores de variables heredadas que usará para crear sus propias situaciones. Sin embargo, si modifica alguna de estas variables que no pertenecen al subsistema entonces el control de coherencia no le dejará guardar los resultados. El resto de valores no heredados se completan con *Nan* (Not a number), para obligar al usuario a introducirlos.

### 4.2.2. Función `dato_r`

Para asegurar que el usuario introduce datos numéricos se ha creado una función llamada `dato_r` (Listado 4.2). Esta función recibe el `handle` del objeto en cuestión (que es la casilla de texto que rellena el usuario). De esta casilla toma el valor introducido, las unidades de esa variable, y comprueba que el dato es numérico. En caso afirmativo reformatea la casilla escribiendo la variable seguida de las unidades. En caso negativo, rellena la casilla con *Nan*. Las unidades de cada campo de pantalla se guardan en la propiedad `TooltipString`. (Dentro de Property inspector).

Todos los campos de introducción de datos de la pantalla llaman a la función `dato_r` y guardan el resultado numérico dentro de la estructura de `handles.p`. De esta forma la variable está disponible en cualquier momento durante la ejecución del cuadro.

### 4.2.3. Función `dato_w`

Para agilizar la escritura de los resultados en pantalla se ha desarrollado una función llamada `dato_w` (Listado 4.3). Esta función recibe el `handle` del campo y el valor que se quiere escribir y lo escribe añadiendo el sufijo (las unidades). Esta función se encarga de discernir si el dato es numérico o no y actúa en consecuencia.

Listing 4.1: Función recupera\_variable

```

1 function [p] = recupera_variables(handles)
2 %Invocamos a las variables globales y recuperadas
3 %de la base de datos
4 p.h = handles.s.OrbitAltitudeCircular;
5 p.theta_m = handles.s.MaximumDeviation;
6
7 p.v=handles.s.Velocity;
8 p.M1 = NaN;
9 p.i = NaN;
10 p.cs = NaN;
11 p.D1 = handles.s.ResidualDipole;
12
13 p.MIz = handles.s.Iz;
14 p.MIy=handles.s.Imin;
15 p.A=handles.s.SuperficieFrontal;
16 p.Cda = handles.s.Cd;
17 p.Cpa = NaN;
18 p.cg = max(handles.s.GravityCenter(1:2));
19 p.q = NaN;
20 p.Ta = NaN;
21 p.Tm = NaN;
22 p.Tsp = NaN;
23 p.Tg = NaN;
24 p.Td = NaN;
25 dato_w(handles.h, p.h);
26 dato_w(handles.theta_m,p.theta_m);
27 dato_w(handles.v, p.v);
28 dato_w(handles.M1, p.M1);
29 dato_w(handles.i, p.i);
30 dato_w(handles.cs, p.cs);
31 dato_w(handles.D1, p.D1);
32 dato_w(handles.MIz, p.MIz);
33 dato_w(handles.MIy, p.MIy);
34 dato_w(handles.A, p.A);
35 dato_w(handles.Cda, p.Cda);
36 dato_w(handles.Cpa, p.Cpa);
37 dato_w(handles.cg, p.cg);
38 dato_w(handles.q, p.q);
39 dato_w(handles.Ta, p.Ta);
40 dato_w(handles.Tm, p.Tm);
41 dato_w(handles.Tsp, p.Tsp);
42 dato_w(handles.Tg, p.Tg);
43 dato_w(handles.Td, p.Td);

```

Listing 4.2: Función dato\_r

```
1 function r = dato_r(hObject)
2 %LEE DATO numerico DE LA PANTALLA
3 %lo interpreta como número y le vuelve a colocar añadiendo
4 %las unidades.
5
6 dato = get(hObject,'String');
7 sufijo = get(hObject,'TooltipString');
8 r = str2double(dato);
9 if isnan(r)
10     dato = 'NaN';
11     sufijo = '';
12 else
13     dato = num2str(r);
14 end
15 set(hObject,'String',[dato ' ' sufijo])
16 end
```

Listing 4.3: Función dato\_w

```
1
2 function dato_w(hObject,dato)
3
4 %Escribe dato en campo text de la pantalla
5
6 %Toma el texto de la Propiedad 'TooltipString'
7 %y lo pone como sufijo
8 %util para añadir las unidades
9
10 sufijo = get(hObject,'TooltipString');
11 if ischar(dato)
12     set(hObject,'String',[dato ' ' sufijo])
13 else
14     set(hObject,'String',[num2str(dato) ' ' sufijo])
15 end
```

#### 4.2.4. Botones del cuadro de diálogo

Los botones del cuadro de diálogo sirven para calcular los diferentes pares. Para ello llaman a las funciones correspondientes explicadas en el apartado teórico 4.1 en la página 45.

Antes de llamar a una función se verifica por programa que todos los argumentos necesarios están cumplimentados. En caso de que falte alguno se muestra un *pop-up* indicando el error y el campo correspondiente se cumple con la palabra *Error*. Hasta que no se resuelvan todos los errores no se llama a las funciones de cálculo (Figura 4.2).

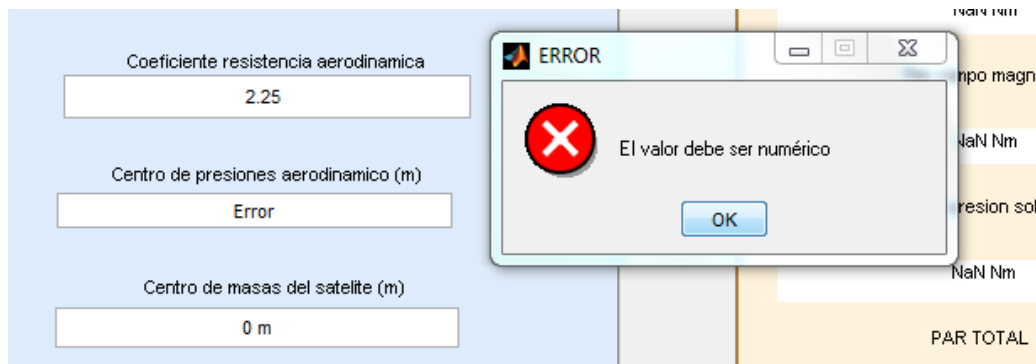


Figura 4.2: Mensaje de error

Una vez ejecutada la función el resultado también se guarda dentro de la estructura de *handles.p* y se muestra el resultado por pantalla en un campo de texto no editable (Figura 4.1).

El cuadro de diálogo tiene un botón para calcular cada uno de los pares de perturbación. Hay un quinto botón que calcula el par total (que verifica que se hayan calculado los pares anteriores). Una vez hecho este cálculo se habilita el botón para pasar al siguiente programa (Actuadores sin gas). Para pasar los datos de un programa al siguiente se cargan en la matriz *user.data* los handles *handles.s*, *handles.p* y *handles.sc* (Listado 4.4).

Recuérdese que toda la documentación acerca de la programación se encuentra en el CD adjunto.

Listing 4.4: Traspaso de variables de un interfaz a otro

```

1
2 %*****%
3 %          BOTONES DE SALIDA          %
4 %*****%
5
6
7
8 % --- Executes on button press in g_RW_MW_MT.
9 function g_RW_MW_MT_Callback(hObject, eventdata, handles)
10 % hObject    handle to g_RW_MW_MT (see GCBO)
11 % eventdata  reserved - to be defined in a future version
    of MATLAB
12 % handles    structure with handles and user data (see
    GUIDATA)
13
14 RW_MW_MT('UserData',{handles.s handles.p, handles.sc});
15
16 delete(gcf);

```

# Capítulo 5

## Actuadores sin combustible

### 5.1. Conceptos teóricos

#### 5.1.1. Ruedas de inercia

Las ruedas de inercia suelen ser dos, una para el eje x y otra para el eje y.

Almacenan el momento inducido por las perturbaciones. Dicho momento se acumula y hay que disiparlo antes de llegar a la saturación. Los siguientes datos se piden al usuario, se obtienen de cálculos anteriores o se extraen de la base de datos.

- Input

- Datos de misión

- P** Período de la órbita (seg)

- $\theta_a$**  Ángulo permitido de precisión (deg)

- $T_D$**  Par de perturbaciones (Ecuación (4.1))

- Output

- $H_{mw}$**  Momento acumulado por la rueda de inercia (Ecuación (5.1))

Ecuación necesaria para el cálculo:

$$H_{mw} = T_D \frac{P}{4\theta_a}. \quad (5.1)$$

En algunos satélites se mantiene estabilizado el eje  $z$  haciendo girar dichos satélites alrededor de este eje. Es lo que se denomina estabilización por spin. Nótese que para tener este tipo de estabilización conviene que el momento de inercia respecto al eje  $z$  sea mayor que los demás y que sea principal de inercia.

■ Input

**H** Momento acumulado

**I** Momento de inercia respecto al eje  $z$

■ Output

**$\omega$**  Velocidad angular de giro del satélite alrededor del eje  $z$  (Ecuación (5.2))

Ecuación necesaria:

$$\omega = \frac{H}{I}. \quad (5.2)$$

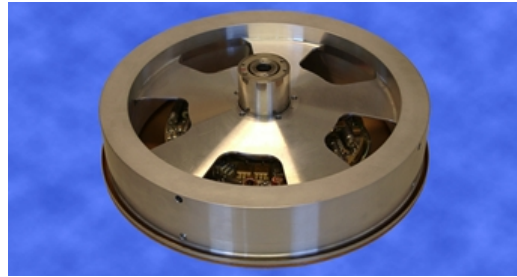


Figura 5.1: Rueda de inercia RSI45

### 5.1.2. Ruedas de reacción

El concepto es el mismo que las ruedas de inercia. Su función es provocar pares girando que hacen cambiar la orientación del satélite y poder así realizar maniobras.

■ Input



- Datos de misión

**P** Período de la órbita (s)

**$\theta$**  Ángulo de maniobra (deg)

**$T_D$**  Par de perturbaciones (Ecuación (4.1))

**t** tiempo de maniobra (seg)

- Output

**$T_{rw}$**  Par necesario para realizar una maniobra (Ecuación (5.3))

**$H_{rw}$**  Momento angular acumulado (Ecuación (5.4))

Ecuaciones necesarias para los cálculos:

$$T_{rw} = 4\theta \frac{I}{t^2}. \quad (5.3)$$

$$H_{rw} = \frac{T_D P}{4} * 0,707. \quad (5.4)$$

### 5.1.3. Pares magnéticos

Para órbitas bajas el campo magnético de la Tierra puede emplearse para orientar al satélite. Se consigue creando dipolos de forma intencionada. El mecanismo es hacer circular una corriente eléctrica por una espira (bobinado) orientada según un eje determinado. Para conseguir generar pares en cualquier dirección del espacio hace falta tres espiras orientadas en tres direcciones independientes. (Las direcciones típicas son xyz)

- Input:

- Datos de órbita

**h** Altitud de la órbita

- Datos del módulo de potencia

**T** Par de perturbación (Ecuación (4.1)) o maniobra

- Otros datos

**M** campo magnético característico de la Tierra

**R** radio terrestre

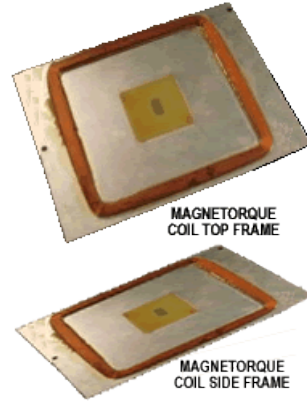


Figura 5.2: Pares magnéticos instalados en el satélite LUSEX

#### ■ Output

**D** Dipolo magnético necesario

$$D = \frac{T(R+h)^3}{2M}. \quad (5.5)$$

## 5.2. Programación

### 5.2.1. Interfaz de datos

Cuando se ha terminado con el cálculo de perturbaciones (sección 4.2), se habilita el botón de paso a actuadores sin combustible. Ver figura 5.3.

Recibe del programa anterior (subsección 4.2) los datos divididos en tres grupos. Por un lado los datos del satélite, que se guardan en la estructura *handles.s*. Por otro lado, los datos introducidos manualmente y los calculados que se guardan en la estructura *handles.p* y la copia de seguridad para la comprobación antes de guardar resultados en *handles.sc*. Ver listado 5.1.

Se ha definido una función *recupera\_variables* en la que se añaden a la estructura *handles.p* los datos nuevos (así como los nuevos cálculos) de este programa. También carga en pantalla todos los datos relevantes conocidos hasta el momento (par TDS) . Esto se muestra en el listado 5.2.

Este programa sólo pide dos datos nuevos por pantalla (factor de seguridad y tiempo de maniobra). La introducción y lectura de datos de la pantalla

Listing 5.1: Inicialización del cuadro de diálogo de actuadores

```

1  % --- Executes just before RW_MW_MT is made visible.
2  function RW_MW_MT_OpeningFcn(hObject, eventdata, handles,
   varargin)
3  % This function has no output args, see OutputFcn.
4  % hObject      handle to figure
5  % eventdata    reserved - to be defined in a future version
   of MATLAB
6  % handles      structure with handles and user data (see
   GUIDATA)
7  % varargin     command line arguments to RW_MW_MT (see
   VARARGIN)
8
9  % Choose default command line output for RW_MW_MT
10 handles.output = hObject;
11
12 recibido = get(hObject, 'UserData');
13
14 s = recibido{1};
15 p = recibido{2};
16 sc = recibido{3};
17
18 %Datos del satélite
19 handles.s = s;
20 handles.sc = sc;
21
22 %Datos de perturbaciones. Unas las recibimos y otras
23 %las asignamos en esta pantalla.
24 handles.p = p;
25
26 p = recupera_variables(handles);
27
28 handles.p = p;
29
30 % Update handles structure
31 guidata(hObject, handles);
32
33 % UIWAIT makes RW_MW_MT wait for user response (see
   UIRESUME)
34 % uiwait(handles.figure1);

```

Listing 5.2: Función para cargar variables en pantalla

```
1 function [p] = recupera_variables(handles)
2
3 %Vamos a añadir más variables a handles.p
4
5 %variable temporal
6 p = handles.p;
7
8 p.MS = NaN;
9 dato_w(handles.MS,p.MS);
10
11
12 dato_w(handles.theta_m,p.theta_m);
13
14
15 p.tm = NaN;
16 dato_w(handles.tm,p.tm);
17
18 p.P = handles.s.Period;
19 dato_w(handles.P,p.P);
20
21
22 p.theta_a = handles.s.PointingAccuracy;
23 dato_w(handles.theta_a,p.theta_a);
24
25
26
27 %set(handles.TDS,'String',TDS);
28 dato_w(handles.Td,sprintf('%5.3E',p.Td));
```

The screenshot shows a software window titled 'RW\_MW\_MT'. It contains three main panels:

- Datos\_actuadores (left, blue background):** Contains input fields for:
  - Factor de seguridad: NaN
  - Angulo de maniobra (\*): 30 grados
  - Tiempo de maniobra (min): NaN min
  - Periodo orbital (seg): 96164.0905 segundos
  - Angulo permitido (precision, \*): 0.01 grados
- Pares (middle, orange background):** Contains:
  - Par para contrarrestar perturbacion: 3.513E-006
  - TDS (Torque Distribution Strategy)
  - Par de maniobra rueda de reaccion
  - TRW (Torque Reaction Wheel)
  - Par para magnetopares (dipolo)
  - D2
- Momentos almacenados (right, orange background):** Contains:
  - Momento almacenado rueda reaccion: HRW
  - Momento almacenado rueda de inercia: IMW
  - Momento almacenado spinner: omega\_s

At the bottom, there are four buttons: 'Calcular pares', 'Seleccionar Ruedas', 'Calcular momentos almacenamiento', and 'Ir a Propulsores'.

Figura 5.3: Interfaz de actuadores

tiene la misma funcionalidad que el programa de perturbaciones. El cálculo se divide en dos fases, una para calcular pares y otra para calcular momentos. En el apartado de conceptos teóricos (sección 6.1) se especifican los cálculos realizados. Una vez calculado todo, se habilitan dos botones: *seleccionar ruedas*, que permite acceder al catálogo de ruedas (se explica en la subsección 5.2.2) y el botón de ir a propulsores que se explica en el siguiente capítulo, en la subsección 6.2.1.

### 5.2.2. Operaciones con el catálogo

En el cuadro de cálculo de fuerzas y momentos, una vez calculados, se habilita el botón de *seleccionar ruedas*. Al pulsar este botón se abre el catálogo. A este nuevo cuadro se le pasan los datos del satélite. El modo de pasarlo es utilizando la propiedad *userdata* de la figura.

La instrucción es:

```
Catalogo_ruedas('UserData',handles.s)
```

El cuadro recoge estos datos con la instrucción:

```
satelite=get(hObject,'UserData')
```

Lo añade a la estructura *handles* de este nuevo cuadro con la instrucción:

```
handles.s = satelite
```

El contenido del catálogo se ha introducido previamente en la base de datos con las herramientas que ésta dispone en una tabla llamada “ruedas”. En este cuadro de diálogo se muestra una tabla con el contenido de la base de datos (ver figura 5.4). Se pueden manipular los datos para crear nuevas ruedas basándose en las existentes. Para diferenciar las ruedas reales de las “inventadas”, se ha habilitado un campo lógico en la base de datos que es cierto (*true*) para las ruedas reales.

Id	Comercial	Nombre	Momento	Par	Consumo_max	Consum_vacio	Velocidad	Masa	Diametro	Grosor
<input type="checkbox"/>	<input checked="" type="checkbox"/>	HR12-1	12	0.2000	195	22	6000	6	317	146
<input type="checkbox"/>	<input checked="" type="checkbox"/>	HR12-2	25	0.2000	195	22	6000	7	317	146
<input type="checkbox"/>	<input checked="" type="checkbox"/>	HR12-3	50	0.2000	195	22	6000	9.5000	317	146
<input type="checkbox"/>	<input checked="" type="checkbox"/>	HR14-1	25	0.2000	195	22	6000	7.5000	368	146
<input type="checkbox"/>	<input checked="" type="checkbox"/>	HR14-3	75	0.2000	195	22	6000	10.6000	368	146
<input type="checkbox"/>	<input checked="" type="checkbox"/>	HR16-1	50	0.2000	195	22	6000	9	412	151
<input type="checkbox"/>	<input checked="" type="checkbox"/>	HR16-2	75	0.2000	195	22	6000	10.4000	412	151
<input type="checkbox"/>	<input checked="" type="checkbox"/>	HR16-3	100	0.2000	195	22	6000	12	412	151
<input type="checkbox"/>	<input checked="" type="checkbox"/>	MSCI-1000	1	0.0300	7	2	10000	1.4000	130	90
<input type="checkbox"/>	<input checked="" type="checkbox"/>	MSCI-200	0.2500	0.0300	7	2	10000	1	100	90
<input type="checkbox"/>	<input checked="" type="checkbox"/>	HR12-3	150	0.2000	195	22	6000	9.5000	317	146
<input type="checkbox"/>	<input checked="" type="checkbox"/>	HR14-1	25	0.3000	195	22	6000	7.5000	368	146
<input type="checkbox"/>	<input checked="" type="checkbox"/>	MSCI-1000	1	0.0300	7	2	10000	1.4000	130	90
<input type="checkbox"/>	<input checked="" type="checkbox"/>	HR12-3	151	0.2000	195	22	6000	9.5000	317	146
<input type="checkbox"/>	<input checked="" type="checkbox"/>	HR12-3HR1...	87.5000	0.2000	195	22	6000	8.5000	342	146
<input type="checkbox"/>	<input checked="" type="checkbox"/>	HR12-1	12	0.2000	195	22	6000	6	317	146
<input type="checkbox"/>	<input checked="" type="checkbox"/>	HR12-1HR1...	18.5000	0.2000	195	22	6000	6.5000	317	146

Figura 5.4: Catálogo de ruedas

La función *Tabla\_db (nombre\_de\_tabla)*, cuya programación se puede ver en el listado 5.3, devuelve los datos de la tabla, los nombres de los campos y los identificadores de los registros. Estos identificadores son útiles para modificar o eliminar un registro de la base de datos. En el diseño de la tabla ‘ruedas’ se ha incluido un campo autonumérico que sirve de índice para las operaciones. Este campo no se muestra por pantalla. En su lugar se pone una casilla para marcar a la hora de realizar operaciones sobre el registro. Sobre los registros del catálogo se pueden realizar diversas operaciones seleccionándolos y pulsando los botones correspondientes. Para seleccionar registros lo

que se hace es activar el primer checkbox del registro. Para ello se programa en el evento *uitable1\_CellSelectionCallback*. En esta parte de código primero se verifica la fila y columna sobre la cual se ha hecho click con el ratón, y si se trata de la primera o segunda columna, se extrae el valor lógico de la primera (true o false) y se invierte.

A continuación se explica la funcionalidad de cada botón:

**Mezclar** Crea una rueda nueva a partir de las ruedas seleccionadas. Los campos numéricos se calculan como media aritmética y los alfanuméricos se concatenan. Lo primero que hace esta función es sacar a la variable *datos* el contenido de la tabla de la pantalla. Hace el vector *columna* a con los valores true/false de los registros no marcados y marcados. La variable *marcados* se llena con el número de registros marcados (2 si se mezclan 2 ruedas, 3 si se mezclan 3...). A continuación se extrae el número de filas y columnas que tiene la tabla y se crea un *cell\_array* de una fila y tantas columnas como la tabla. Este *cell\_array* se llama 'nuevo'. Los dos primeros campos de este registro se ponen a *false* porque la rueda nueva va a ser no comercial. Seguidamente se recorren todas las filas de la tabla parándonos solo en las marcadas. En éstas se evalúan todos sus campos a partir del tercero. Si el campo es alfanumérico se van concatenando, si es numérico se va calculado la media aritmética. Una vez evaluadas todas las filas y creado el registro nuevo, se inserta en la base de datos y se actualiza la tabla de pantalla con el contenido de la base de datos. Para ver la función que realiza esta acción véase la sección 5.4 en la página 65.

**Copiar** Hace una copia de cada una de las ruedas seleccionadas. Al igual que mezclar, se carga en la variable a los *true/false* de las filas marcadas. Se obtiene el número de registros de la tabla (variable *r*). Se hace la conexión a la base de datos y se recorre el bucle para buscar los registros marcados. Por cada registro marcado se inserta de nuevo en la base de datos. La propia base de datos se encarga de generar el identificador autonumérico que distingue a estos registros aparentemente iguales. Al finalizar el ciclo se cierra la conexión a la base de datos y se actualiza el contenido de la pantalla igual que antes. Para ver la función que realiza esta acción véase 5.5 en la página 66.

Tanto el botón mezclar como crear crean ruedas “no comerciales”. Se entiende que estas son modificables por el usuario.

Listing 5.3: Código para seleccionar un registro dentro de una tabla

```
1 % Executes when selected cell(s) is changed in uitable1.
2 function uitable1_CellSelectionCallback(hObject, eventdata,
   handles)
3 % hObject    handle to uitable1 (see GCBO)
4 % eventdata  structure with the following fields (see
   UITABLE)
5 % Indices: row and column indices of the cell(s) currently
   selecteds
6 % handles    structure with handles and user data (see
   GUIDATA)
7
8 datos = handles.datos;
9
10 if size(eventdata.Indices,1) == 0;
11     return
12 end
13
14 fila = eventdata.Indices(1);
15 col = eventdata.Indices(2);
16
17 %Pasa dos veces, y en una de ellas Indices está vacío
18 if ~isnan(fila + col)
19     if col <= 2
20         %Si han pinchado en una de las dos primeras columnas
21         %Cambio el valor lógico del primer campo del registro
22
23         flag = datos{fila,1};
24         datos{fila,1} = ~flag;
25         set(hObject,'Data',datos);
26         handles.datos = datos;
27     end
28 end
29 % Update handles structure
30 guidata(hObject, handles);
```



Listing 5.4: Crear un registro a partir de otros existentes

```

1  % --- Executes on button press in Mezclar.
2  function Mezclar_Callback(hObject, eventdata, handles)
3  %Con esta opción se crea una rueda nueva,
4  %promedio de las elegidas
5
6  datos = handles.datos;
7  a = cell2mat(datos(:,1));
8  marcados = sum(a);
9  [r c] = size(datos); %rows and columns
10
11 %creamos un registro nuevo (solo uno)
12 nuevo = cell(1,c);
13 nuevo(1:2) = {false false};
14
15 for fila=1:r
16     if a(fila) %fila marcada
17         for col = 3:c %las dos primeras columnas no son
18             %evaluadas
19             if ischar(datos{fila,col})
20                 nuevo{col} = [nuevo{col} '/' datos{fila,col}];
21             else
22                 nuevo{col} = sum(nuevo{col}) + datos{fila,
23                     col}/marcados;
24             end
25         end
26         %desmarco la fila
27         datos{fila,1} = false;
28     end
29 end
30
31 %Me conecto a la base de datos
32 conn = database('Satelite','Admin','');
33 %inserto el registro
34 insert(conn, handles.tabla, handles.colnames(3:end),nuevo
35     (3:end));
36 %cierro la conexion
37 commit(conn);
38 close(conn);
39
40 %y actualizamos la tabla
41 Actualizar_Callback(hObject, eventdata, handles);
42
43 % Update handles structure
44 guidata(hObject, handles);

```

Listing 5.5: Código para hacer la copia de un registro

```

1  % --- Executes on button press in Copiar.
2  function Copiar_Callback(hObject, eventdata, handles)
3  % hObject    handle to Copiar (see GCBO)
4  % eventdata  reserved - to be defined in a future version
   of MATLAB
5  % handles    structure with handles and user data (see
   GUIDATA)
6  % Con este botón insertamos en base de datos las ruedas
   marcadas
7  % Se insertan como no oficiales
8
9  datos = handles.datos;
10 a = cell2mat(datos(:,1));
11 %En a tenemos las marcadas
12
13 r  = size(datos,1); %rows
14
15 %conectamos a la base de datos
16 conn = database('Satelite','Admin','');
17
18 for fila=1:r
19     if a(fila) %fila marcada
20         insert(conn, handles.tabla,...
21             handles.colnames(3:end),datos(fila,3:end));
22         datos{fila,1}= false;
23     end
24 end
25 commit(conn);
26 close(conn);
27
28 %y actualizamos la tabla
29 Actualizar_Callback(hObject, eventdata, handles);
30
31 % Update handles structure
32 guidata(hObject, handles);

```

**Borrar** Borra las ruedas seleccionadas. Solo permite borrar las ruedas “no comerciales”. La programación es idéntica a la de copiar pero la instrucción ejecutar con cada registro marcado es la de borrado (*delete*). En el vector columna a se tienen los registros marcados y en el vector columna b los registros correspondientes a ruedas comerciales. Se recorre la tabla buscando los registros marcados y no comerciales, y se ejecuta la sentencia de borrado identificando al registro por su identificador único. El método es redundante puesto que imponemos la condición de no comercial en la sentencia de borrado y en la condición if previa a esta sentencia. Para borrar, todas las precauciones son pocas. (Véase listado 5.6 en la página siguiente)

**Modificar** Para modificar una rueda basta con cambiar el valor en la tabla y dar al botón modificar. Si la rueda es comercial no permite modificaciones. Si no es comercial se marca la casilla de indicador para recordarnos que hay que pulsar el botón modificar para que la base de datos se actualice.

**Ordenar** Se ha incluido una *listbox* donde se puede elegir el campo de la tabla por el cual se puede ordenar los datos. Cambiando el valor de este listbox automáticamente la tabla se ordena por el campo elegido.

### 5.2.3. Selección y situación

#### 5.2.3.1. Selección

**Situar** Para añadir o quitar ruedas al diseño del satélite se ha habilitado el botón *Situar*. Las ruedas seleccionadas para el satélite se almacenan en otra tabla llamada *R\_colocada*. Si se pulsa este botón sin tener ninguna rueda seleccionada, se abre el cuadro donde se ven las ruedas seleccionadas en intentos anteriores y permite borrarlas de forma independiente. Si se selecciona una o más ruedas, además de mostrar las ya añadidas al satélite, se abre un cuadro para situar cada una de las ruedas seleccionadas. Este cuadro permite al usuario elegir la posición de la rueda elegida. El programa se queda esperando a que termine el posicionamiento de cada rueda, y si es correcto (se puede abortar la inserción de la rueda) se añade un registro a la tabla de *R\_colocada*.

Listing 5.6: Código para borrar un registro

```

1 % --- Executes on button press in Borrar.
2 function Borrar_Callback(hObject, eventdata, handles)
3 % hObject    handle to Borrar (see GCBO)
4 % eventdata  reserved - to be defined in a future version
   of MATLAB
5 % handles    structure with handles and user data (see
   GUIDATA)
6
7
8 datos = get(handles.uitable1,'Data');
9 a = cell2mat(datos(:,1));
10 b = cell2mat(datos(:,2));
11 %En a tenemos las marcadas
12 % y en b las oficiales
13
14 [r ~] = size(datos); %rows and columns
15
16 %conectamos a la base de datos
17 conn = database('Satelite','Admin','');
18
19
20 for fila=1:r
21     if a(fila) && ~b(fila) %fila marcada y no oficial
22         condicion=[' where Oficial = false and Id = ' ...
23             num2str(handles.ids(fila))];
24         exec(conn,...
25             ['delete from ',handles.tabla , condicion]);
26     end
27     datos{fila,1} = false;
28 end
29 commit(conn);
30 close(conn);
31
32 Actualizar_Callback(hObject, eventdata, handles);
33 % Update handles structure
34 guidata(hObject, handles);

```

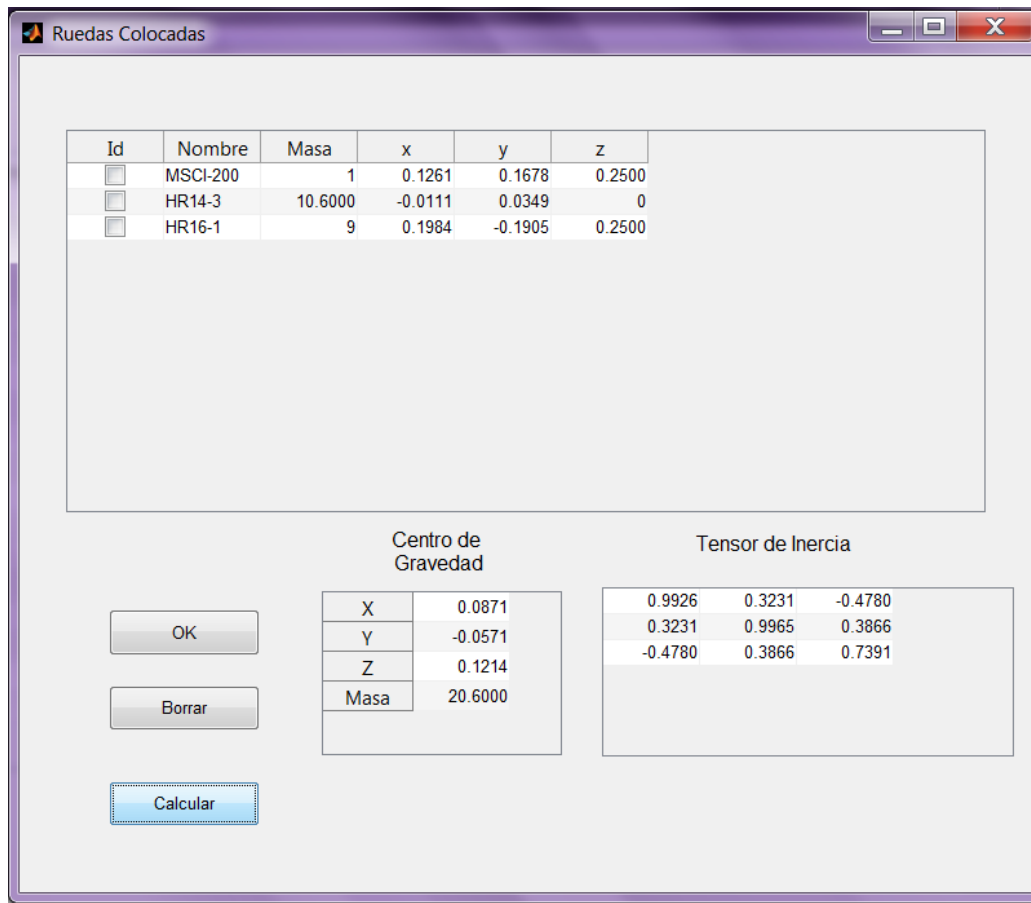


Figura 5.5: Contenido de la tabla de ruedas elegidas

El cuadro así mismo da la opción de cálculo para mostrar el centro de gravedad, masa total y tensor de inercia de las ruedas seleccionadas hasta el momento. (Ver figura 5.5)

### 5.2.3.2. Situación

El cuadro `Coloca_r` recibe como input los datos del satélite y el nombre de la rueda que se va a colocar. (Véase listado 5.7). La función de este cuadro es devolver las coordenadas de la posición de colocación de la rueda. Los datos se reciben en la propiedad `UserData` de la figura. Como son dos datos los recibidos (la estructura de variables del satélite y el nombre de la rueda

a colocar) se envían y reciben en un *cell\_array* . La primera componente del *cell\_array* es la estructura con las variables del satélite y la segunda componente es un string con el nombre de la rueda. Con este nombre se construye el nombre de la barra de título de la ventana.

A continuación se crea el resultado que este cuadro va a dar a su llamador. Se hace con la instrucción:

```
handles.salida=[Nan Nan Nan]
```

Se definen las variables L2, W2 y sn para identificar la mitad del largo, la mitad del ancho y el número de bandejas que tiene el satélite. Con estas variables y la altura del satélite se construyen las superficies Z (siendo ésta una matriz de superficies) para dibujar las bandejas. El cuadro dispone de dos imágenes (axes), véase figura 5.6 en la página 78. En la primera se hace el dibujo tridimensional de las bandejas y en la segunda una vista bidimensional de la bandeja elegida. Se empieza por la primera imagen y se parametrizan los límites coincidiendo con las dimensiones del satélite. Con *axis.equal* se consigue que el factor de escala sea el mismo para los tres ejes. (Para que la imagen no se distorsione). Se programa un ángulo de visión que se podrá cambiar de forma interactiva. La segunda imagen se configura igual pero de forma bidimensional. No se configura punto de vista y los límites son largo y ancho del satélite.

Por último se configura la lista de bandejas en la parte inferior de la pantalla. La vista tridimensional tiene la posibilidad de ser girada con los *slider* posicionados abajo y a la izquierda. Su única función es cambiar el punto de vista de la imagen. El usuario debe primero elegir la bandeja donde va a colocar la rueda. Lo hace haciendo click sobre la lista de bandejas. Al elegir una bandeja, se calcula la altura de la misma, se asigna a la tercera componente (z) de la colocación de la rueda y se muestra por pantalla. Se borran las bandejas y se dibuja solo la elegida. El usuario debe elegir la posición xy para terminar de colocar la bandeja. Puede hacerlo directamente en las casillas o utilizando la segunda imagen. Si el usuario intenta introducir un valor que esté fuera de límites no se acepta, rellenándose la casilla correspondiente con el valor que tuviera anteriormente. (Ver listado 5.8).

Si se quiere colocar mediante imagen se debe pulsar el botón *posición*. Dicho botón funciona de la siguiente manera: lo primero que hace es auto-desactivarse. Al mismo tiempo activa la segunda imagen *axes2* y le cambia el título ('Pulse botón derecho para terminar'). A continuación se introduce en

Listing 5.7: Recogida de datos de un cuadro de diálogo en su llamada

```
1  %Tomamos los datos de la base de datos, porque nos los
   pasan en la llamada
2  %al cuadro de dialogo
3  if nargin > 3
4      recibido = get(hObject,'UserData');
5      s = recibido{1};
6      nombre = recibido{2};
7  else
8      r = load('datos_satelite.mat');
9      s = r.s1;
10     nombre = 'Anonimo';
11 end
12
13 set(hObject,'Name',['Coloca Rueda ' nombre]);
14
15 handles.s = s;
16
17 %Creamos el resultado
18 handles.salida = [NaN NaN NaN];
19
20
21 %Cargamos todos los parámetros del cuadro que dependen de
   las variables
22 %externas que nos han pasado. Hay que hacerlo aqui porque
   si lo hacemos en
23 %otro sitio, todavía no están disponibles
24
25 L2 = handles.s.LengthCube/2;
26 W2 = handles.s.WidthCube/2;
27 sn = handles.s.SheetsNumber;
```

Listing 5.8: Verificación de que el dato introducido está dentro de límites

```
1 function x_Callback(hObject, eventdata, handles)
2 % hObject    handle to x (see GCBO)
3 % eventdata  reserved - to be defined in a future version
  of MATLAB
4 % handles    structure with handles and user data (see
  GUIDATA)
5
6 % Hints: get(hObject,'String') returns contents of x as
  text
7 %          str2double(get(hObject,'String')) returns contents
  of x as a double
8
9 xlim = get(handles.axes2,'XLim');
10 x = str2double(get(hObject,'String'));
11 if isnan(x) || x < xlim(1) || x > xlim(2)
12     set(hObject,'string',num2str(handles.salida(1)));
13 else
14     handles.salida(1) = x;
15 end
16 % Update handles structure
17 guidata(hObject, handles);
```



un bucle en el que guarda la posición cada vez que se hace click con el botón izquierdo del ratón. Cada vez que se pulsa se actualizan los valores *xy* de las casillas y de la variable de salida. Si se pulsa el botón derecho se sale del bucle quedándose guardada la última selección válida. Se vuelve a activar el botón y a colocar el título original por si se quiere repetir la operación. Para ver la parte de código correspondiente a esta funcionalidad véase la página 74.

Una vez que se tienen las tres coordenadas fijadas, se devuelven al programa llamador pulsando ‘ok’. Si se cierra el cuadro de otra manera no se devuelven las coordenadas y la rueda no se inserta en la base de datos de ruedas elegidas. Para conseguir que el cuadro llamador se quede esperando a que éste termine (con Ok o sin él) hay que programar un serie de pasos en el código de este cuadro:

1. Cuando se abre el cuadro (la función es *coloca\_r\_OpeningFcn*) añadir la instrucción:

```
uiwait(handles.figure1)
```

2. En la función *coloca\_r\_OutputFcn* declarar la variable de salida

```
varargout{1}=handles.salida
```

3. En el botón de salida ‘Ok’ hay que introducir la instrucción

```
uiresume(handles.figure1)
```

4. Por último en la función *figure1\_CloseRequestFcn* se vacía el vector de salida (`handles.salida=[]`) y se añade la instrucción

```
uiresume(hObject)
```

Si de este cuadro salimos con el botón ‘Ok’ el cuadro llamador recibe el vector con las tres componentes de la posición. Si no, recibirá un vector vacío. El programa comprueba que se tienen las 3 componentes con valores numéricos sumándolas y verificando que el resultado es un número. En caso afirmativo se crea un *cell\_array* llamado nuevo compuesto por:

Listing 5.9: Cumplimentar las coordenadas xy gráficamente con el botón posición y cuadrícula

```
1  % --- Executes on button press in P_posicion.
2  function P_posicion_Callback(hObject, eventdata, handles)
3  % hObject      handle to P_posicion (see GCBO)
4  % eventdata    reserved - to be defined in a future version
5  % of MATLAB
6  % handles      structure with handles and user data (see
7  % GUIDATA)
8
9  set(hObject, 'Enable', 'off');
10
11 axes(handles.axes2);
12 title('Pulse "Botón derecho" en el ratón para terminar')
13 boton = 1;
14 while boton == 1
15     [x,y,boton]=ginput(1);
16     if boton == 1
17         set(handles.x, 'String', num2str(x));
18         set(handles.y, 'String', num2str(y));
19         handles.salida(1:2) = [x y];
20     end
21 end
22
23 title('Pulse "Posición" para definir la colocación')
24
25 set(hObject, 'Enable', 'on');
26 % Update handles structure
27 guidata(hObject, handles);
```

Listing 5.10: Llamada a la rutina de colocación de la rueda e inserción en la base de datos

```

1  %conectamos a la base de datos
2  conn = database('Satelite','Admin','');
3
4  for fila=1:r
5      if a(fila) %fila marcada
6          %Abrimos el cuadro para posicionar
7          %Pasamos datos del satelite y nombre de la rueda.
8          posicion = ...
9          coloca_r('UserData',{handles.s q(fila).Nombre});
10         %y se queda esperando
11
12         %si me devuelve una posición de verdad
13         %En posicion(3) esta la coordenada z elegida.
14         %Se puede sumar la mitad del espesor de la rueda
15         if ~isnan(sum(posicion))
16             nuevo = {q(fila).Nombre q(fila).Masa};
17             nuevo(3:5) = num2cell(posicion);
18             insert(conn,...
19                 'R_colocada', colnames2(2:end),nuevo);
20         end
21         %Desmarcamos la fila
22         datos{fila,1} = false;
23
24     end
25 end
26 commit(conn)
27 close(conn)

```

$$nuevo = \begin{Bmatrix} \text{Nombre Rueda} \\ \text{Masa Rueda} \\ \text{Coordenada } x \\ \text{Coordenada } y \\ \text{Coordenada } z \end{Bmatrix}.$$

Ahora se inserta este registro en la base de datos en la tabla *R\_colocada*.

Para borrar una rueda posicionada basta seleccionarla en el cuadro de ruedas colocadas y dar al botón borrar. La programación es idéntica a borrar una rueda no comercial en el catálogo, saltándose el paso de comprobación

de que es comercial. (Ver página 67)

#### 5.2.4. Cálculos

##### Cálculo del centro de gravedad y tensor de inercia de los elementos colocados

Para realizar estos cálculos se ha desarrollado la función *masas*, cuyo argumento es el nombre de la tabla donde están guardados los elementos con sus masas y coordenadas. La función devuelve un escalar que es la masa, un vector que es el centro de gravedad y una matriz que es el tensor de inercia. Lo primero que se hace es inicializarlo a sus valores nulos. A continuación trae con la función *Tabla\_db (tabla)* los registros de la tabla correspondiente: ruedas, sensores o propulsores. (Estas funciones se ha creado no sólo para este apartado sino para todos los que lo requieran). Se crea una estructura llamada *q* con los datos jerarquizada por los nombres de los campos. El cálculo se ejecuta en un bucle donde se recorren todos los registros de la tabla. La masa se calcula como sumatorio de masas, el centro de gravedad como sumatorio de momentos estáticos dividido por la masa total y el tensor de inercia acumulando los tensores de inercia de cada una de las ruedas de los elementos. (Ver listado 5.11)

$$masa = \sum masa_i,$$

$$\mathbf{cg} = \frac{\sum masa_i * \mathbf{r}_i}{masa},$$

$$\mathbf{Inercia} = \sum masa_i * (\mathbf{r}_i^2 * \mathbf{I}_3 - \mathbf{r} * \mathbf{r}^T).$$

Donde  $\mathbf{r}_i$  es el vector de posición de cada rueda e  $\mathbf{I}_3$  la identidad de 3x3.

Listing 5.11: Función para calcular masas, centro de gravedad y tensor de inercia de los elementos colocados

```
1 function [ masa cg inercia ] = masas( tabla )
2 %MASAS
3 %Esta funcion hace la suma de las masas
4 %y momentos de inercia de los dispositivos elegidos
5 %consultando la tabla donde se han guardado.
6
7 masa = 0;
8 cg = [0 0 0 ];
9 inercia = zeros(3);
10
11 [datos,colnames,ids] = Tabla_db(tabla);
12
13 q = cell2struct(datos,colnames,2);
14
15 a = size(datos,1);
16
17 for i = 1:a
18     masa = masa +q(i).Masa;
19
20     %vector fila con las coordenadas
21     vf = [q(i).x q(i).y q(i).z];
22
23     %sumo momentos estáticos
24     cg = cg + q(i).Masa * vf;
25
26     %calculo tensor de inercia
27     inercia = inercia + ...
28     (vf*vf'*eye(3) - vf'*vf)*q(i).Masa;
29
30 end
31
32 %convierto los momentos estáticos en coordenadas
33 %del centro de masas
34 cg = cg / masa;
35
36 end
```

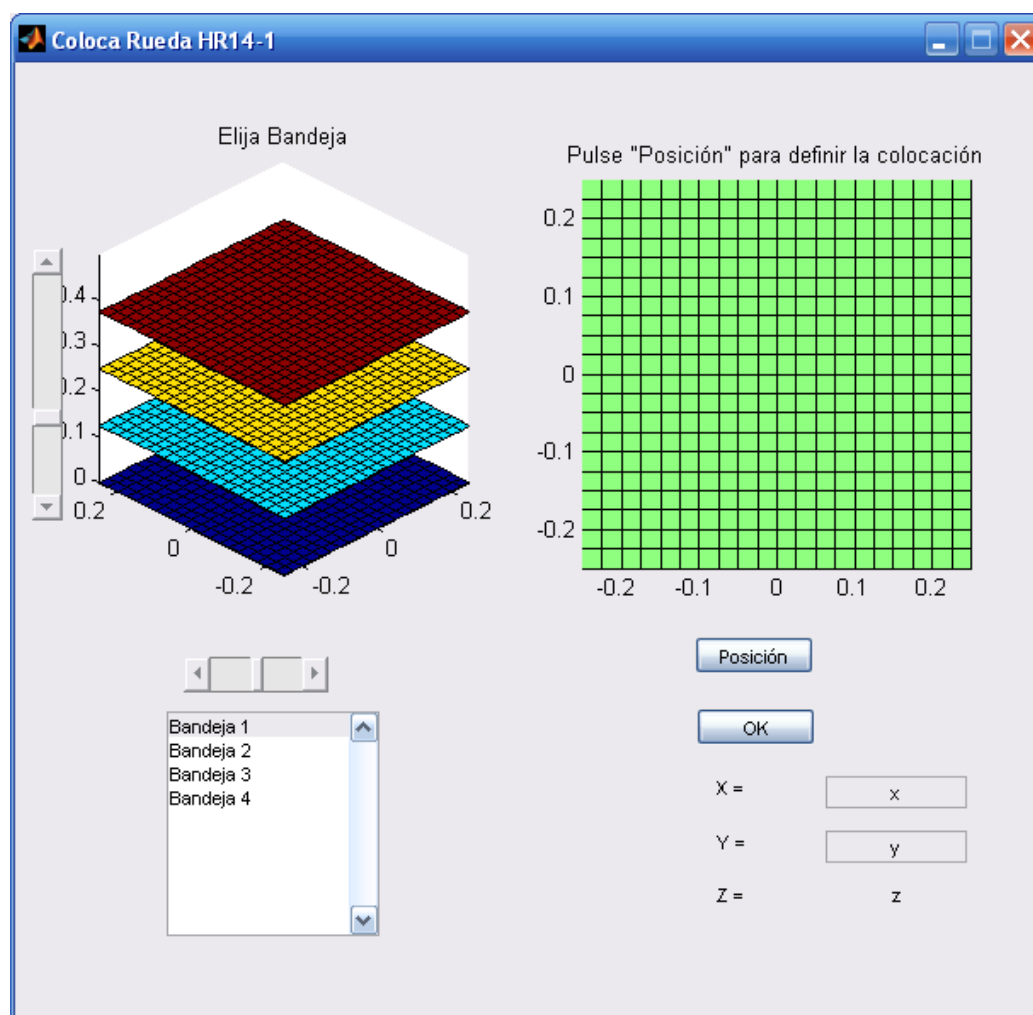


Figura 5.6: Cuadro de diálogo para situar una rueda

# Capítulo 6

## Actuadores con combustible

### 6.1. Conceptos teóricos

Los actuadores con combustible, también llamados microcohetes, son motores cohete de combustible líquido normalmente. Se pueden arrancar por instantes de tiempo muy pequeños, para dar el impulso deseado para un ajuste de actitud. La configuración normal es por pares, de modo que encendiendo motores de sentidos opuestos generen un par que sirve para modificar la actitud. Si se encienden dos del mismo sentido genera un impulso que modifica la trayectoria. En su colocación (en el exterior del satélite) hay que tener precaución para que los gases eyectados no dañen al propio satélite. [11]

Se pueden distinguir varios tipos de maniobras:

1. Pequeñas maniobras para ajuste de actitud. Normalmente usadas para compensar perturbaciones cuando no hay otros actuadores. Esta función la suelen desempeñar las ruedas o magnetopares.
2. Maniobra de descarga de ruedas. Como se menciona en la subsección 5.1.1 las ruedas se saturan y hay que descargarlas. Para conseguirlo, hay que accionar la rueda en sentido contrario al que está saturado y al mismo tiempo activar los microcohetes que generen el par opuesto. De este modo, el par aplicado total será aproximadamente nulo. El satélite no cambiará notablemente su actitud. Si la rueda es de inercia, se ha saturado por perturbaciones seculares. Lo lógico entonces es aplicar la desaturación hasta llevarla a un punto próximo a la saturación en sentido opuesto.

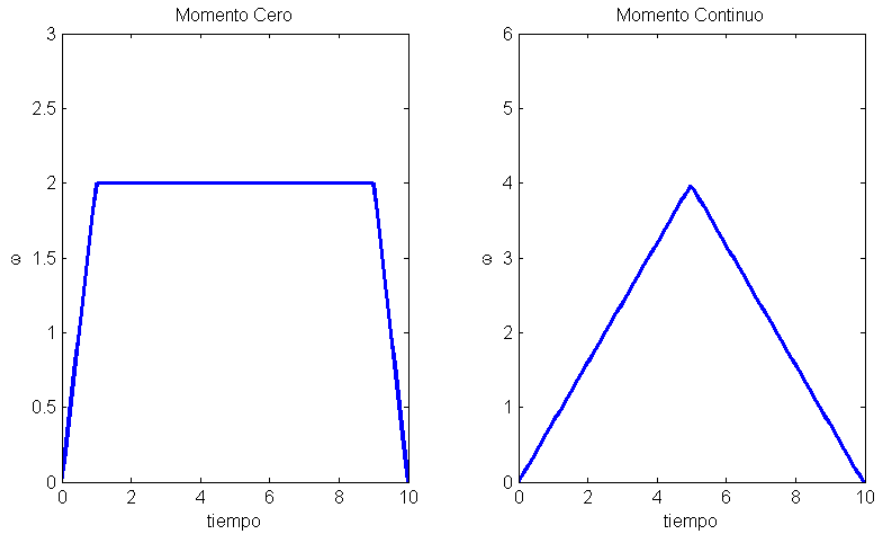


Figura 6.1: Maniobras de momento cero y momento continuo

3. Grandes maniobras. Cuando se quiere hacer un cambio de actitud grande, por ejemplo cuando cambia la consigna del objetivo a apuntar con el satélite. Estas maniobras se dividen en dos tipos (ver figura 6.1):

- a) Maniobras de momento cero. En esta maniobra sólo una porción del tiempo el actuador está funcionando. Se denomina ciclo de trabajo la proporción de tiempo que el actuador actúa acelerando o frenando respecto al tiempo total de la maniobra.
- b) Maniobras de momento continuo. En estas maniobras los actuadores están funcionando durante toda la maniobra. La mitad del tiempo acelerando y la otra mitad frenando.

■ Input

- Datos de misión

$\theta_{ac}$  Ángulo total de maniobra (deg)

$t_{ac}$  Tiempo total de la maniobra (min)

$\mathbf{T_D}$  Par de perturbaciones (Ecuación (4.1))

$r$  Ciclo de trabajo (%)



$\mathbf{t_m}$  Vida de misión (años)  
 $\mathbf{n_g}$  Número de maniobras grandes por año  
 $\mathbf{n_p}$  Número de maniobras de corrección por año  
 $\mathbf{H_w}$  Momento acumulado por la rueda de inercia (Ecuación (5.1))

- Datos de satélite

$\mathbf{I}$  Momentos de inercia ( $kg * m^2$ )  
 $\mathbf{L}$  Distancia de colocación (m)

- Datos de propulsores

$\mathbf{t_g}$  Tiempo de pulso para gran maniobra (seg)  
 $\mathbf{t_p}$  Tiempo de pulso para pequeña maniobra (seg)  
 $\mathbf{I_{sp}}$  Impulso específico del combustible (seg)

- Output

$\mathbf{F_D}$  Fuerza impulsora para contrarrestar perturbaciones (Ecuación (6.1))

$\mathbf{F_S}$  Fuerza impulsora para maniobras grandes (Ecuaciones (6.3) y (6.4), dependiendo si se diseña la maniobra con momento cero o momento continuo)

$\mathbf{N_p}$  Número de pulsos totales para el microcohetes

$\mathbf{I_T}$  Impulso total ( $N*s$ ) (Ecuación (6.6))

$\mathbf{M_p}$  Masa del propulsante (kg) (Ecuación (6.7))

Ecuaciones necesarias para los cálculos:

$$F_D = \frac{T_D}{L}, \quad (6.1)$$

$$\dot{\theta} = \frac{\theta_{ac}}{t_{ac}} * \frac{\pi}{180 * 60},$$

$$\ddot{\theta} = \frac{\dot{\theta}}{rt_{ac}}, \quad (6.2)$$

$$F_S = \frac{I\ddot{\theta}}{L}, \quad (6.3)$$

$$F_S = \frac{H_w \dot{\theta}}{Lr}, \quad (6.4)$$

$$N_p = t_m (n_g + n_p), \quad (6.5)$$

$$I_T = t_m (n_g t_g F_S + n_p t_p F_D), \quad (6.6)$$

$$M_p = \frac{I_T}{I_{sp} * 9,8}, \quad (6.7)$$

## 6.2. Programación

### 6.2.1. Interfaz de datos

Tras calcular y seleccionar las ruedas se procede a la siguiente fase del programa: los propulsores. Este cuadro de diálogo hereda los datos de los anteriores (secciones 4.2 y 5.2).

La recepción de los datos del satélite y de los datos introducidos manualmente y calculados por los programas anteriores se realiza de forma análoga a lo que se explicó en la subsección 5.2.1. La forma de actualizar los datos de pantalla también es igual a los interfaces anteriores. En este interfaz se ha incluido el acceso al catálogo de sensores para su selección. Este botón se encuentra accesible desde el primer momento sin importar los cálculos previos o posteriores. El funcionamiento es idéntico a la selección de propulsores que se explica en el apartado 7.2.1.

La primera opción de cálculo que permite calcular las fuerza precisa que se hayan cumplimentado los datos del panel datos propulsores. Se ejecuta pulsando el botón *calcular fuerzas*. Si falta algún dato, se dispara un mensaje de error. Los cálculos se realizan según se ha explicado en el apartado 6.1.

En este momento se habilita el botón de calcular otros resultados. Para poder efectuar los cálculos hace falta cumplimentar los datos del panel *datos de misión*. Una vez finalizado este proceso se habilita el botón de *selección de propulsores* y *Grabar en BD*.

The screenshot shows a software window titled 'Propulsores' with three main panels:

- Datos propulsores (Left Panel, light blue background):**
  - Brazo del propulsor (m): 0.25 m
  - % tiempo aceleracion: 10 %
  - Tiempo de quemado (s): 80 s
  - Impulso específico (s): 800 s
  - Tiempo maniobra momento zero (s): 30 s
  - Segundos en un pulso de gran maniobra: 5 s
  - Segundos en un pulso de amortiguación de momento: 0.5 s
- Fuerzas propulsores (Middle Panel, light orange background):**
  - Fuerza para contrarrestar perturbaciones: 1.6968e-005 N
  - Fuerza para maniobras: 0.24367 N
  - Fuerza para amortiguar momento: 1.0471 N
  - Fuerza de momento cero: 1.6522
- Otros resultados (Right Panel, light orange background):**
  - Masa del propulsante: 0.020676 kg
  - Tiempo de vida del propulsor (pulsos): 230 pulsos

Below the 'Fuerzas propulsores' panel is the **Datos mision (light blue background)** section:

- Años de vida de mision: 5
- Encendidos / año para maniobras grandes: 12
- Encendidos / año para amortiguar el momento: 34

At the bottom right, there is a 'Propulsores' section with three buttons: 'Calcular fuerzas', 'Calcular otros resultados', and 'Selección propulsores'. A large red button labeled 'Grabar en BD' is located at the bottom center.

Figura 6.2: Interfaz de cálculo de propulsores

## 6.2.2. Selección y situación de propulsores

### 6.2.2.1. Selección

Una vez estén cumplimentados y calculados todos los datos de la pantalla (Figura 6.2) se pulsa el botón de selección de propulsores. Se abre entonces el catálogo de propulsores que se encuentra en la base de datos local, al igual que el catálogo de actuadores. Sobre este catálogo se puede operar de manera análoga al catálogo de actuadores (opciones de mezclar, copiar, borrar y modificar). Véase 5.2.2.

### 6.2.2.2. Situación

El funcionamiento del botón *situar* es parecido al de situar una rueda, pero con pequeñas diferencias.

Para situar uno o más propulsores, primero hay que marcarlos en la columna *Id* y pulsar el botón situar. Aparece el interfaz *coloca propulsor* de

la figura 6.3. En este interfaz se muestra el paralelepípedo que representa el satélite y hay que elegir en el *listbox* la cara sobre el cuál va a ir situado. Se puede girar la vista del satélite con las barras de deslizamiento igual que se hacía con las ruedas. Una vez elegida la cara se quita el color de las demás y en la imagen de la derecha aparece la cara seleccionada con círculos si ha elegido una cara positiva (vector saliente) y con x si se elige una cara negativa (vector entrante). Ver figura 6.4.

Pulsando sobre el botón posición *cara elegida* (En la figura 6.4 se ve posición Y+) permite con el cursor situar el punto de colocación. Cada vez que se pulse el botón izquierdo del ratón se apunta la posición elegida. Pulsando el botón de la derecha del ratón se acepta la última posición como buena y aparece en la figura un triángulo representando el propulsor colocado y orientado. Esto se muestra en la figura 6.5. La orientación del sensor se cambia a posteriori actuando sobre la barra de deslizamiento que está encima de la representación de la cara. Todos los valores numéricos de posición y orientación se pueden introducir manualmente en las casillas inferiores.

Si de este cuadro salimos con el botón ‘Ok’ el cuadro llamador recibe el vector con las tres componentes de la posición, la cara elegida y el ángulo de orientación. Si no, recibirá un *array* vacío. El programa comprueba que se tienen las 3 componentes con valores numéricos sumándolas y verificando que el resultado es un número. En caso afirmativo se crea un *cell\_array* llamado nuevo compuesto por:

$$nuevo = \left\{ \begin{array}{l} \textit{Nombre Propulsor} \\ \textit{Masa Propulsor} \\ \textit{Empuje} \\ \textit{Cara} \\ \textit{Ángulo} \\ \textit{Coordenada x} \\ \textit{Coordenada y} \\ \textit{Coordenada z} \end{array} \right\}.$$

Ahora se inserta este registro en la base de datos en la tabla *P\_colocado*. Para guardar la cara elegida para situar el propulsor se ha seguido la codificación mostrada en el cuadro 6.1 y se ha programado según los listados 6.1 y 6.2.

Nótese la similitud con la colocación de una rueda (subsección 5.2.3.2)

Listing 6.1: Codificación de las caras del satélite en la base de datos

```

1  %Parametrizamos la lista de Caras
2  lista = cell(6);
3
4  lista{1} = 'Cara +X';
5  lista{2} = 'Cara -X';
6  lista{3} = 'Cara +Y';
7  lista{4} = 'Cara -Y';
8  lista{5} = 'Cara +Z';
9  lista{6} = 'Cara -Z';
10
11  %-----
12
13  switch handles.Cara
14
15  %Caras +Z y -Z
16  case 5
17      patch(L2*[-1 -1 1 1],W2*[-1 1 1 -1],H2*[1 1 1 1],'r');
18      z = H2;
19      ncara = 'Z+';
20  case 6
21      patch(L2*[-1 -1 1 1],W2*[-1 1 1 -1],-H2*[1 1 1 1],'r');
22      z = -H2;
23      ncara = 'Z-';
24  %Caras +Y y -Y
25  case 3
26      patch(L2*[-1 1 1 -1],-W2*[1 1 1 1],H2*[-1 -1 1 1],'c');
27      y = W2;
28      ncara = 'Y+';
29  case 4
30      patch(L2*[-1 1 1 -1],W2*[1 1 1 1],H2*[-1 -1 1 1],'c');
31      y = -W2;
32      ncara = 'Y-';
33  %Caras +x y -X
34  case 1
35      patch(-L2*[1 1 1 1],W2*[-1 1 1 -1],H2*[-1 -1 1 1],'y');
36      x = L2;
37      ncara = 'X+';
38  case 2
39      patch(L2*[1 1 1 1],W2*[-1 1 1 -1],H2*[-1 -1 1 1],'y');
40      x = -L2;
41      ncara = 'X-';
42  end
43
44  %

```

Listing 6.2: Representación de la cara del satélite elegida

```

1 function dibuja(handles)
2 %DIBUJA
3 L2 = handles.s.LenghtCube/2;
4 W2 = handles.s.WidthCube/2;
5 H2 = handles.s.HeightCube/2;
6
7 axes(handles.axes2);cla;hold on;axis equal;
8 switch handles.Cara
9     case {1 2}
10         axis ([-W2 W2 -H2 H2 ])
11         patch(W2*[-1 1 1 -1],H2*[-1 -1 1 1],[1 1 1 1],'y');
12     case {3 4}
13         axis ([-L2 L2 -H2 H2 ])
14         patch(L2*[-1 1 1 -1],H2*[-1 -1 1 1],[1 1 1 1],'c');
15     case {5 6}
16         axis ([-L2 L2 -W2 W2 ])
17         patch(L2*[-1 -1 1 1],W2*[-1 1 1 -1],[1 1 1 1],'r');
18 end
19
20 XLim = get(handles.axes2,'XLim');
21 YLim = get(handles.axes2,'YLim');
22
23 %Puntos para mostrar la cuadrícula
24 N = 9;
25 [X,Y] = meshgrid(linspace(XLim(1),XLim(2),N),...
26     linspace(YLim(1),YLim(2),N));
27
28 marcas = ones(size(X))+ 1;
29 switch handles.Cara
30     case {1 3 5}
31         plot3(X,Y,marcas,'ok');
32     case {2 4 6}
33         plot3(X,Y,marcas,'xk');
34 end
35
36 lon = .1; %longitud de la flechita
37 lon2 = [lon lon/3 lon/3];
38 ang = [0 ,.8*pi, -.8*pi] + (handles.Ang+90)*pi/180;
39 switch handles.Cara
40     case {1 2}
41         xp = handles.posicion(2) + lon2.* cos(ang);
42         yp = handles.posicion(3) + lon2.* sin(ang);
43     case {3 4}
44         xp = handles.posicion(1) + lon2.* cos(ang);
45         yp = handles.posicion(3) + lon2.* sin(ang);
46     case {5 6}
47         xp = handles.posicion(1) + lon2.* cos(ang);
48         yp = handles.posicion(2) + lon2.* sin(ang);
49 end
50 patch(xp,yp,[2 2 2],'k');
51 end

```

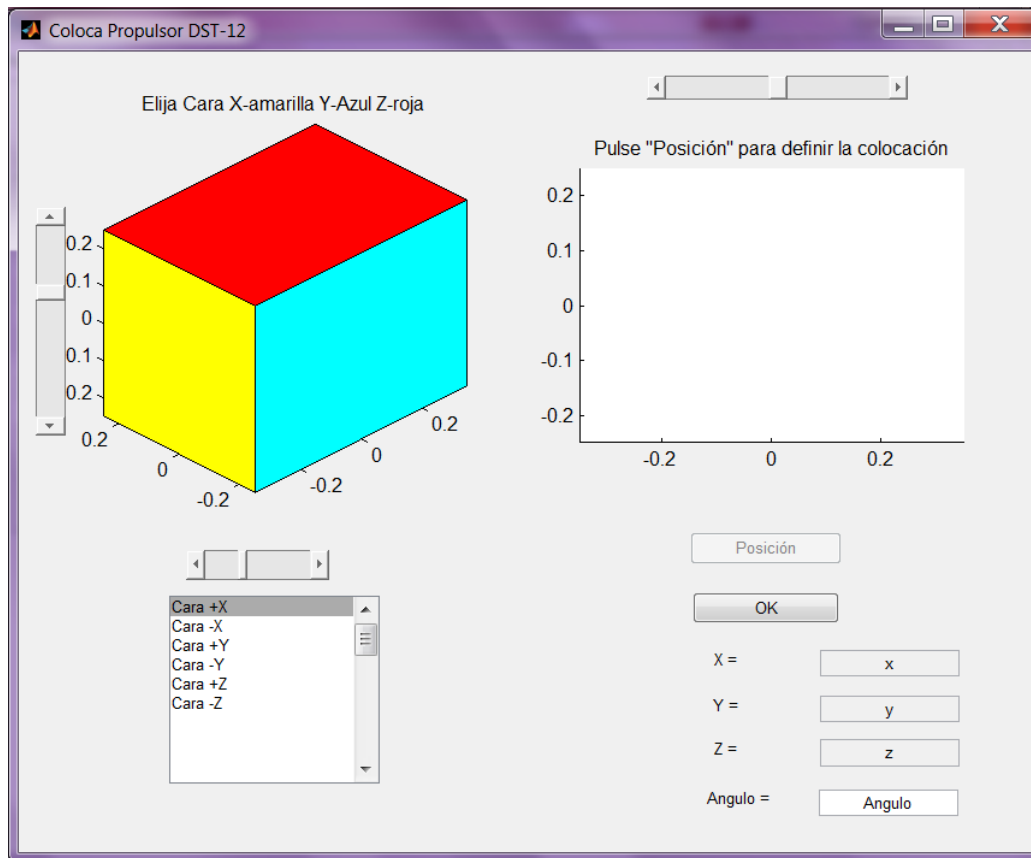


Figura 6.3: Interfaz de colocación de propulsores. Elección de la cara sobre la que va a situarse el sensor.

### 6.2.3. Cálculos

#### Cálculo del centro de gravedad y tensor de inercia de los elementos colocados

Para realizar estos cálculos se ha utilizado la misma función *masas*, cuyo argumento es el nombre de la tabla donde están guardados los elementos con sus masas y coordenadas. El funcionamiento de esta función se explica en el apartado 5.2.4.

Cara	Campo en la base de datos
X+	1
X-	2
Y+	3
Y-	4
Z+	5
Z-	6

Cuadro 6.1: Codificación de las caras del satélite

#### 6.2.4. Grabar en BD

Cuando se pulsa este botón se calcula la suma de masas de todos los elementos seleccionados para este satélite (ruedas, sensores y propulsores) con la función explicada en la sección 5.2.4. Se guarda en la variable *handles.p.masa*. Por último llama a la función *Grabar\_BD* que se explica en la subsección 2.5.2. Consultar el CD adjunto para ver la programación completa.



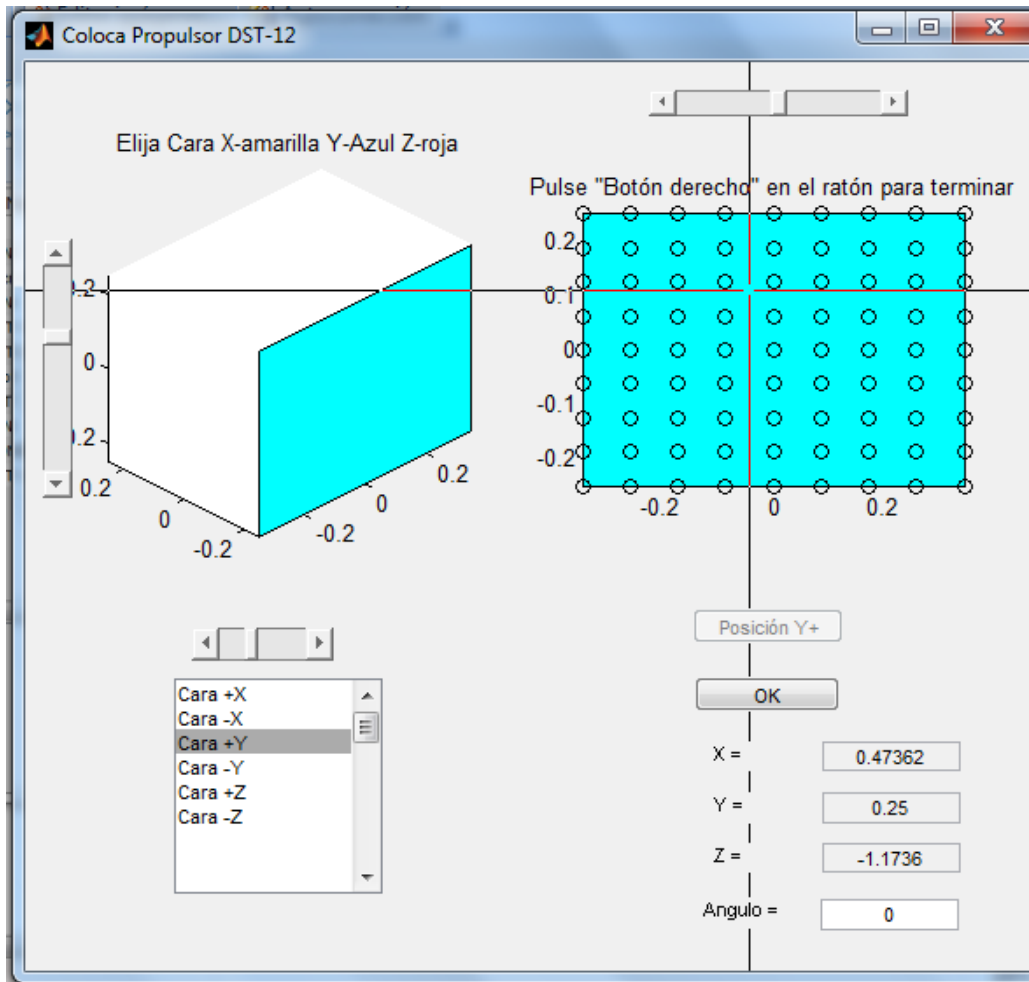


Figura 6.4: Interfaz de propulsores. Situación sobre la cara.

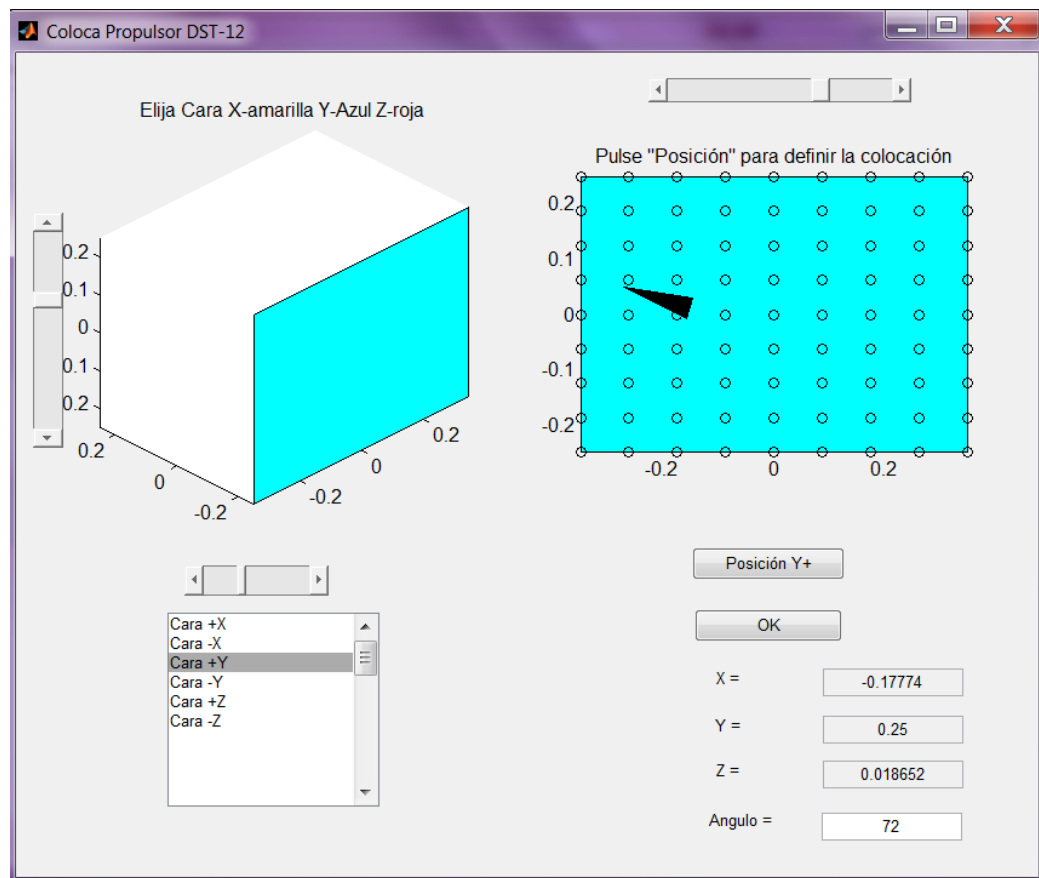


Figura 6.5: Interfaz de colocación de propulsores. Orientación del propulsor.

# Capítulo 7

## Sensores

### 7.1. Conceptos teóricos

Los sensores se encargan de dar una señal legible por el control que sirve para determinar la orientación del satélite. Dependiendo del tipo de sensor se podrá proporcionar uno o dos ángulos. Para la determinación total del satélite harán falta varios sensores. Según el tipo de referencia utilizada por el sensor se distinguen diferentes tipos, siguiendo la clasificación de [10]

#### 7.1.1. Sensor de Sol

Pueden dar el ángulo de orientación respecto al Sol. Su precisión es considerable, unos pocos segundos de arco, pero no son operativos en la zona nocturna de la órbita. Dan sólo dos componentes del vector de actitud. No exigen alimentación eléctrica ya que disponen de células fotovoltaicas. El modelo más simple consiste en dos células iguales bajo un panel (conocido como ‘paraguas’) que hace sombra sobre ellas de forma simétrica cuando el Sol incide verticalmente. Midiendo la diferencia de iluminación entre las dos células se calcula el ángulo de incidencia del Sol.

Otro modelo más preciso consiste en 2 sensores, cada uno está compuesto esencialmente por una cámara rectangular con una fina ranura en la tapa superior. La luz que entra por la ranura proyecta una imagen de una fina línea en el fondo de la cámara. El fondo de la cámara está alineado con una red de células fotosensibles que miden efectivamente la distancia de la imagen desde una línea central.

El funcionamiento interno del sensor es el siguiente:

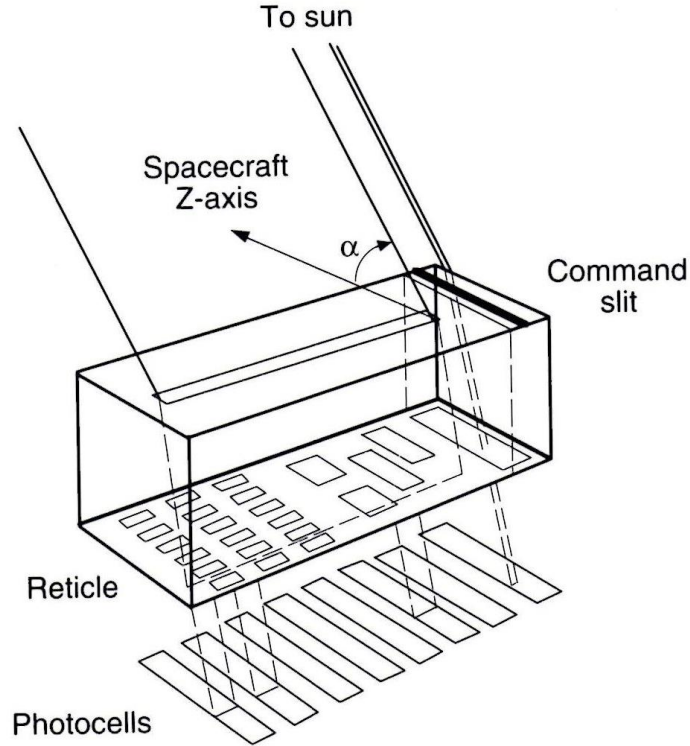


Figura 7.1: Esquema de sensor de Sol de rejilla. Extraído de [10]

■ Parámetros del sensor

**d** Distancia desde la imagen proyectada a una línea central

**h** Altura de la cámara h

$$\tan(\alpha) = \frac{d}{h}$$

Para determinar la dirección del Sol respecto a los ejes del sensor: (Ver figura 7.3)

$$\tan(\alpha) = \frac{\hat{x} * \hat{S}}{\hat{z} * \hat{S}},$$

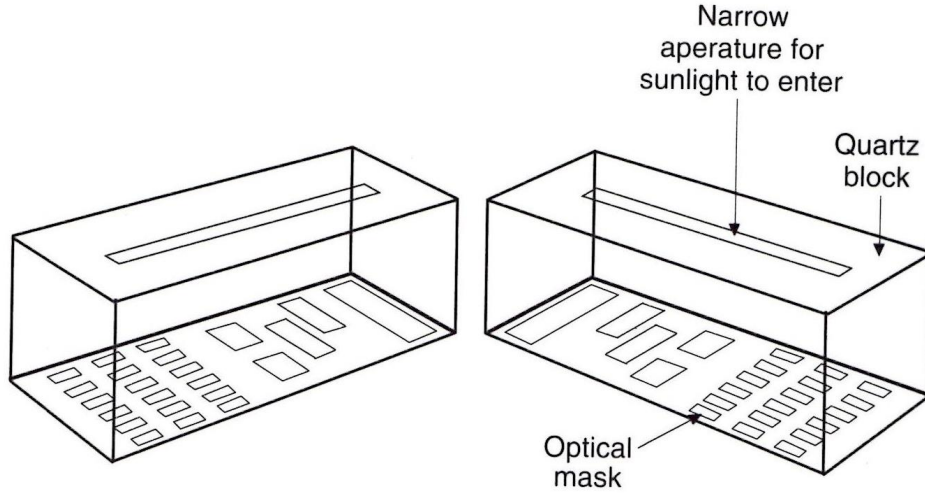


Figura 7.2: Esquema de sensor de sol de dos rejillas. Extraído de [10]

$$\tan(\beta) = \frac{\hat{y} * \hat{S}}{\hat{z} * \hat{S}}.$$

Así determinamos la dirección del Sol en ejes del sensor:

$$\hat{S}_{sun} = \frac{1}{\sqrt{1 + \tan^2(\alpha) + \tan^2(\beta)}} * \begin{Bmatrix} \tan(\alpha) \\ \tan(\beta) \\ 1 \end{Bmatrix}.$$

Para obtenerlo en ejes cuerpo sólo debemos aplicar la matriz de rotación C (matriz de posición del sensor respecto al satélite) y se inserta ruido.

$$\hat{S}_B = C * \hat{S}_I + \Delta \hat{S}_B.$$

Hay sensores de Sol que son puramente digitales (llamados DSAD) que determinan los ángulos dirección del Sol determinando cuál de las células está más iluminada. Su exactitud está limitada por el diámetro angular del Sol (unos 0.5 grados). Los sensores de vector de Sol normalmente tienen campos de visión de  $\pm 60^\circ$ . De tal modo que si el satélite no está estabilizado inercialmente a una actitud a lo largo de la misión, será necesario usar más de una cabeza de sensor para asegurar que el Sol esté siempre visible.

Al igual que el magnetómetro, un sensor solar solo no es suficiente para determinar la actitud. Existe otra variante que son los sensores de sol

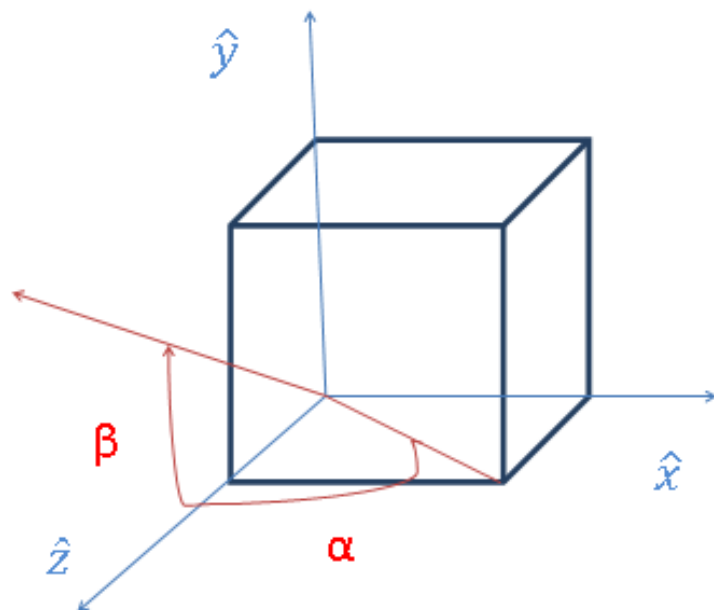


Figura 7.3: Definición de los ángulos del sensor de Sol

giratorios. No gira el sensor, sino el satélite. Su resolución es de 1 grado aproximadamente. Este tipo de sensor, al ser giratorio, sí que determina la actitud completa.

### 7.1.2. Sensor de estrellas

Son de mucha precisión (más allá de un segundo de arco) y dan las tres componentes del vector de actitud. No dan la señal de forma continua, ya que necesita ver dos pasadas sucesivas de un grupo de estrellas para dar la orientación. Tiene el inconveniente de que puede ser cegado por la intensa luz del Sol o de la Luna. Como la luz de las estrellas es muy débil (inferior a  $10^{-12} \text{ W/cm}^2$ ) estos instrumentos son grandes, pesados y caros [13]. A la hora de procesar datos, el sensor de estrellas de dos ejes es muy similar al sensor de Sol. Aunque las tecnologías sean diferentes, virtualmente todos los sensores de estrellas bieje funcionan de la misma manera. La luz de las estrellas alcanza una superficie sensible a la luz, el punto de impacto es determinado y los ángulos de plano  $\alpha$  y  $\beta$  se determinan a partir de este punto. Si siguen una sola estrella se llaman rastreadores estelares, si siguen

a varias se suelen llamar cámaras estelares.

Los sensores de estrellas de dos ejes se suelen colocar en satélites estabilizados en tres ejes, porque los sensores no funcionan adecuadamente a velocidades angulares por encima de unos pocos grados por segundo.

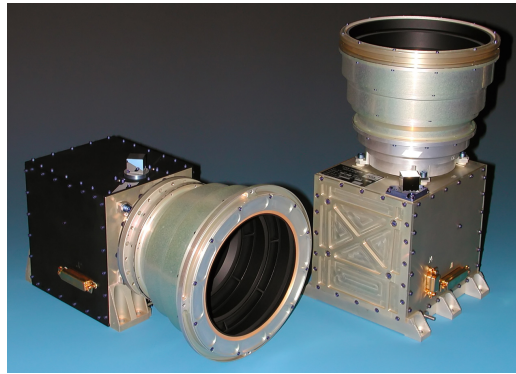


Figura 7.4: Imagen del sensor de estrellas SED26 de la empresa Sodern

### 7.1.3. Sensor horizonte

Determina por infrarrojos el límite de la superficie de la tierra. Son útiles para conseguir una orientación nadir. Suelen ser de dos tipos, de escaneo y estáticos.

#### 7.1.3.1. Estáticos

Tienen un campo de visión ligeramente más amplio que el ángulo que cubre la Tierra e incluyen una serie de sensores que perciben la radiación infrarroja de la superficie terrestre. Con estos sensores, la orientación del nadir se puede determinar con una exactitud de 0.1 grados en una órbita cercana a la Tierra a 0.01 grados en órbitas geosíncronas. Normalmente restringidos a órbitas circulares porque así el ángulo que abarca la Tierra es constante.

#### 7.1.3.2. Escaneo

Emplean un espejo giratorio o prisma para enfocar un haz de luz estrecho. Se suele llamar bolómetro. Su campo de visión suele ser un cono de semiángulo aproximadamente de unos 45 grados. La electrónica del sensor detecta

los intervalos de señal infrarroja de la Tierra recibida/perdida durante cada barrido del cono de escaneo. Entre esos dos tiempos se determina la amplitud que abarca de la Tierra. Así se deduce el ángulo de balance. Este ángulo de balance no es exactamente el que se define como ángulo de balance de Euler.

### 7.1.3.3. Cálculos

#### ■ Input

$\omega_{\text{scanner}}$  Velocidad de rotación del escáner.

$t_{\text{AOS}}$  Tiempo de llegada de la primera señal

$t_{\text{LOS}}$  Tiempo de pérdida de la última señal

$\rho$  Radio angular terrestre

$\gamma$  Semiángulo de cono

#### ■ Output

$\Omega$  Amplitud terrestre (Ecuación (7.1))

$\eta$  Ángulo nadir (ángulo de balance del escáner) (Ecuación (7.2))

$$\Omega = \omega_{\text{scanner}}(t_{\text{LOS}} - t_{\text{AOS}}), \quad (7.1)$$

$$\cos(\rho) = \cos(\gamma) \cos(\eta) + \sin(\gamma) \sin(\eta) \cos\left(\frac{\Omega}{2}\right). \quad (7.2)$$

Nota: para valores de  $\rho \geq \gamma$  un solo valor de  $\eta$ ,  $0 \leq \eta \leq \pi$  satisface la ecuación (7.2). Dicha ecuación dos soluciones, y solamente una es correcta. Se debe utilizar información adicional para eliminar la errónea. Si se utilizan dos escáneres de horizonte, se detecta que la solución falsa es muy diferente en ambos.. El radio angular se reduce con la altura siendo  $\pi/2$  a altitud cero e infinito para valores muy grandes de altitud. (Se considera  $\rho = \gamma$  para altura 1700km) Así que para órbitas cercanas a la Tierra hay una única solución para  $\eta$ .

Si hay un elemento Pickoff (u origen de ángulo de barrido) en el sensor cuyo cruce puede ser detectado con cada pasada del sensor, entonces el ángulo de paso del escáner (el ángulo alrededor del eje de exploración de cono, que con frecuencia, pero no siempre se corresponde con el ángulo de cabeceo del satélite) también se puede determinar.



$$p = \omega_{scanner} \left( \frac{1}{2} (t_{LOS} + t_{AOS}) - t_{pickoff} \right).$$

Si el eje de giro del satélite es el eje  $\mathbf{k}$  y el ángulo de cabeceo se mide desde el eje  $\mathbf{i}$ , entonces el vector nadir  $\hat{\mathbf{E}}_{HS}$  en coordenadas del sensor es:

$$\hat{\mathbf{E}}_{HS} = \begin{Bmatrix} \sin(\eta) \cos(p) \\ \sin(\eta) \sin(p) \\ \cos(\eta) \end{Bmatrix}. \quad (7.3)$$

Con la ecuación (7.4) se traduce el vector nadir a ejes cuerpo y con la ecuación (7.5) se despeja el vector en ejes inerciales.

$$\hat{\mathbf{E}}_B = S_{HS} \hat{\mathbf{E}}_{HS}. \quad (7.4)$$

$$\hat{\mathbf{E}}_B = C \hat{\mathbf{E}}_I + \Delta \hat{\mathbf{E}}_{HS}. \quad (7.5)$$

La exactitud de este tipo de sensores varía desde los 0.1 a 1 grados. Depende hasta cierto punto de la cantidad de datos procesados. El ruido que se introduce debe ser por:

- Ruido electrónico aleatorio
- Tierra achatada
- El sensor no detecta la señal de tierra u océano, sino el punto de la atmósfera que alcanza la intensidad suficiente. Depende de la estación y de la latitud

En órbitas cercanas a la Tierra despreciar estas correcciones disminuye la exactitud de 0.3 a 1 grado.

#### 7.1.4. Giróscopos

Dan la actitud relativa a una orientación fija. Los parámetros a tener en cuenta son:

- Grado de precisión
- Masa

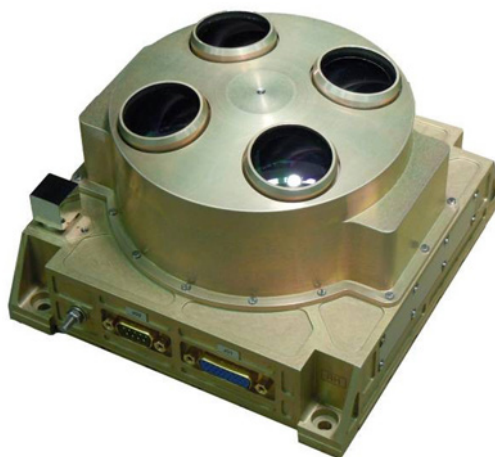


Figura 7.5: Imagen sensor horizonte IRES-C de la ESA

- Lugar de colocación
- Consumo eléctrico

Estos sensores incluyen la electrónica necesaria para los cálculos y directamente dan en su señal la orientación, liberando al control de estos cálculos tan específicos. Pueden medir la velocidad de giro del satélite (Rate Gyros RG) o desplazamientos angulares (Rate-Integrated Gyros RIG)[13]. Su funcionamiento se basa en la conservación del momento angular. El giróscopo es un disco que se hace girar sobre su eje y está montado sobre una suspensión Cardán. Cuando el satélite cambia de actitud la suspensión se mueve con él y el disco tiende a mantener su eje de giro inercial. Los ángulos de la cuna son las referencias que nos dan la actitud del satélite respecto de esa dirección de referencia. Aparte del giróscopo de Cardán, existe el giróscopo de correa o strap-down. Un problema general de los giróscopos es la deriva (drift). Las medidas del giróscopo no son absolutas sino relativas a un sistema de referencia, ya sea físico o matemático. Para giróscopos de Cardán este sistema es la propia plataforma del giróscopo, cuya orientación puede cambiar debido a la fricción en la dirección del Cardán. Para los giróscopos de correa, el sistema es el sistema calculado por inercia, que variará del verdadero debido a:

- Ruido en las medidas
- Discretización de las medidas del giróscopo en el ordenador

- Acumulación de errores de redondeo

La gran ventaja de los sensores es que proporcionan salida continua y no necesitan referencias exteriores como por ejemplo los sensores de estrellas. Tampoco se ven afectados por eclipses, variación del campo magnético, etc. Hay dos tipos de construcción: mecánica y láser. Estos últimos no son realmente giróscopos, sino que detectan la velocidad angular de un satélite basándose en la diferencia de caminos recorrido por dos haces láser en forma de anillo. Cuando el satélite gira, un camino es más corto que el otro y esta diferencia se aprecia en la fase de la onda del láser. Dicha diferencia se traduce en velocidad angular.



Figura 7.6: Imagen de un giróscopo

### 7.1.5. Magnetómetros

Miden las tres componentes del campo magnético. Son fiables, ligeros y requieren baja potencia. Carecen de partes móviles y tienen amplio rango de temperaturas de funcionamiento.

- Parámetros (todos en SI)

$\mathbf{B_I}$  Valor conocido del campo magnético en coordenadas inerciales

**S<sub>Bmag</sub>** Matriz de alineamiento del magnetómetro, transforma ejes magnetómetro a ejes cuerpo

**B<sub>mag</sub>** Campo magnético medido por el magnetómetro. Con un solo magnetómetro no se tienen datos suficientes, por lo que es necesario utilizar más de un magnetómetro.

**m** Valor del dipolo magnético terrestre

**r** Vector de posición del satélite

■ Señal de salida

**B<sub>B</sub>** Valor del campo magnético en ejes cuerpo

**C<sub>BI</sub>** Matriz de rotación

El vector de campo magnético terrestre se aproxima con el de un dipolo situado en el centro de la Tierra con la ecuación (7.6). Para más información consúltase [1].

$$\mathbf{B}_I = \frac{\mu_0}{4 * \pi} * \frac{3(\mathbf{m} \cdot \mathbf{r})\mathbf{r} - r^2\mathbf{m}}{r^5}, \quad (7.6)$$

$$\mathbf{B}_B = S_{Bmag} * \mathbf{B}_{mag}, \quad (7.7)$$

$$\mathbf{B}_B = C_{BI} * \mathbf{B}_I + \Delta\mathbf{B}_B. \quad (7.8)$$

La matriz de colocación  $S_{mag}$  se determina antes del lanzamiento pero es reestimada en el espacio. Con la ecuación (7.7) se calcula el campo magnético medido en ejes cuerpo. Finalmente con la ecuación (7.8) se estimaría la matriz de actitud.

Nota: es importante destacar que aunque el vector devuelto por el magnetómetro tiene tres componentes, sólo tiene dos grados de libertad. Por lo tanto no es suficiente una sola medida para determinar la actitud en los 3 ejes. Se necesitan por lo menos dos medidas de dos vectores. En esta ecuación se introduce ruido  $\Delta\mathbf{B}_B$  aleatorio, que en magnetómetros se debe principalmente al modelo de campo magnético ( $\mathbf{B}_I$ ). Para órbitas cercanas a la Tierra, el error en la dirección del campo modelado varía desde los 0.5 grados hasta los 3 grados cerca de los polos magnéticos. Lejos de la Tierra no es recomendable usar este tipo de sensor porque el campo magnético terrestre es demasiado

débil y es perturbado por el campo magnético interplanetario, muy difícil de modelar. Se recomienda su uso en los periodos de eclipse, cuando los sensores solares no reciben señal. [13]

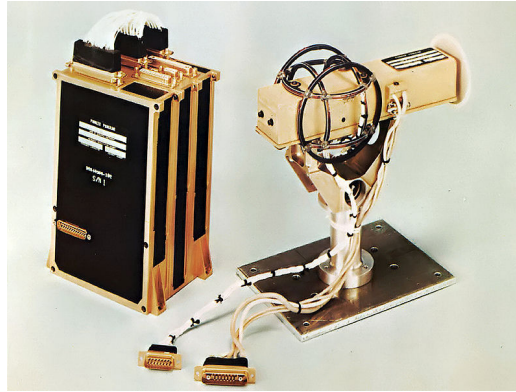


Figura 7.7: Magnetómetro instalado en la nave Pioneer 10 y 11

## 7.2. Programación

### 7.2.1. Selección y situación de sensores

En el programa Matlab, en el interfaz de actuadores con combustible se encuentra habilitado un botón *selección de sensores*. Pulsando este botón se carga el interfaz del catálogo de sensores (figura 7.8), similar a los catálogos de ruedas y propulsores. (Subsecciones 5.2.2 y 6.2.2 respectivamente).

La tabla de la base de datos local donde se guarda el catálogo se llama sensores, y se puede alimentar con las herramientas propias de la base de datos. Las operaciones que se pueden realizar sobre este catálogo son las mismas que las definidas en la subsección 5.2.2. En el interfaz de catálogo de sensores está el botón *situar* que llama al cuadro de *situar sensores*. Tanto la llamada como el funcionamiento del cuadro es idéntico al de propulsores (subsección 6.2.2).

Una vez seleccionado, situado y orientado un sensor, se guarda en la tabla *S\_colocado* un registro del siguiente tipo:

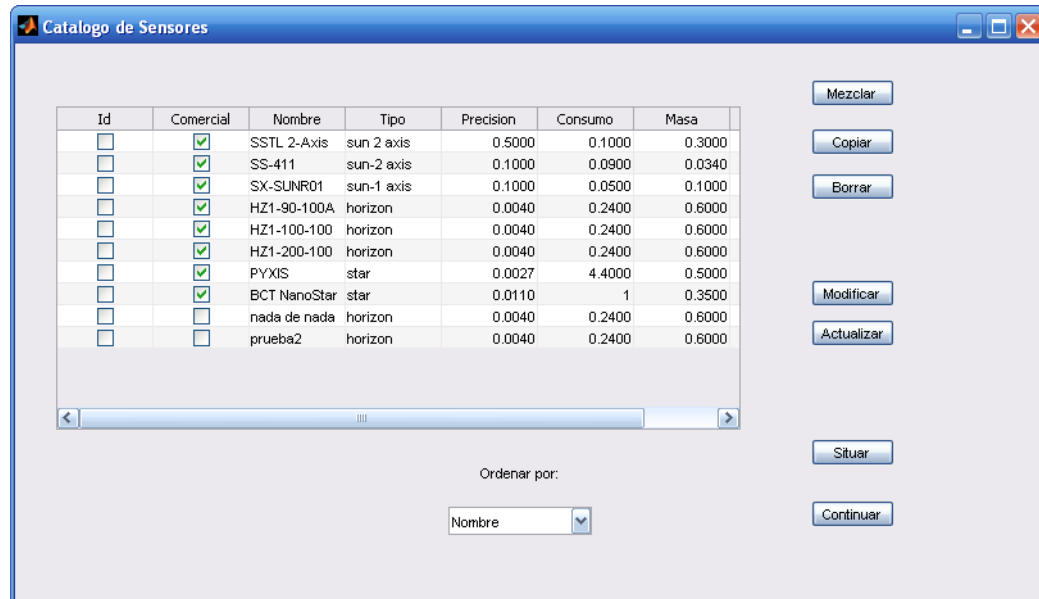


Figura 7.8: Catálogo de sensores

$$nuevo = \left\{ \begin{array}{c} \text{Nombre del sensor} \\ \text{Masa} \\ \text{Cara} \\ \text{Ángulo} \\ \text{Coordenada } x \\ \text{Coordenada } y \\ \text{Coordenada } z \end{array} \right\}.$$

El resto de los pasos es exactamente igual que en la selección y situación de propulsores.

### 7.2.2. Cálculos

#### Cálculo del centro de gravedad y tensor de inercia de los elementos colocados

Para realizar estos cálculos se ha utilizado la misma función *masas*, cuyo argumento es el nombre de la tabla donde están guardados los elementos con

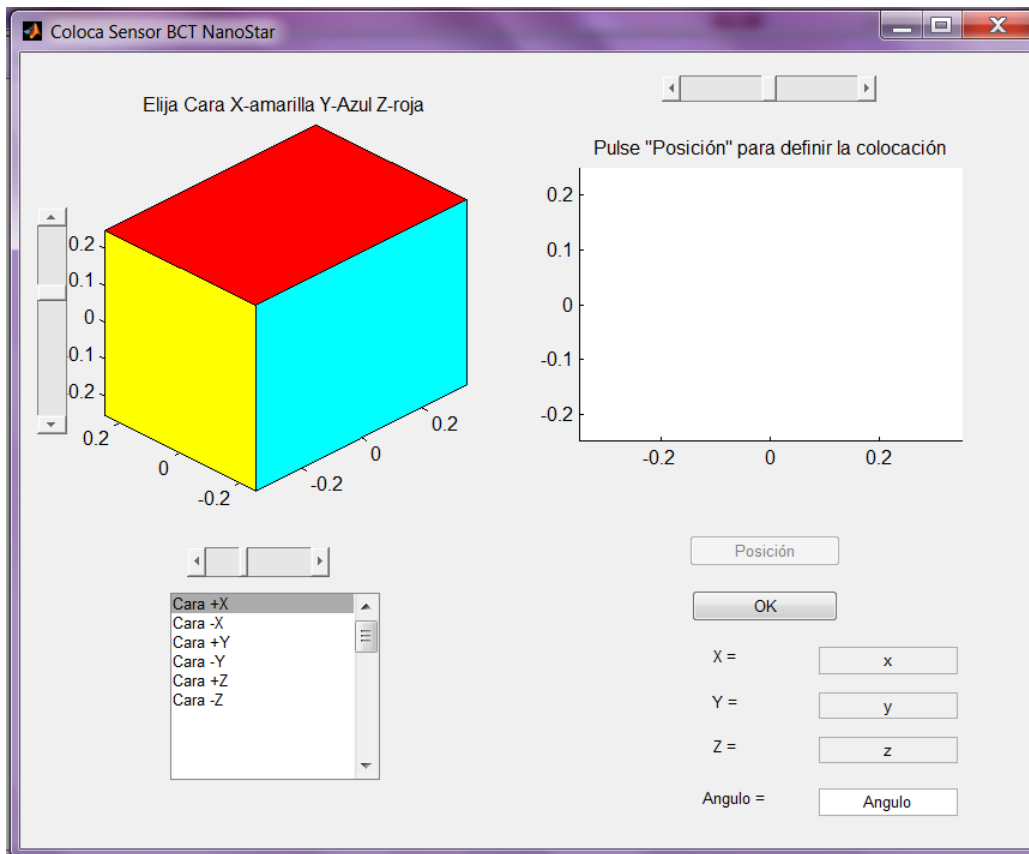


Figura 7.9: Interfaz para colocar un sensor. Selección de cara

sus masas y coordenadas. El funcionamiento de esta función se explica en el apartado 5.2.4.

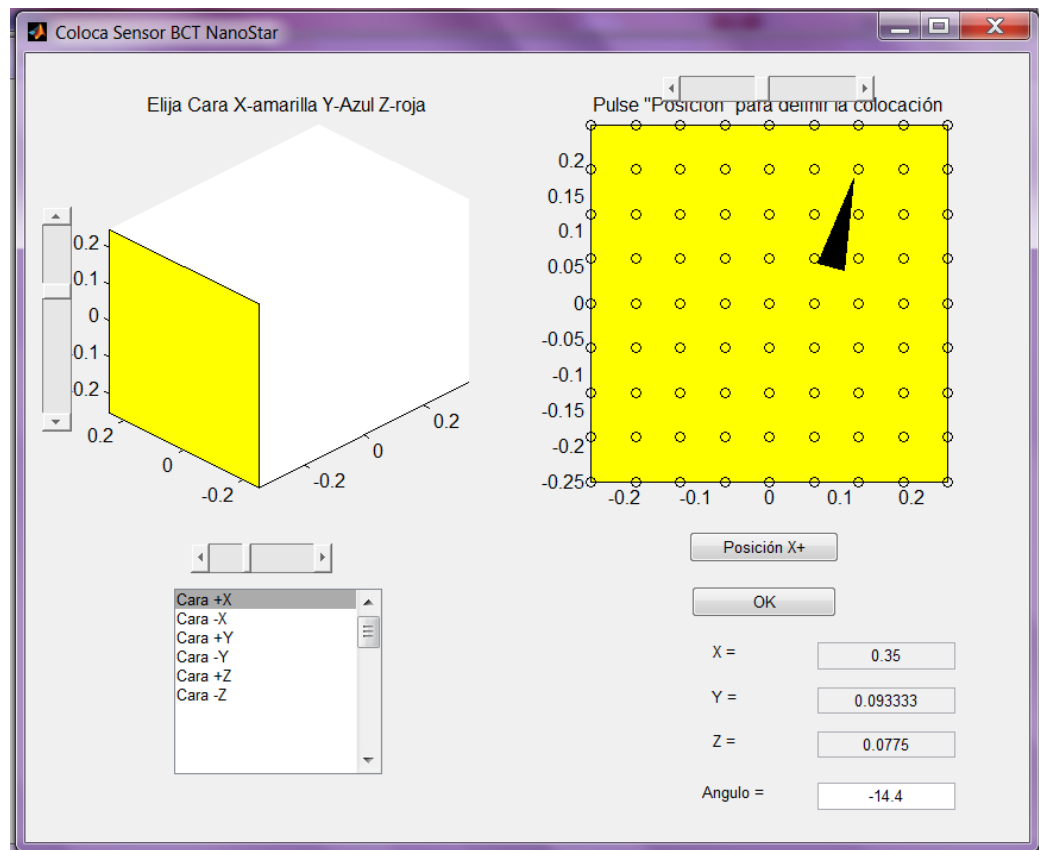


Figura 7.10: Interfaz para posicionar y orientar un sensor.



# Capítulo 8

## Conclusiones de la Parte II

### 8.1. Cumplimiento de objetivos

En esta segunda parte del proyecto los objetivos fueron la definición y cálculo aproximados de las perturbaciones de un satélite en órbita estacionaria, diseño de una interfaz de usuario para el diseño de la prefase-A y que fuera capaz de manipular los datos de esa interfaz.

Estos objetivos se han cumplido desarrollando los interfaces con la herramienta Guide de Matlab. Aunque en un principio no es cómodo ni intuitivo el desarrollo de aplicaciones de interfaz de usuario con esta herramienta, sí que es eficiente a la hora de manipular los datos. La disposición de datos con el formato de estructura hace que este desarrollo sea muy compacto y fácilmente ampliable.

Las fórmulas empleadas para realizar los cálculos son simplificadas y escalares. Coinciden en diferentes libros ([2, 10, 13, 16, 11]). Aunque no aparezca mencionado en los apartados de cálculos de los capítulos 4, 5 y 6, se han simulado los mismos ejemplos que vienen en las referencias [4, 16], obteniéndose los mismos resultados. Con lo cual se puede decir que este módulo es bueno para el predimensionado del sistema ADCS del satélite.

Como objetivo adicional se ha hecho una interfaz utilizando las posibilidades gráficas de Matlab. Las posibilidades gráficas de la herramienta son muy buenas, aunque para el programador es incómodo hacerlas amigables al usuario.

Otro objetivo conseguido es poder trabajar offline. Activando una sola línea de código en la función *Import\_DB3* se puede trabajar con la herra-

mienta aunque la base de datos no este disponible. Obviamente tampoco se pueden grabar externamente los resultados obtenidos.

## 8.2. Trabajos futuros

Se pueden añadir más tipos de componentes e información a la base de datos, el acceso a la misma se podría optimizar. Por ejemplo, se podrían añadir magnetopares y en el momento de consulta al catálogo se podría hacer de una forma filtrada, en base a los datos disponibles hasta el momento.

En el interfaz gráfico, a la hora de insertar un componente no se tienen en cuenta sus dimensiones. Se puede mejorar este punto sumando a las coordenadas dadas por el interfaz las coordenadas del centro de masas del elemento añadido. Lo mismo para su momento de inercia.

Para trabajar offline ahora se debe editar el código y activar una línea que está comentada. La mejora consistiría en poder hacer esto desde el interfaz o que se hiciera automáticamente y que los resultados quedasen a la espera de ser grabados en base de datos en el momento en que vuelva a haber conexión.

Durante el desarrollo de la programación de la herramienta, ha habido momentos en los que era necesaria una variable no accesible todavía de la base de datos. Se podría pensar en alguna forma de *workaround* para no retrasar el desarrollo mientras se recibe el acceso a las variables requeridas.

El programa actual sólo trata con satélites prismáticos. Una buena mejora sería hacer que aceptase diferentes formas del satélite, así como elementos auxiliares como paneles, antenas, etc.

Hay muchas mejoras de detalle aumentando el número de variables intercambiadas con los demás módulos. Por ejemplo, la altura de las bandejas para actuadores sin gas.

# Parte III

## Simulación



# Capítulo 9

## Sistemas de referencia

### 9.1. Introducción

Una vez concluido el diseño de prefase A el siguiente paso del proyecto es el estudio del comportamiento del satélite en el dominio del tiempo. Para ello es necesario tener valores tridimensionales de todos los vectores (vector de posición, velocidades, velocidades angulares, pares, fuerzas, etc.). En este proyecto se utilizará la herramienta Simulink para hacer el estudio en el tiempo junto con Matlab para representar las gráficas de los resultados.

### 9.2. Sistemas de referencia

#### 9.2.1. Ejes inerciales

Las leyes de la dinámica de Newton (mecánica clásica) se cumplen cuando las magnitudes se miden respecto a sistemas de referencia denominados inerciales. La característica de un sistema de referencia denominado inercial es que sus ejes apuntan en direcciones fijas (no cambian con el tiempo) y su origen está en reposo o movimiento rectilíneo uniforme. Pero como el reposo absoluto no existe (o no se puede definir) se aceptan como un origen inercial el centro del Sol (sistema heliocéntrico) o el centro de la Tierra (sistema geocéntrico). Para elegir las direcciones fijas se acepta por convenio que las estrellas ocupan lugares fijos. En los dos sistemas inerciales mencionados anteriormente se toma como eje  $x$  el primer punto de Aries (Llamado  $\Upsilon$ ), definido como la línea Tierra-Sol en equinoccio vernal. En el sistema helio-

céntrico se toma como eje  $z$  la perpendicular al plano de la eclíptica en el sentido positivo del giro de traslación de la Tierra alrededor del sol. En el sistema geocéntrico se toma como eje  $z$  el eje de giro de rotación de la Tierra. Este eje apunta a la Estrella Polar. El eje  $y$  en ambos sistemas es el que forma un triedro a derechas con los ejes  $xz$ . Se muestra un resumen en cuadro 9.1 e ilustraciones en figuras 9.1 y 9.2.

	Dirección	
	Sistema heliocéntrico	Sistema geocéntrico
x	Primer punto Aries $\Upsilon$	Primer punto Aries $\Upsilon$
y	Triedro a derechas con $x,z$	Triedro a derechas con $x,z$
z	$\omega_{\text{traslación}}$ de la Tierra	$\omega_{\text{rotación}}$ de la Tierra

Cuadro 9.1: Sistemas de referencia inerciales

En el caso particular del presente proyecto en el que se simulan órbitas alrededor de la Tierra el sistema de referencia adecuado es el geocéntrico y se considera que el Sol gira alrededor de la Tierra.

### 9.2.2. Ejes órbita

Como las órbitas se encuentran siempre contenidas en un plano (suele ser así para los problemas generales de actitud) a veces suele resultar útil un sistema de coordenadas referido a este plano. En este sistema el origen es el foco de la órbita (la Tierra en el caso de un satélite). El eje  $z$  es el perpendicular al plano de la órbita. El eje  $x$  es el que apunta al pericentro de la órbita y el eje  $y$  perpendicular a los dos anteriores formando un triedro a derechas. Véase figura 9.3.

### 9.2.3. Ejes trayectoria

En estos ejes el origen es el cuerpo en movimiento (el satélite en este caso). El eje  $z$  apunta al foco de la trayectoria (centro de la Tierra, también se le conoce como nadir). El eje  $y$  es perpendicular a la trayectoria y el eje  $x$  perpendicular a los anteriores (contenido en el plano de la órbita) y en el sentido de avance. Los tres forman un triedro a derechas. El ángulo entre la tangente a la trayectoria y el eje  $x$  se denomina en inglés *flight-path-angle* y se calcula [13]:

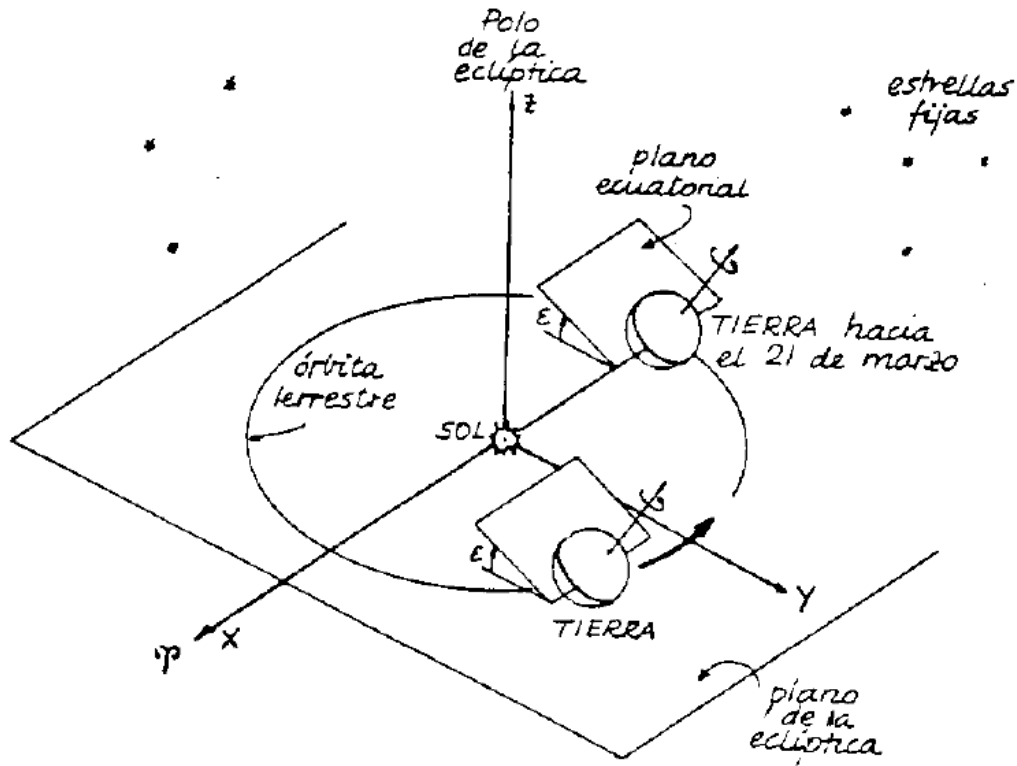


Figura 9.1: Sistema de referencia inercial heliocéntrico  
Extraído de [13]

$$\gamma = \arctan \left[ \left( 1 - \frac{r}{p} \right) \tan \nu \right].$$

Siendo  $r$  el radio vector que une el satélite con el astro,  $p$  el parámetro de la cónica y  $\nu$  la anomalía verdadera. Los tres parámetros se verán de nuevo en el capítulo 11.

#### 9.2.4. Ejes cuerpo

Son unos ejes sólidamente unidos al cuerpo con su origen en el centro de gravedad. Si la misión del satélite consiste en apuntar a algún punto en particular (ya sea Nadir o a una estrella fija) éste será el eje  $z$ . Para estabilizar la actitud del satélite, éste normalmente gira alrededor del eje  $z$ .

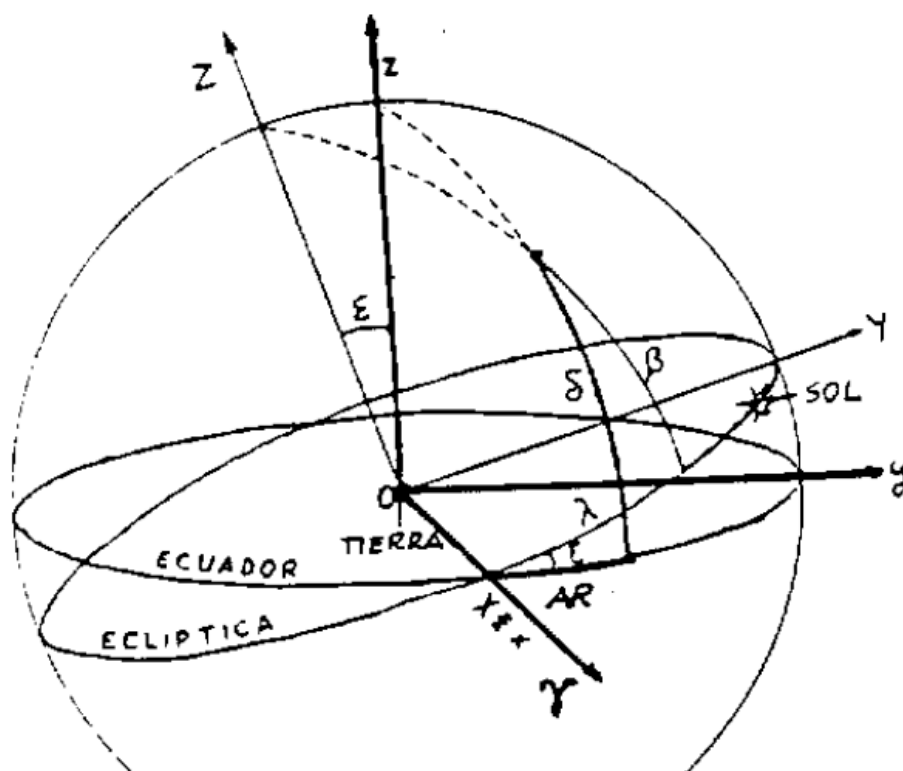


Figura 9.2: Sistema de referencia inercial geocéntrico  
 Extraído de [13].  $\varepsilon = 23,45^\circ$ , corresponde a la inclinación de la órbita solar en este sistema.

Por tanto es muy importante que este eje sea principal de inercia. De otro modo el satélite vibraría naturalmente y el sistema de control de actitud estaría trabajando innecesariamente y sin conseguir su objetivo. El eje  $x$  y el eje  $y$  se eligen de forma que se considere oportuna, puede establecerse de forma que el eje  $x$  apunte a la dirección de la trayectoria, pero en el caso de que esté autoestabilizado por spin dicho eje apuntará hacia un punto distinto en cada momento. Si el satélite tiene forma prismática, lo normal es que apunten perpendicularmente a las caras. Ilustración en figura 9.4.



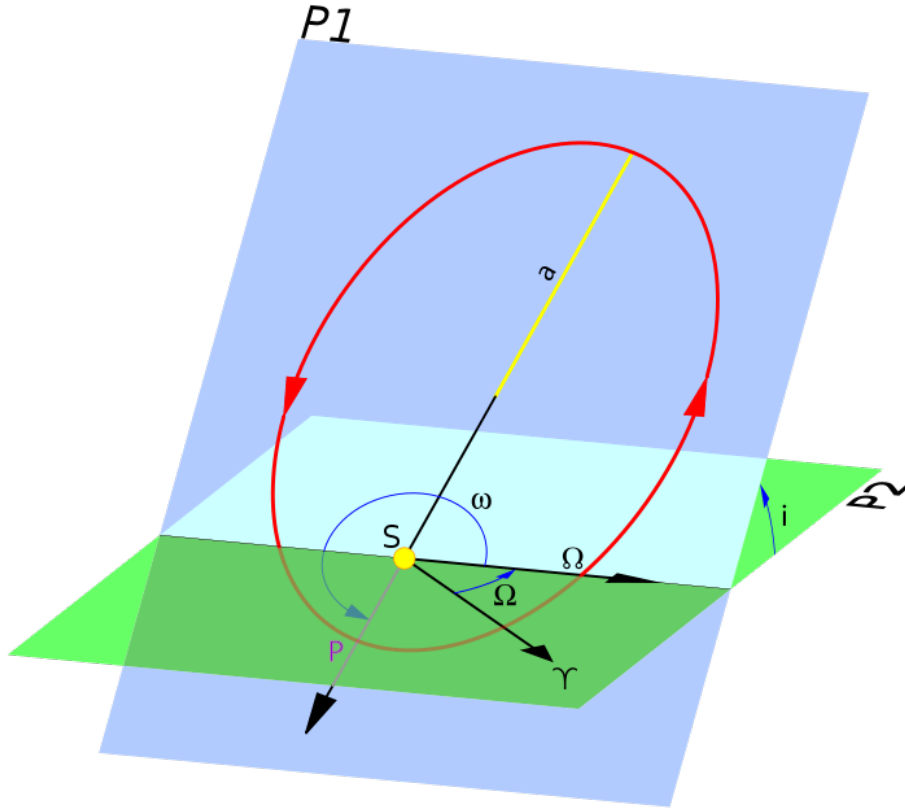


Figura 9.3: Sistema de referencia no inercial asociado al plano de la órbita

### 9.3. Matrices de rotación

Para poder trasladar unas magnitudes de unos sistemas de referencia a otros hace falta un operador que es la matriz de rotación entre sistemas.

Supónganse dos sistemas de referencia 1 y 2. Ambos comparten el eje  $z$ , y el sistema 2 se consigue girando el triedro de 1 un ángulo  $\Omega$  alrededor del eje  $z$ . De este modo los vectores unitarios del sistema 2 referidos al sistema 1 son:

$$\mathbf{i}_2 = \cos \Omega * \mathbf{i}_1 + \sin \Omega * \mathbf{j}_1,$$

$$\mathbf{j}_2 = -\sin \Omega * \mathbf{i}_1 + \cos \Omega * \mathbf{j}_1,$$

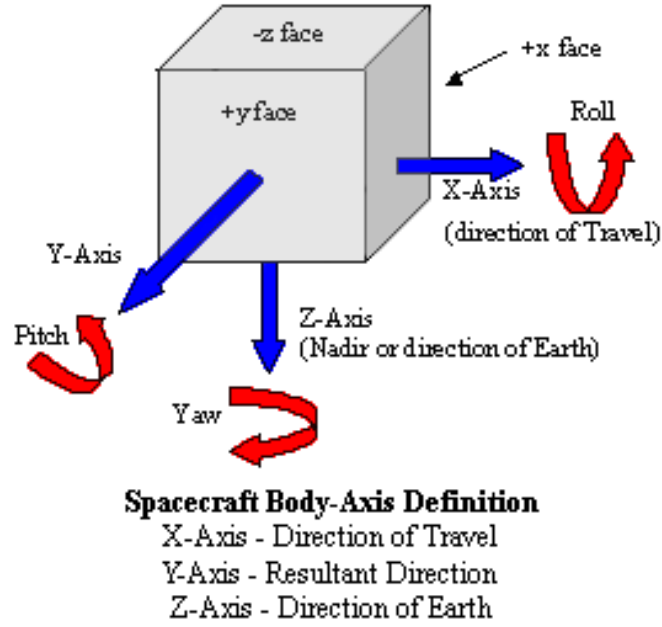


Figura 9.4: Sistema de referencia no inercial en ejes cuerpo

$$\mathbf{k}_2 = \mathbf{k}_1.$$

Si se define la matriz  $Q_{21}$  de modo que  $\mathbf{v}_2 = Q_{21}\mathbf{v}_1$  siendo  $\mathbf{v}_2$  y  $\mathbf{v}_1$  el mismo vector definido en los sistemas 2 y 1 respectivamente. La matriz  $Q_{21}$  será entonces:

$$Q_{21} = \begin{bmatrix} \cos \Omega & \sin \Omega & 0 \\ -\sin \Omega & \cos \Omega & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

De forma análoga si el define un sistema de referencia 3 que se consigue girando el sistema 2 alrededor del eje x un ángulo  $i$  la matriz  $Q_{32}$  será

$$Q_{32} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos i & \sin i \\ 0 & -\sin i & \cos i \end{bmatrix}.$$

Para pasar de los ejes inerciales a unos ejes órbita se realizan estos dos giros seguidos por un tercer giro de nuevo sobre el eje z con un ángulo  $\omega$ .

De este modo  $Q_{43}$  será:

$$Q_{43} = \begin{bmatrix} \cos \omega & \sin \omega & 0 \\ -\sin \omega & \cos \omega & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Los ejes conseguidos con estos tres giros a partir del sistema geocéntrico son los ejes órbita, definida ésta por los ángulos de Euler:

$\Omega$  Argumento del nodo ascendente

$i$  Inclinação de la órbita

$\omega$  Argumento del perigeo

Y la matriz  $Q_{41} = Q_{43} * Q_{32} * Q_{21}$  es la matriz que transforma cualquier vector de ejes inerciales (1) a ejes órbita (4).  $\mathbf{v}_4 = Q_{41}\mathbf{v}_1$ . De este modo la matriz  $Q_{41}$  queda:

$$Q_{41} = \begin{bmatrix} \cos \omega \cos \Omega - \sin \omega \cos i \sin \Omega & \cos \omega \sin \Omega + \sin \omega \cos i \cos \Omega & \sin \omega \sin i \\ -\sin \omega \cos \Omega - \cos \omega \cos i \sin \Omega & -\sin \omega \sin \Omega + \cos \omega \cos i \cos \Omega & \cos \omega \sin i \\ \sin i \sin \Omega & -\sin i \cos \Omega & \cos i \end{bmatrix}. \quad (9.1)$$

Los tres giros consecutivos de la matriz (9.1) se ilustran en la figura 9.5.

Como las 3 matrices son ortogonales, el producto de ellas también lo es y su inversa es su traspuesta.  $\mathbf{v}_1 = Q_{41}^T \mathbf{v}_4$ .

Esta forma de conseguir matrices de rotación a base de giros sucesivos es una manera intuitiva de operar, pero tiene gran carga de cálculo y puede llevar a indeterminaciones. Es por esto que se existe otra forma de cálculo para el tratamiento de los giros. Esto nos lleva al siguiente apartado.

## 9.4. Cuaterniones y rotación espacial

### 9.4.1. Introducción

Del mismo modo que el campo de los números complejos se define añadiendo el símbolo abstracto  $i$  que satisface la ecuación  $i^2 = -1$ , y con esta regla se puede deducir toda la aritmética del cuerpo de los complejos, se

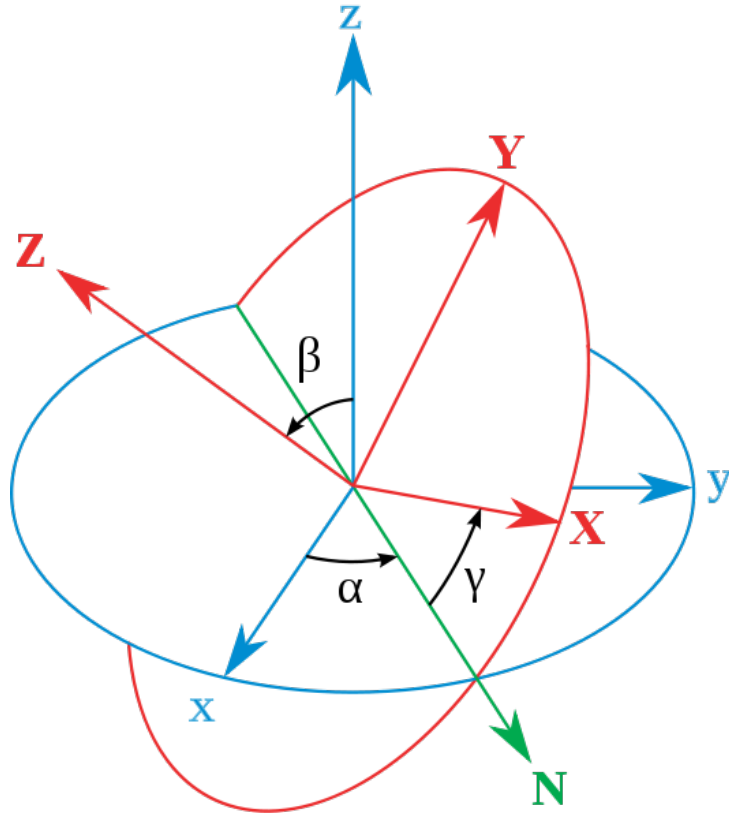


Figura 9.5: Ángulos de rotación de Euler zxz  
En la ilustración el ángulo  $\alpha$  corresponde a  $\Omega$ ,  $\beta$  a  $i$  y  $\gamma$  a  $\omega$ .

define los cuaterniones con tres entidades abstractas  $i, j, k$  que satisfacen la ecuación:

$$i^2 = j^2 = k^2 = ijk = -1.$$

Un cuaternión es un elemento del espacio tetradimensional definido de la forma

$$q = q_0 + q_1 * i + q_2 * j + q_3 * k.$$

Siendo  $q_0, q_1, q_2, q_3$  números reales. Un cuaternión se comporta como un número sumado a un vector del espacio tridimensional. Para más información se puede consultar la referencia [7].

### 9.4.2. Rotación

Cualquier rotación se puede representar por un cuaternión de norma unidad. Intuitivamente es sencillo convertir un giro en un cuaternión.

Sean dos sistemas de referencia A y B con origen común. Se puede convertir el triedro A en el triedro B girando un ángulo  $\phi$  alrededor de cierto eje de dirección  $\hat{a}$  ( $\hat{a} = (a_x, a_y, a_z)$ ), ver figura 9.6. Este eje cumple la propiedad de que sus componentes (o cosenos directores) son los mismos en ambos sistemas de referencia. El cuaternión que representa este giro es

$$q = \begin{bmatrix} \cos\left(\frac{\phi}{2}\right) \\ a_x \sin\left(\frac{\phi}{2}\right) \\ a_y \sin\left(\frac{\phi}{2}\right) \\ a_z \sin\left(\frac{\phi}{2}\right) \end{bmatrix} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}. \quad (9.2)$$

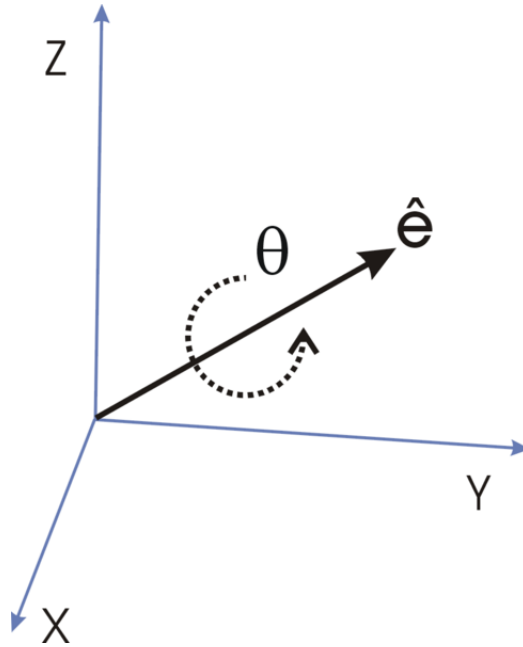


Figura 9.6: Representación de rotación alrededor de un eje  
El vector  $\hat{a}$  descrito se representa en la figura como  $\hat{e}$  y el ángulo  $\phi$  está representado como  $\theta$

Se define  $\mathbf{q}$  como la parte vectorial del cuaternión:

$$\mathbf{q} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix}. \quad (9.3)$$

Desarrollando el giro la matriz ortogonal equivalente es R:

$$R = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_0^2 & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_2q_1 - q_0q_3) & -q_1^2 + q_2^2 - q_3^2 + q_0^2 & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & -q_1^2 - q_2^2 + q_3^2 + q_0^2 \end{bmatrix}. \quad (9.4)$$

Imponer como se dijo al principio de esta subsección que este cuaternión tiene norma uno es equivalente a decir que  $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$ .

### 9.4.3. Operaciones con cuaterniones

Las operaciones más importantes utilizadas en el presente proyecto son:

- Cambios del sistema de referencia (R definido en (9.4))

$$v_2 = R * v_1.$$

- Derivada de un cuaternión. Si un sistema de referencia está girando con una velocidad angular  $\omega$  el cuaternión correspondiente al giro de este sistema de referencia respecto a unos ejes inerciales, y  $\mathbf{q}$  está definido por (9.3), entonces la derivada es:

$$\frac{d}{dt}q_0 = -\frac{1}{2}\omega \cdot \mathbf{q}, \quad (9.5)$$

$$\frac{d}{dt}\mathbf{q} = \frac{1}{2}(q_0\omega - \omega \times \mathbf{q}). \quad (9.6)$$

Es posible que con los errores de cálculo y redondeo la norma del cuaternión se desvíe de 1. Por ello en los cálculos se suele normalizar el cuaternión antes de operar con él.

$$q_n = \frac{1}{\sqrt{q^T q}} q.$$

# Capítulo 10

## Modelo dinámico

### 10.1. Conceptos teóricos

Para estudiar la actitud del satélite, se considera éste como un sólido rígido. La ecuación que rige la dinámica de un sólido rígido en ejes inerciales es:

$$\frac{d}{dt}\mathbf{L} = \mathbf{N}. \quad (10.1)$$

Siendo

$\mathbf{L}$  Momento angular.

$\mathbf{N}$  Momentos de las fuerzas exteriores

$\boldsymbol{\omega}_B$  Velocidad de rotación de los ejes cuerpo respecto de los ejes inerciales.  
Se usará a continuación.

$\mathbf{I}_B$  Tensor de inercia referido a ejes cuerpo (tiene la ventaja de que no varía con el tiempo si el sólido es rígido). También se usará a continuación.

Sin embargo, para el estudio del cuerpo se suelen emplear ejes unidos al mismo, con origen en su centro de masas. Dicho sistema se encuentra definido en 9.2.4. La ecuación (10.1) referida a ejes cuerpo queda entonces:

$$\frac{d}{dt}\mathbf{L}_B + \boldsymbol{\omega}_B \times \mathbf{L}_B = \mathbf{N}_B. \quad (10.2)$$

El momento angular en ejes cuerpo queda de la forma:

$$\mathbf{L}_B = \mathbf{I}_B \boldsymbol{\omega}. \quad (10.3)$$

Sustituyendo (10.3) en (10.2) y teniendo en cuenta que el tensor de inercia  $\mathbf{I}_B$  no varía con el tiempo se obtiene finalmente:

$$\mathbf{I}_B \frac{d}{dt} \boldsymbol{\omega} + \boldsymbol{\omega} \times (\mathbf{I}_B \boldsymbol{\omega}) = \mathbf{N}_B. \quad (10.4)$$

Para más información y mayor desarrollo se puede consultar cualquier libro de mecánica clásica, por ejemplo [15].

Lo que se pretende en este capítulo es llevar este modelo simple a Simulink para obtener la dinámica del satélite.

## 10.2. Modelo Simulink de la dinámica del satélite

### 10.2.1. Descripción del subsistema de cálculo de velocidad angular

El subsistema Simulink para que satisface la ecuación (10.4) es que aparece en la figura 10.1. En realidad se ha despejado  $\dot{\boldsymbol{\omega}}$  de la ecuación (10.4) de esta forma:

$$\dot{\boldsymbol{\omega}} = \mathbf{I}_B^{-1} (\mathbf{N}_B + (\mathbf{I}_B \boldsymbol{\omega}) \times \boldsymbol{\omega}).$$

#### ■ Input

- Variables en función del tiempo

**Par\_c** Par de control

**Par** Suma de pares de perturbación

- Constantes

**Inercia** Tensor de inercia en ejes cuerpo

- Parámetros

**w0** Velocidad angular tomada en el instante inicial

#### ■ Output



$\omega$  Velocidad angular del sistema de referencia y el satélite

El funcionamiento del subsistema es el siguiente: teniendo despejada  $\dot{\omega}$  en función de  $\omega$  se coloca el integrador, cuya entrada y salida van a corresponder con estas dos variables. Se tiene en el rectángulo central el tensor de inercia (constante). Se multiplica matricialmente por  $\omega$ . El resultado se multiplica vectorialmente por  $\omega$ . A este nuevo resultado se le suman los pares exteriores perturbaciones y control ( $par\_c$  y  $par$ ). Lo último que queda es multiplicar matricialmente la inversa del tensor de inercia por este resultado e introducirlo al integrador.

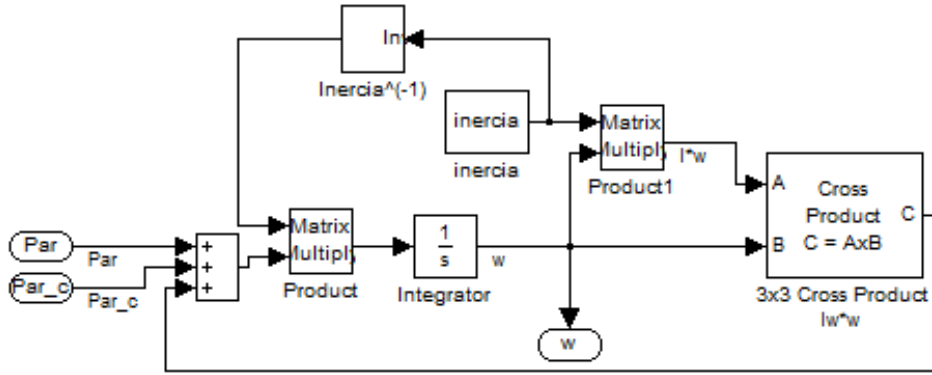


Figura 10.1: Subsistema de respuesta dinámica

### 10.2.2. Comprobación de modelo de cálculo de velocidad angular

Para comprobar la validez de este subsistema se ha ejecutado la simulación para el caso simplificado (siguiendo la idea de [10]) en el que imponemos dos hipótesis:

1.  $\mathbf{I}_B$  es un tensor de revolución, lo que quiere decir que  $I_x = I_y$  y que es matriz diagonal
2.  $\mathbf{N}_B = \mathbf{0}$  Lo que quiere decir que no se aplican pares exteriores.

Estas dos hipótesis simplifican la ecuación (10.4) en los tres ejes:

$$I_x \dot{\omega}_x + (I_z - I_x) \omega_y \omega_z = 0,$$

$$I_x \dot{\omega}_y + (I_x - I_z) \omega_z \omega_x = 0,$$

$$I_z \dot{\omega}_z = 0.$$

Cuya solución es:

$$\omega_z = cte,$$

$$\ddot{\omega}_x + \lambda^2 \omega_x = 0,$$

$$\ddot{\omega}_y + \lambda^2 \omega_y = 0.$$

Siendo

$$\lambda = \frac{I_x - I_z}{I_x} \omega_z,$$

La solución es

$$\omega_x = \omega_T \cos(\lambda t + \varphi),$$

$$\omega_y = \omega_T \sin(\lambda t + \varphi).$$

Donde  $\omega_T = \sqrt{\omega_{x0}^2 + \omega_{y0}^2}$  y  $\varphi$  una constante de desfase que depende de los valores iniciales de  $\omega_0$ .

Haciendo el ejemplo numérico en el que introducimos los parámetros indicados en el cuadro 10.1, se obtienen los resultados obtenidos en ese mismo cuadro.

Haciendo una simulación con estos parámetros se observa que los resultados se cumplen (Ver figura 10.2)

Parámetro	Valor	Resultado	Valor
$I_x$	40	$\lambda$	$-\frac{5}{2}$
$I_y$	40	T	$\frac{4\pi}{5} \simeq 2,5$
$I_z$	60	$\omega_T$	3
$\omega_{x0}$	3		
$\omega_{y0}$	0		
$\omega_{z0}$	5		

Cuadro 10.1: Valores del ejemplo numérico para comprobación

### 10.2.3. Descripción del subsistema de cálculo de actitud

El modelo anterior calcula la velocidad angular del cuerpo a lo largo del tiempo. Lo que se necesita conocer es la actitud del satélite, que no es más que integrar dicha velocidad angular. Utilizando las ecuaciones 9.5 y 9.6 se consigue el cuaternión que determina la actitud. (Ver figura 10.3). Se vuelven a escribir las ecuaciones para comodidad del lector:

$$\frac{d}{dt}q_0 = -\frac{1}{2}\boldsymbol{\omega} \cdot \mathbf{q}, \quad (10.5)$$

$$\frac{d}{dt}\mathbf{q} = \frac{1}{2}(q_0\boldsymbol{\omega} - \boldsymbol{\omega} \times \mathbf{q}). \quad (10.6)$$

Para explicar este subsistema que lo que hace es cumplir una ecuación diferencial, lo primero que se debe hacer es despejar la derivada de la variable a estudiar. Se coloca el integrador, cuya salida es la variable y la entrada es su derivada (*Integrator*). En este caso el cuaternión es un vector de cuatro componentes. La primera es escalar y las otras tres forman un vector. De este modo con un desmultiplexor se descompone las 4 componentes de  $\mathbf{q}$  y con un multiplexor se agrupa la parte vectorial. Haciendo el producto escalar de la entrada  $\boldsymbol{\omega}$  ( $\mathbf{w}$ ) por la parte vectorial del cuaternión y multiplicando por -0.5 se tiene la derivada de la parte escalar. (Ecuación (10.5)).

En la parte vectorial (Ecuación (10.6)) lo que se hace es multiplicar la entrada  $\boldsymbol{\omega}$  ( $\mathbf{w}$  en el bloque) por la componente escalar del cuaternión. Por otra parte se hace el producto vectorial de  $\boldsymbol{\omega}$  y la parte vectorial del cuaternión. Se suman los dos resultados y se divide por dos. Con eso se tiene la derivada de la parte vectorial del cuaternión. Se concatenan la parte escalar con la vectorial y se tiene la derivada completa del cuaternión, que es la que se

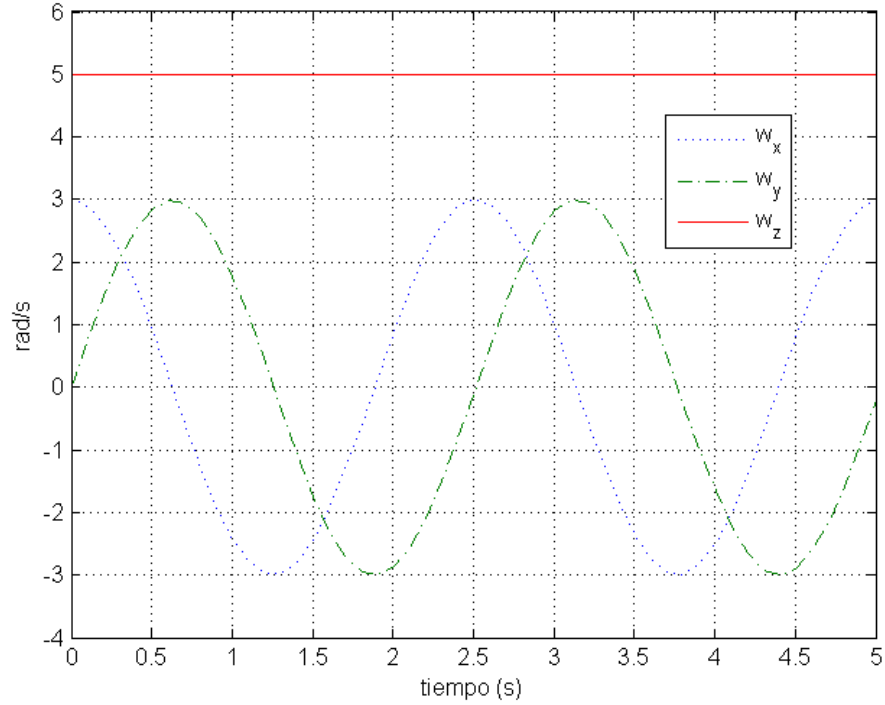


Figura 10.2: Resultado de la simulación de prueba  
Se observa que  $\omega_z = 5$  constante y  $\omega_x$  y  $\omega_y$  oscilan con periodo de 2.5 segundos.

introduce en el integrador.

De este modo la entrada de este módulo es  $\omega$  y el parámetro de actitud inicial que entra como estado inicial del integrador. Este valor se calcula en el apartado 10.2.1.

#### 10.2.4. Integración de subsistema cálculo de velocidad angular y cálculo de actitud

En la figura 10.4 se observan los dos modelos anteriores dentro del modelo total. El modelo dinámico (figura 10.1) corresponde al rectángulo coloreado de la izquierda y el subsistema de cálculo de actitud (figura 10.3) es el rectángulo colorado de la derecha. La salida del primero ( $\omega$ ) es la entrada superior del segundo. Se observa que la salida del segundo ( $q$ , aparece como actitud en

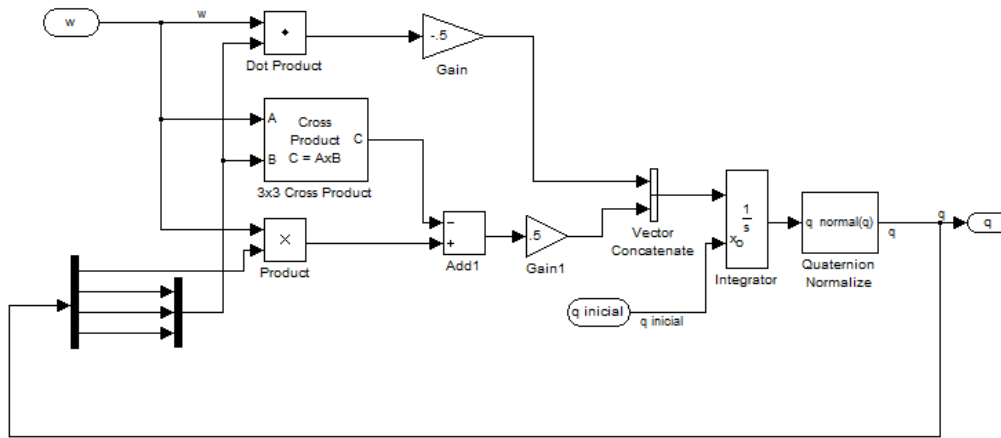


Figura 10.3: Subsistema que calcula el cuaternión que determina la actitud

la figura 10.4) se almacena en un *goto* (situado en la librería de signal routing en el menú de Simulink) para luego poder utilizarlo en múltiples bloques sin necesidad de unirlos con líneas.

En la misma figura se observa el bloque de control y la otra entrada del bloque dinámico que corresponde a la suma de todos los pares de perturbaciones.

El bloque de control que se ve en esta figura se explica en el apartado 13.2.3.

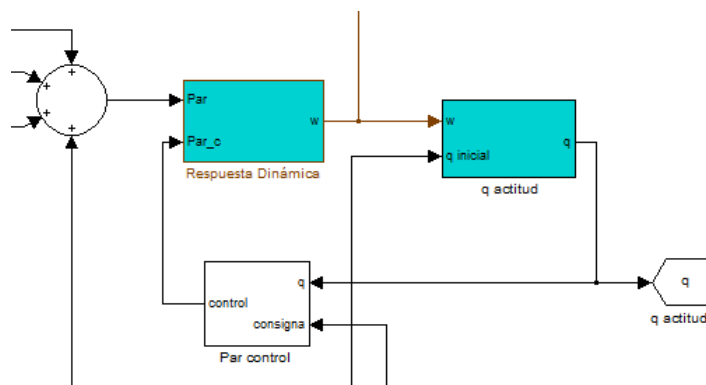


Figura 10.4: Integración del sistema dinámico y cálculo de actitud



# Capítulo 11

## Simulación de órbitas

### 11.1. Conceptos teóricos

#### 11.1.1. Órbita en su plano

Para el cálculo de la actitud del satélite es necesario saber su posición, velocidad y orientación respecto a otros astros, en este caso el Sol. Como se ha escogido el sistema geocéntrico como inercial (subsección 9.2.1) tanto el Sol como el satélite describen órbitas alrededor de la Tierra. El primero lo hace con la constante gravitacional de la Tierra ( $\mu_T = GM_T$ ) y el segundo con la constante gravitacional del Sol ( $\mu_\odot = GM_\odot$ ). Así mismo se considera que el satélite describe una órbita cerrada. La ecuación que da la ley horaria de un movimiento orbital elíptico es [12]:

$$\dot{E} (1 - e \cos E) = \sqrt{\frac{\mu}{a^3}}. \quad (11.1)$$

Siendo

**E** Anomalía excéntrica

**a** Semieje mayor de la órbita

**e** Excentricidad de la órbita ( $0 \leq e < 1$ )

La ecuación que da el módulo del radio vector  $r$  en una órbita elíptica es:

$$r = a (1 - e \cos E). \quad (11.2)$$

La anomalía verdadera se relaciona con la anomalía excéntrica con la ecuación[13]:

$$\tan\left(\frac{\nu}{2}\right) = \sqrt{\frac{1+e}{1-e}} \tan\left(\frac{E}{2}\right). \quad (11.3)$$

Por último, para la presente simulación se necesita conocer la velocidad en función de los datos anteriores:

$$v_x = \dot{x} = -a\dot{E} \sin E, \quad (11.4)$$

$$v_y = \dot{y} = a\sqrt{1-e^2}\dot{E} \cos E. \quad (11.5)$$

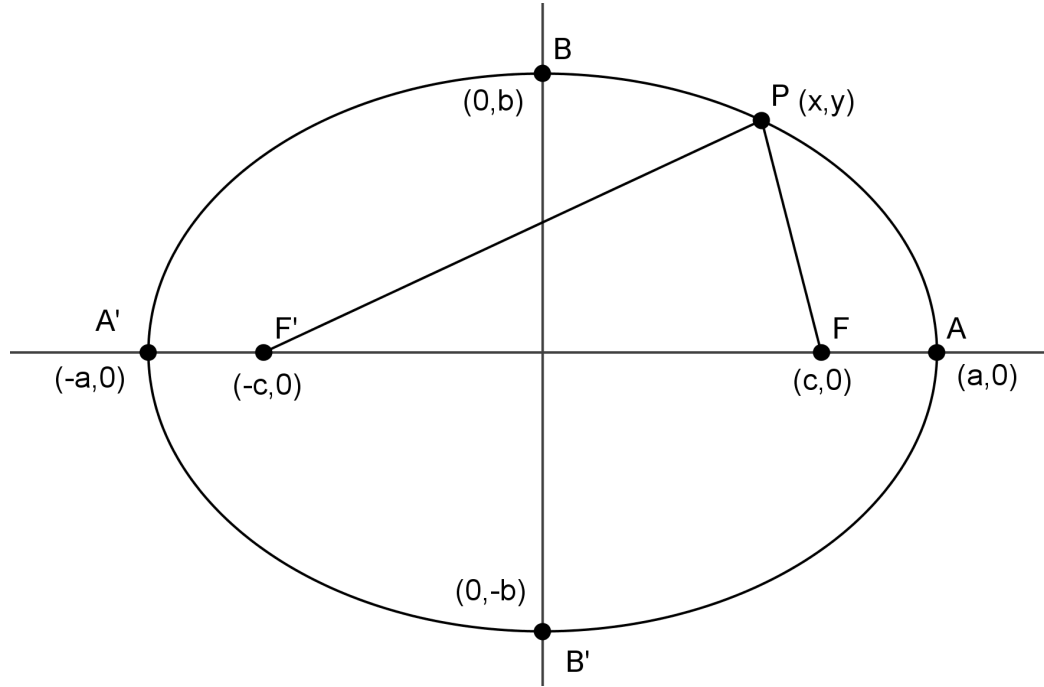


Figura 11.1: Parámetros de la elipse

### 11.1.2. Paso de órbita a sistema inercial

Todas las fórmulas del apartado anterior están referidas al sistema de ejes órbita (subsección 9.2.2). Dando los parámetros del sistema del plano de la



órbita se traducirán a sistema de ejes inerciales inerciales. Así la órbita del Sol se considera circular ( $e = 0,0167$ ) con una inclinación  $i = 23,45^\circ$  y la longitud del nodo ascendente  $\Omega = 0$ .

### 11.1.3. Eclipse

Para el desarrollo de la misión del satélite es importante saber cuándo éste se encuentra dentro de la zona de sombra provocada por la Tierra [13]. Se utiliza para saber cuándo son operativas las placas solares o para el sistema de refrigeración, entre otros. En este proyecto en particular, se necesita para el cálculo del par de perturbación solar, explicado en apartado 12.3. Los datos necesarios para calcular un eclipse son:

- Input

$\mathbf{r}$  Radio vector del satélite

$\mathbf{r}_\odot$  Radio vector solar

$R_T$  Radio terrestre (6378 km)

- Output

**Eclipse** Variable true/false que determina si el satélite está en zona de sombra o no

Según figura 11.3 para que el satélite esté en zona de eclipse tienen que cumplirse dos condiciones (suponiendo que  $\mathbf{r}_\odot \gg \mathbf{r}$  y por lo tanto que el Sol se encuentra en el infinito) :

1. Ángulo A formado por el radio vector del satélite y el radio vector del Sol sea obtuso.

$$\mathbf{r} \cdot \mathbf{r}_\odot < 0.$$

2. Proyección del radio vector del satélite sobre el plano que pasa por el centro de la Tierra y es perpendicular a la dirección del Sol es menor que el radio de la Tierra.

$$|\mathbf{r} \times \mathbf{u}_\odot| = r \sin B < R_T.$$

## 11.2. Modelo Simulink de cálculo de órbita de satélite

### 11.2.1. Modelo Simulink de cálculo de órbita en ejes órbita

Con el siguiente modelo (figura 11.5) se calcula:

$\mathbf{r}$  radio vector

$\nu$  anomalía verdadera

$\mathbf{v}_x$  velocidad en eje x órbita

$\mathbf{v}_y$  velocidad en eje y órbita

Para ello necesita los parámetros

$a$  Semieje mayor

$e$  Excentricidad

$\mu$  Constante gravitacional

$E_0$  Anomalía excéntrica en el instante inicial

El funcionamiento del subsistema es el siguiente: lo primero que se resuelve es la ecuación diferencial despejando  $\dot{E}$  de la ecuación (11.1).

$$\dot{E} = \frac{n}{1 - e \cos E}, \quad (11.6)$$

Siendo

$$n = \sqrt{\frac{\mu}{a^3}}.$$

La salida del integrador es  $E$ . Se calcula su coseno y se multiplica por la excentricidad. Se resta esta cantidad de 1 y  $n$  se divide por este resultado. Esto es el valor de  $\dot{E}$  ( $E_p$ ) que es la entrada al integrador. El denominador de la ecuación (11.6) se multiplica por  $a$  y da de resultado el módulo del radio vector, como se indica en la ecuación (11.2).

De la salida del integrador se saca  $E$ , que multiplicándola por 0.5 y calculando la tangente se tiene un factor de la ecuación (11.3). El otro factor se

consigue sumando  $1+e$ , restando  $1-e$ , dividiendo y extrayendo la raíz cuadrada. Multiplicando ambos factores, calculando el arcotangente y multiplicando por 2 se obtiene la anomalía verdadera  $\nu$  (`true_anom`).

Para obtener  $v_x$  se toma la entrada del integrador  $\dot{E}$ , se multiplica por  $-a$  y por el seno de  $E$ , cumpliendo la ecuación (11.4).

Para obtener  $v_y$  se multiplica la entrada al integrador  $\dot{E}$  por  $a$  y por el coseno de  $E$ . Aprovechando que se tiene  $1+e$  y  $1-e$  de los cálculos de la anomalía verdadera, se multiplican para obtener  $1 - e^2$  y se extrae la raíz cuadrada. Es el último factor que falta para obtener  $v_y$  según la ecuación (11.5).

### 11.2.2. Modelo Simulink de transferencia de ejes órbita a ejes inerciales

Los cálculos del apartado anterior están realizados en el sistema de referencia de ejes órbita y es necesario pasarlo al sistema de referencia de ejes inerciales y posteriormente a ejes cuerpo. La idea es llevar todas las fuerzas, pares y posiciones que sean necesarios para distintos cálculos a un sistema de referencia común, siendo el inercial el más indicado. Las magnitudes que actúen directamente sobre el satélite se pasarán a ejes cuerpo para introducirlo en el modelo dinámico (sección 10.2.1).

El bloque que realiza esta conversión es el de la figura 11.6. Las entradas de este bloque son:

- Input

- Las salidas del anterior (subsección 10.2.1)

- r**, Radio vector

- true.anom** Anomalía verdadera

- Vx** Velocidad según el eje x órbita

- Vy** Velocidad según el eje y órbita

- Parámetros de la órbita del satélite

- ang.orbita** Ángulos de Euler (ver página 115) de la misma ( $\Omega$ ,  $i$ ,  $\omega$ ).

- Output

**r\_iner** Vector posición en ejes inerciales

**V\_iner** Vector velocidad en ejes inerciales

El funcionamiento del bloque de la figura 11.6 es el siguiente: Primero se obtiene la matriz de rotación correspondiente a los tres giros sucesivos zxz según se detalla en la sección 9.3. Se realiza con una librería del módulo *aerospace* de Simulink, el bloque destinado a esta función aparece en la figura con el nombre *Cosine Matrix Rotation*, cuyo contenido es el de la figura 11.7. Este bloque recibe los tres ángulos de Euler, los convierte en un vector de tres senos y tres cosenos, hace las operaciones indicadas en la ecuación (9.1) y las asigna a cada una de las componentes de dicha matriz. La salida es la matriz 3x3 que en el bloque recibe el nombre de Out1.

La salida del bloque de la figura 11.7 es la inversa de la matriz que se necesita para traducir de ejes órbita a ejes inerciales. Por eso en el bloque de la figura 11.6 se calcula la matriz transpuesta. Por ser una matriz ortogonal, la matriz transpuesta es la inversa.

Por otro lado, con la anomalía verdadera se genera el vector unitario del radio vector en coordenadas órbita.

$$u_r = \begin{Bmatrix} \cos \nu \\ \sin \nu \\ 0 \end{Bmatrix}.$$

Posteriormente se multiplica por el módulo para obtener el radio vector. Con la matriz de paso calculada anteriormente se consigue el radio en coordenadas inerciales. Con la velocidad se hace lo mismo, se crea el vector

$$v = \begin{Bmatrix} v_x \\ v_y \\ 0 \end{Bmatrix}.$$

Dicho vector se traduce a coordenadas inerciales con la misma matriz.

### 11.2.3. Integración de los bloques de cálculo de órbita y paso a coordenadas inerciales

Los subsistemas descritos en las subsecciones 11.2.1 y 11.2.2 están unidos según se ve en la figura 11.8.

La situación de estos cálculos en el esquema global corresponde al rectángulo inferior de la izquierda de la figura 11.9. Nótese que el radio vector en coordenadas inerciales se almacena en un *goto* para emplearlo en cálculos posteriores.

#### 11.2.4. Comprobación de la órbita del satélite en ejes órbita

Se ha simulado una órbita para comprobar la validez del módulo. Los parámetros introducidos han sido:

- Datos de la órbita
  - Semieje mayor  $a = 16378 \text{ km}$
  - Excentricidad  $e = 0,3$
  - Longitud del nodo ascendente  $\Omega = 20^\circ$
  - Inclinação de la órbita  $i = 25^\circ$
  - Argumento del perigeo  $\omega = 50^\circ$
- Dato calculado
  - Período de la órbita  $T = \frac{2\pi}{\sqrt{\mu_T/a^3}} = 348 \text{ min}$

Se ha hecho una simulación del módulo de la órbita para un tiempo ligeramente superior al periodo y se observa que la órbita es elíptica (figura 11.10), cerrada está contenida en su plano y los valores de velocidad, energía potencial y cinética cumplen las leyes de la mecánica orbital.

Las energía cinética y potencial específicas del satélite son:

$$e_{cinética} = \frac{v^2}{2},$$

$$e_{potencial} = -\frac{\mu_T}{r},$$

$$e_{total} = e_{cinética} + e_{potencial} = cte.$$

En la figura 11.11 se comprueba que se cumplen las tres ecuaciones a lo largo de la simulación de la órbita.

## 11.3. Modelo Simulink de cálculo de órbita solar

### 11.3.1. Cálculo de la órbita del Sol

El rectángulo superior de la figura 11.9 llamado SOL corresponde a los cálculos de la órbita del Sol. Como puede verse en la figura 11.9 es muy similar al del cálculo de la órbita del satélite. En este caso, no se calcula la velocidad del Sol por no ser necesaria. Se observa que los ángulos de la órbita son los correspondientes a la órbita solar.

Al abrir el bloque *Orbita SOL*, centrado arriba, pide los parámetros de la misma (ver figura 11.13). El cálculo de la órbita del Sol en ejes órbita es el de la figura 11.14. El cálculo es idéntico al del satélite al que se le ha quitado la parte del cálculo de velocidad. (Figura 11.5).

Para pasar el radio vector del Sol a coordenadas inerciales se realizó con el bloque (abriendo el bloque *órbita sol Inerciales*) de la figura 11.15. Este bloque se programó antes del conocimiento de la existencia de la librería *Aerospace*. Por eso se hizo manualmente proyectando directamente las componentes  $x$  e  $y$  del plano órbita sobre los inerciales. El mismo bloque se hizo para el satélite, pero en éste se sustituyó por la librería, comprobando antes que daban los mismos resultados.

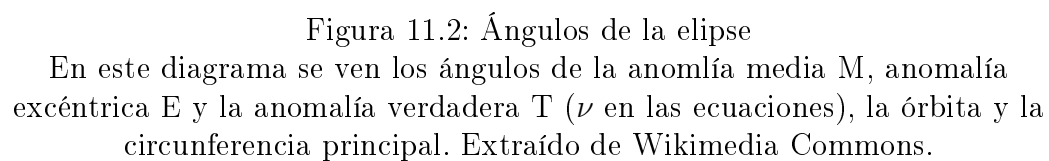
## 11.4. Cálculo de eclipse

Para determinar la condición de eclipse se ha creado el siguiente bloque en Simulink que cumple las condiciones de 11.1.3. Ver figura 11.16.

El funcionamiento del bloque es el siguiente: Primero se calcula el vector unitario que lleva la dirección del Sol. Se multiplica escalarmente por el radio vector del satélite y si es negativo se cumple la primera condición (ángulo obtuso). Por otra parte, se hace el producto vectorial del radio vector satélite por el unitario de la dirección del Sol. El resultado es un vector cuyo módulo valdrá  $r \sin B$ . El bloque *Modulo* calcula este módulo y a continuación se compara con el radio de la Tierra. Si es menor, se cumple la segunda condición. Si se cumplen ambas condiciones a la vez (*AND*) el satélite se encontrará en la zona de eclipse.

### 11.4.1. Distribución de tiempos de eclipse

Para comprobar que el subsistema de eclipse funciona, se ha hecho la simulación de la órbita sugerida en [13]. Esto es una órbita geoestacionaria que tiene el mayor tiempo de eclipse los días de equinoccio. Este eclipse dura según los cálculos del autor unos 70 minutos. Se grafica el período de eclipse y el resultado obtenido es el mostrado en la figura 11.17. Por otra parte, para la órbita geoestacionaria los eclipses desaparecen 21 días después del equinoccio. Se ha hecho la simulación de este día cambiando la anomalía excéntrica en origen de tiempo del Sol en  $E_0 = \frac{21}{365} * 2\pi$  rad. Se observa que este día hay unos 18 minutos de eclipse, mientras que al día siguiente  $E_0 = \frac{22}{365} * 2\pi$ , los minutos de eclipse son cero.





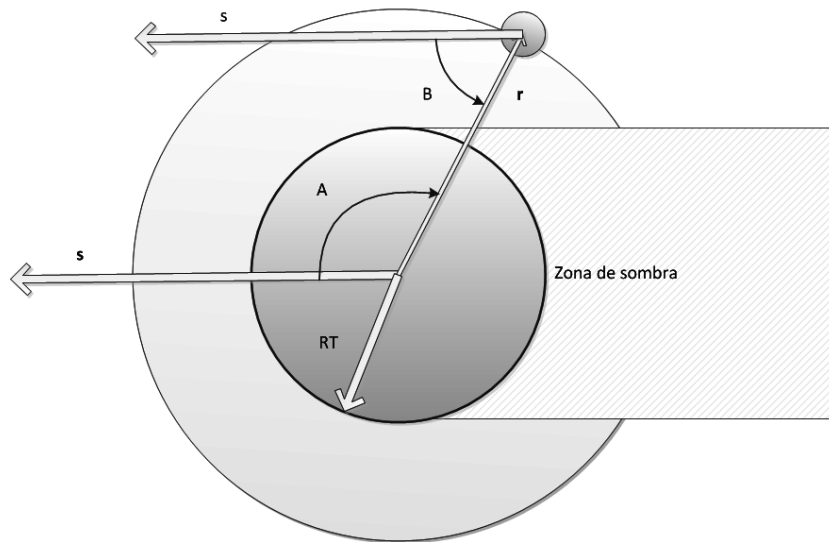


Figura 11.3: Esquema de las condiciones de eclipse  
 $r_{\odot}$  se representa como  $s$  en el dibujo

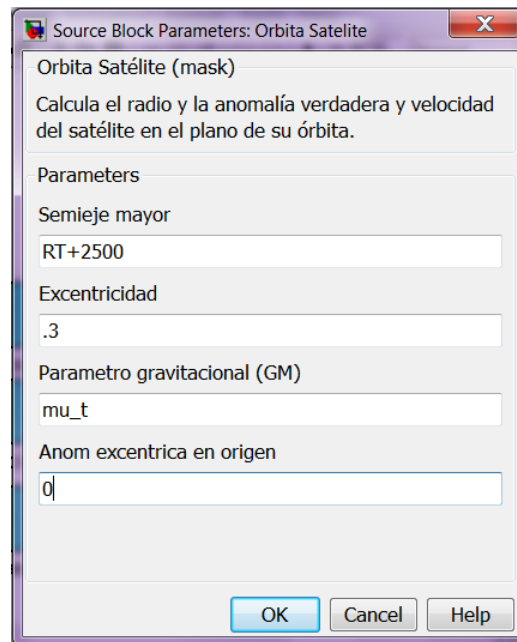
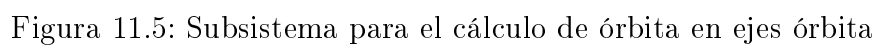


Figura 11.4: Cuadro de diálogo para introducir los parámetros de la órbita  
 $R_T=6378\text{km}$  es el radio de la Tierra. Las constantes que aparecen con letra  
 se declaran al inicializar el módulo



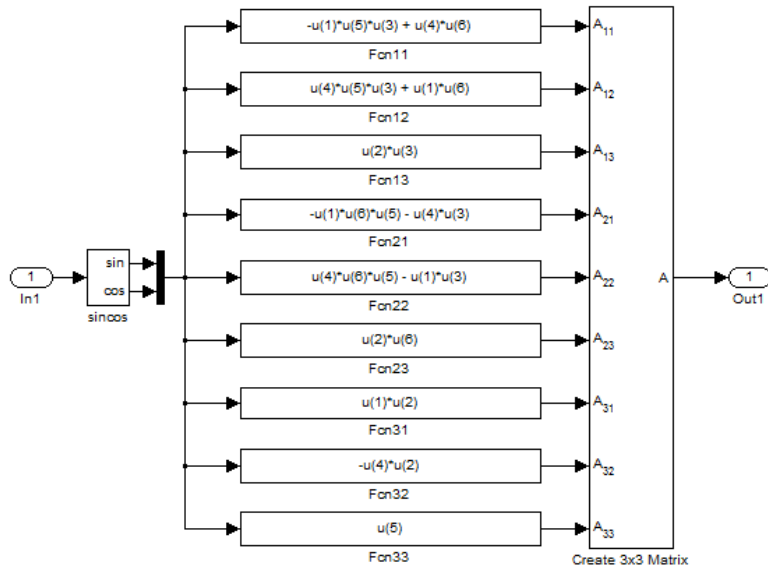


Figura 11.7: Módulo de la librería Simulink para conversión de ejes inerciales a ejes órbita

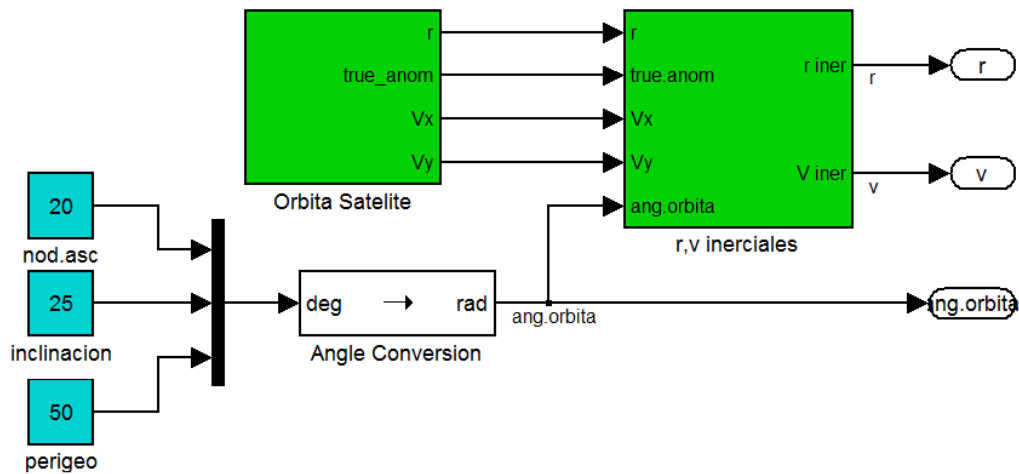


Figura 11.8: Cálculo de la órbita en ejes inerciales

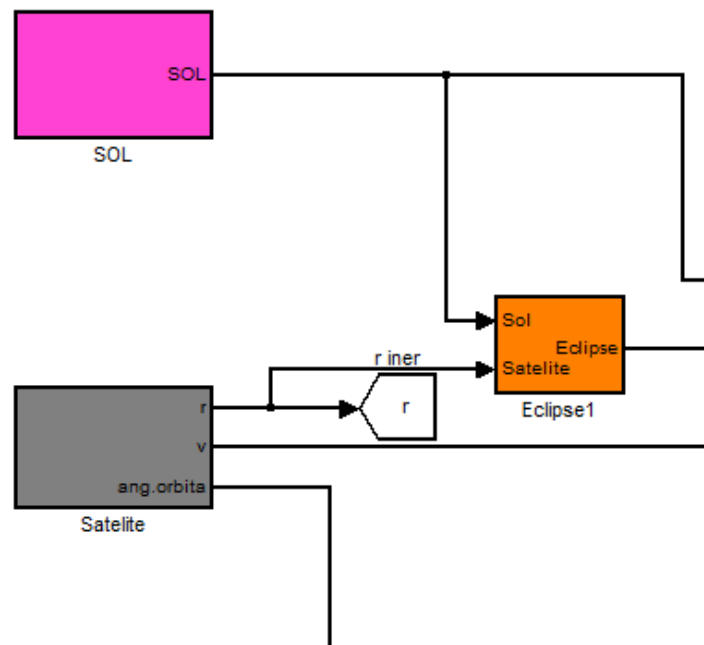


Figura 11.9: Cálculo de orbitas y eclipse del satélite y Sol

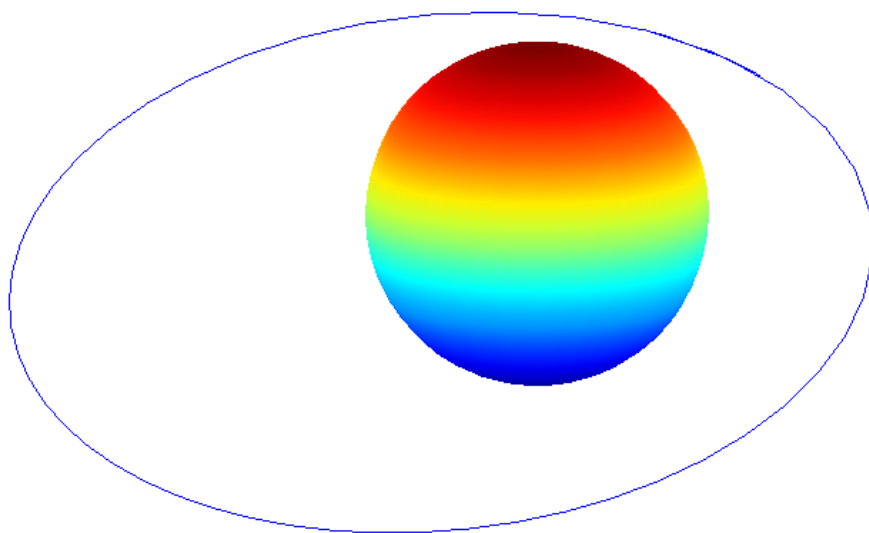


Figura 11.10: Dibujo de la órbita con la Tierra en el foco  
El punto de unión de principio y fin de órbita tiene un pequeño error debido a la discretización.

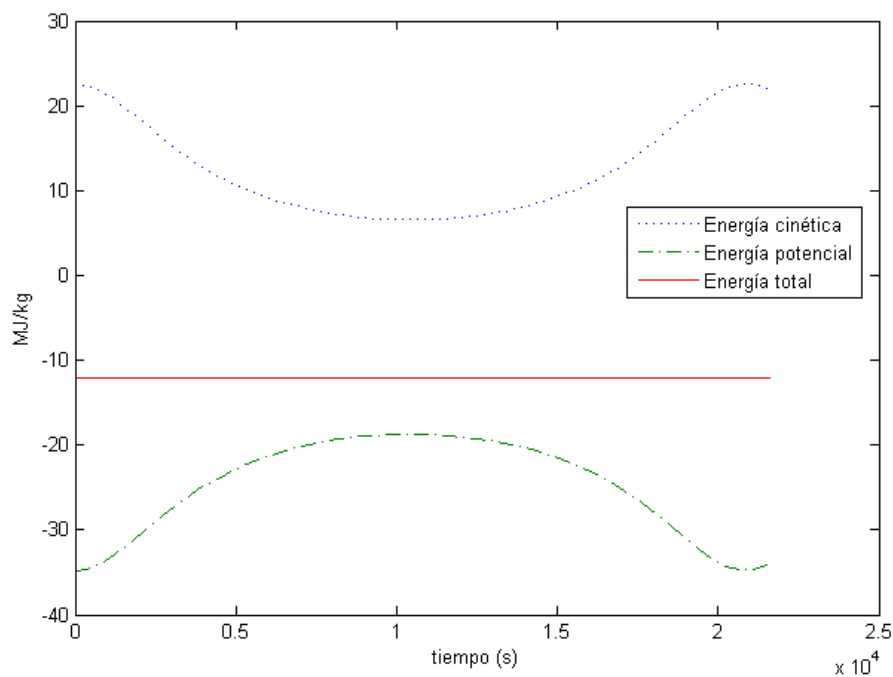


Figura 11.11: Comprobación de que la energía mecánica permanece constante

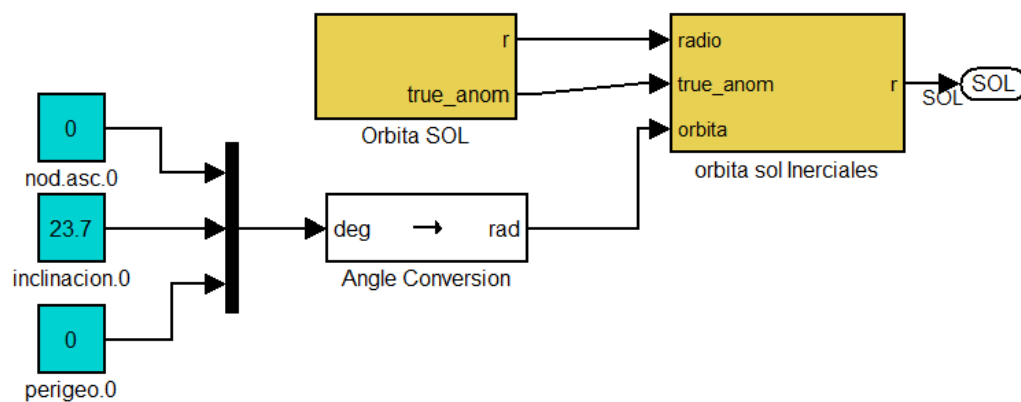


Figura 11.12: Descomposición del cálculo de la órbita del Sol

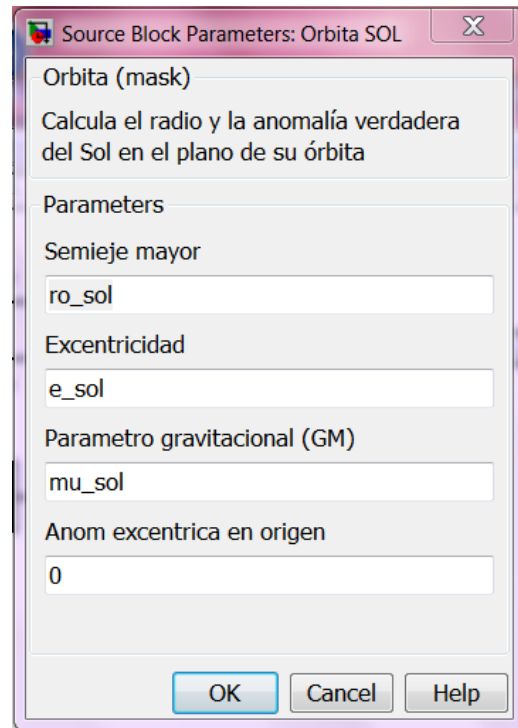


Figura 11.13: Parámetros de la órbita solar

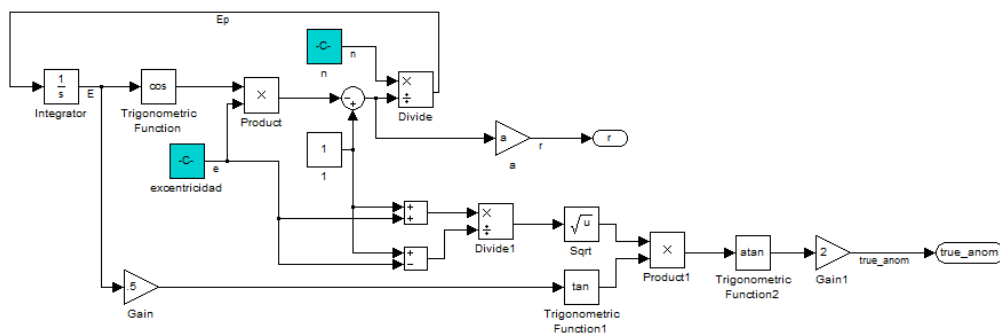


Figura 11.14: Órbita del Sol en ejes órbita

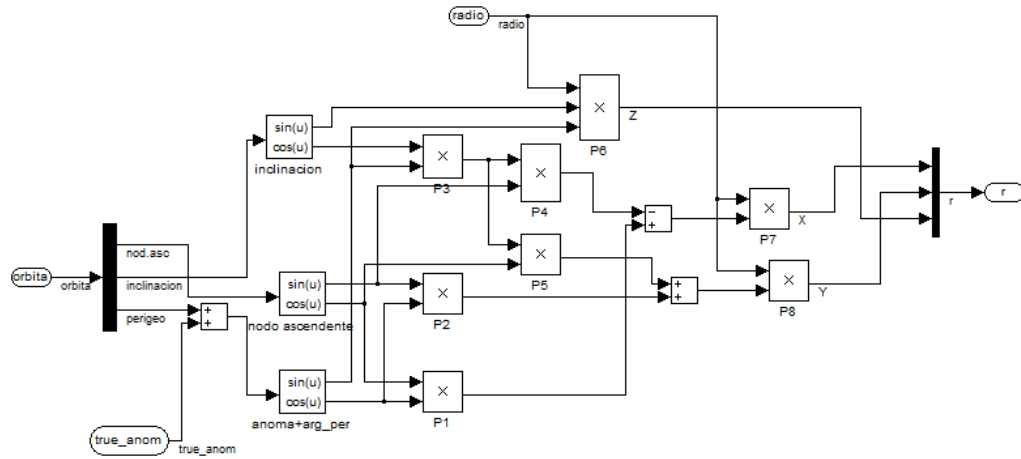


Figura 11.15: Conversión del radio vector del Sol a coordenadas inerciales

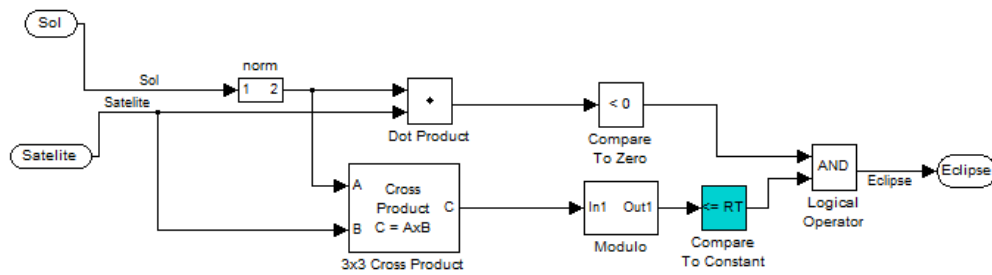


Figura 11.16: Bloque Simulink para la determinación de eclipse



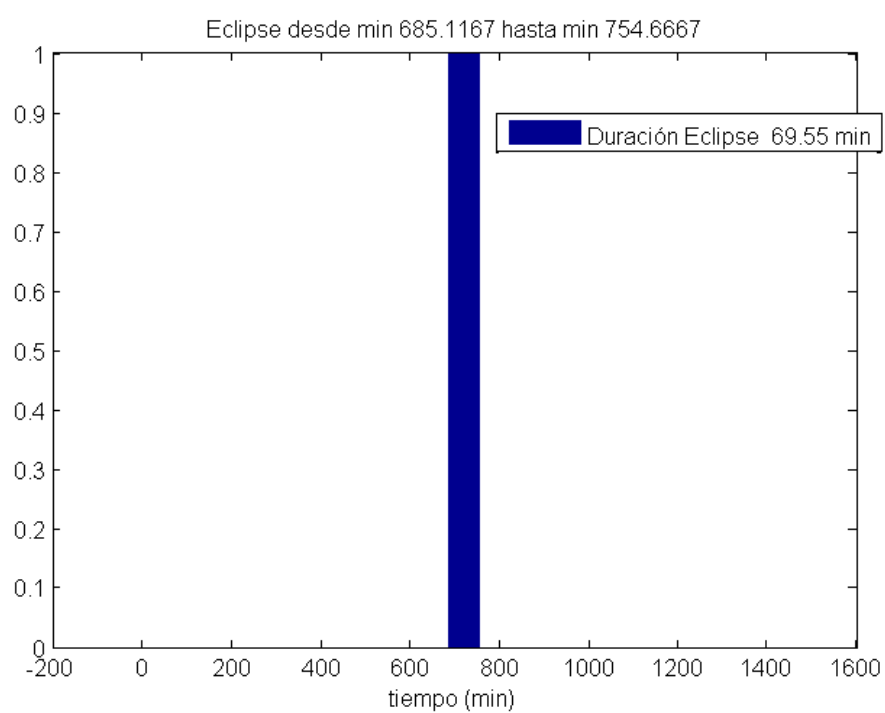


Figura 11.17: Periodo de eclipse el día 21 de Marzo (equinoccio de primavera)



# Capítulo 12

## Simulación de perturbaciones

### 12.1. Introducción

En este proyecto se va a realizar la simulación de las perturbaciones aerodinámica, solar, magnética y gradiente de gravedad. Los conceptos teóricos fueron explicados en la la parte II, sección 4.1. En esta parte, los cálculos se van a hacer con mayor precisión, ya que no se trata de un cálculo previo sino de una simulación y las fórmulas empleadas para fuerzas y pares serán tridimensionales dadas en forma vectorial. En cada tipo de perturbación se explicarán las fórmulas empleadas para la misma.

### 12.2. Perturbación aerodinámica

#### 12.2.1. Conceptos teóricos

Para calcular la fuerza aerodinámica se emplea la fórmula[13]:

$$F_a = \frac{1}{2} \rho v^2 S c_D.$$

Siendo

$\rho$  Densidad del aire (Depende de la altura)

$v$  Velocidad del satélite

$S$  Superficie del satélite proyectada en dirección normal a la velocidad

**$c_D$**  Coeficiente aerodinámico de resistencia

La dirección de esta fuerza es opuesta a la velocidad y está aplicada en el centro de presiones aerodinámico del satélite.

El momento de la perturbación solar es igual a:

$$\mathbf{T}_a = \mathbf{c}_{pa} \times F_a(-\mathbf{u}_v).$$

Siendo

**$c_{pa}$**  Centro de presiones aerodinámico (se supone constante dato de la geometría)

**$\mathbf{u}_v$**  Vector unitario de la velocidad

### 12.2.2. Modelo Simulink

Para calcular el par aerodinámico se ha desarrollado el modelo simulink de la figura 12.1.

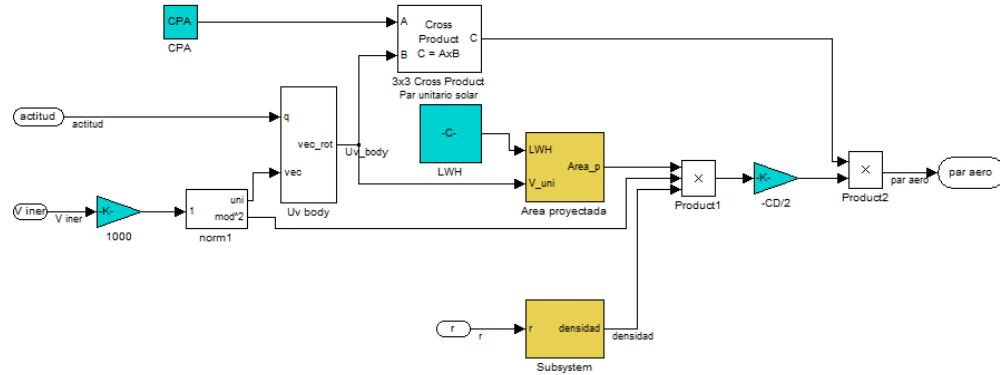


Figura 12.1: Subsistema para el cálculo del par de perturbación aerodinámico

#### ■ Input

- Variables en función del tiempo

**Actitud** Cuaternión de actitud (corresponde a  $q$  en otros módulos)

**V\_iner** Velocidad del satélite en ejes inerciales

**r** Vector de posición del satélite

- Constantes

**CPA** Centro de presiones aerodinámico

**LWH** Dimensiones del satélite (Largo, ancho, alto)

**CD** Coeficiente aerodinámico de resistencia

Tabla de densidades en función de la altura (tomada del modelo Simulink de UPM-SAT-1)

- Output

**par\_aero** Par de perturbación aerodinámico en ejes cuerpo

El funcionamiento de este subsistema es el siguiente: la velocidad del satélite se multiplica por 1000 para pasarlo al sistema internacional. A continuación se calcula el vector unitario y el módulo al cuadrado. Es similar a la figura 12.12, pero sin el multiplicador. El vector unitario lo pasamos a ejes cuerpo con el cuaternión de actitud. El módulo  $Uv\_body$  realiza la operación según la ecuación (9.4) y se ha tomado de la librería *Aerospace*. La salida de este módulo es el vector unitario de la velocidad en ejes cuerpo. El producto vectorial de la parte superior calcula  $\mathbf{c}_{pa} \times \mathbf{u}_v$ . Por lo tanto tan sólo falta calcular  $F_a$ . Se consigue multiplicando el cuadrado de la velocidad por el área proyectada, por la densidad del aire y por  $-\frac{c_D}{2}$ .

### 12.2.3. Módulo área proyectada

El submódulo *Area\_proyectada* es el de la figura (12.2). La primera entrada LWH son las tres dimensiones del satélite, que se supone prismático. Con los productos *Cara XY*, *Cara YZ*, y *Cara XZ* se calculan las áreas de estas tres caras que coinciden en un vértice. Cada cara se convierte en su vector característico mediante la concatenación de vectores. Por ejemplo, la Cara XY es:

$$\mathbf{CaraXY} = \begin{Bmatrix} 0 \\ 0 \\ L * W \end{Bmatrix}.$$

Se multiplica escalarmente por el unitario de la velocidad y esa es el área proyectada de esa cara. Finalmente se saca el valor absoluto por si hemos escogido la cara opuesta a la real. El proceso es análogo con las otras dos caras y el área proyectada total es la suma de las tres. Se han tomado tres caras porque es imposible que el viento le de a más.

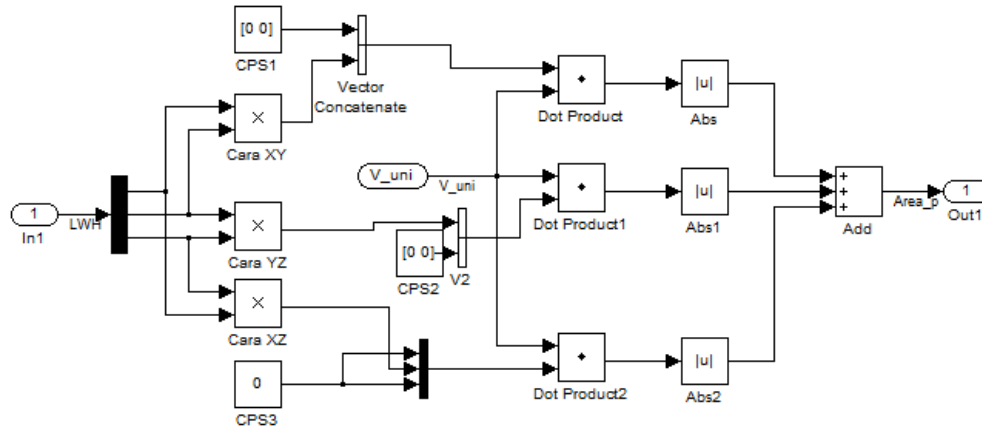


Figura 12.2: Subsistema que calcula el área proyectada

#### 12.2.4. Densidad del aire

Para calcular la densidad del aire se ha tomado la tabla incluida en la simulación del UPM-SAT-1 proporcionada por el director del proyecto. El subsistema desarrollado es el de la figura 12.3. La tabla de valores se encuentra en el cuadro 12.1. Se toma el radio vector del satélite y se halla su módulo (elevando sus componentes al cuadrado, sumando y extrayendo raíz cuadrada), se le resta el radio de la Tierra para obtener la altura. Se multiplica por 1000 para pasarlo a metros y se busca el valor interpolado en la tabla de densidades.

#### 12.2.5. Simulación

En la figura 12.4 se ha graficado el valor del módulo obtenido en la simulación de la órbita especificada en el apartado 11.2.4. Como el satélite está rotando, el módulo de la perturbación varía a lo largo de cada revolución. Por

eso la gráfica parece una banda. Se aprecia que hay dos máximos de módulo que corresponderán al momento en que la velocidad es perpendicular a las superficies más grandes del prisma. La razón de que existan ceros puntuales puede ser porque la velocidad está alineada con la posición del centro de presiones.

## 12.3. Perturbación solar

### 12.3.1. Conceptos teóricos

Para calcular la fuerza de la presión solar se emplea la fórmula[16]:

$$F_{sun} = P_{sun}S(1 + q).$$

Siendo

$P_{sun}$  Presión solar (Pa). Se obtiene como  $F_s/c = \frac{1358\text{W/m}^2}{3e8\text{m/s}}$ .

$q$  Factor de reflexión solar

$S$  Superficie del satélite proyectada en dirección normal a la radiación solar

La dirección de esta fuerza lleva la dirección de la radiación solar y está aplicada en el centro de presión solar del satélite.

El momento de la perturbación solar es igual a:

$$\mathbf{T}_S = \mathbf{c}_{ps} \times F_{sun}(-\mathbf{u}_{\odot}). \quad (12.1)$$

Siendo

$\mathbf{c}_{ps}$  Centro de presiones solar (se supone constante dato de la geometría)

$\mathbf{u}_{\odot}$  Vector unitario de la posición del Sol.

### 12.3.2. Modelo Simulink

El modelo se describe en la figura 12.5.

■ Input

● Variables en función del tiempo

**Actitud** Cuaternión de actitud (corresponde a  $q$  en otros módulos)

**Sol\_iner** Posición del Sol en ejes inerciales

**Eclipse** Variable lógica que determina si hay eclipse

- Constantes

**CPS** Centro de presiones solar

**LWH** Dimensiones del satélite (Largo, ancho, alto)

**qs** Coeficiente de reflexión solar de la superficie del satélite

**PS** Presión solar

- Output

**Par\_solar** Par de perturbación solar en ejes cuerpo

El funcionamiento de este subsistema es el siguiente: primero el vector de posición del Sol en ejes inerciales se convierte en unitario y se convierte a ejes cuerpo gracias al cuaternio de actitud y a la librería *Aerospace* empleada en el par aerodinámico (subsección 12.2). Se multiplica vectorialmente este vector unitario en ejes cuerpo por el vector de posición del centro de presiones solar (*CPS*). Esto equivale a calcular  $c_{ps} \times (-u_{\odot})$ , la parte vectorial de la ecuación (12.1). Por otra parte, se calcula el área proyectada (ver figura 12.2) del satélite según la dirección de la radiación solar, del mismo modo que se hizo en el apartado 12.2.2. Por último, se multiplica el área proyectada por la presión solar y por  $(1+qs)$  para tener el par solar total.

Ahora bien, como este par existirá realmente si el satélite no se encuentra en zona de eclipse. Por eso se ha colocado un *switch* al final que da salida nula en caso de eclipse y el par calculado en caso de no eclipse. El cálculo de eclipse se explica en el apartado 11.4.

### 12.3.3. Simulación

En la figura 12.6 se simula el par solar de la misma órbita que el apardo anterior. Igualmente el módulo oscila en banda pero realmente son oscilaciones en tiempos pequeños a la escala de la gráfica. El hueco que hay alrededor de los 5000 segundos corresponde al período de eclipse. Permanece constante a lo largo de la trayectoria porque la dirección del Sol es constante y la de la consigna también.



## 12.4. Perturbación magnética

### 12.4.1. Conceptos teóricos

Para calcular el campo magnético terrestre se emplea la fórmula del campo magnético del dipolo ideal[1]:

$$\mathbf{B} = \frac{\mu_0}{4 * \pi} * \frac{3 (\mathbf{m} \cdot \mathbf{r}) \mathbf{r} - r^2 \mathbf{m}}{r^5}. \quad (12.2)$$

Siendo

$\mu_0$  Permeabilidad magnética en el vacío  $\mu_0 = 4\pi e - 7^N/A^2$

$\mathbf{m}$  Momento dipolar de la Tierra ( $m = 7.96e15 \text{ Am}^2$ )

$\mathbf{r}$  Radio vector del satélite

El vector  $\mathbf{m}$  apunta aproximadamente de sur a norte (eje z inercial en el sistema geocéntrico). Sin embargo, la desviación del norte magnético y el norte geográfico es de unos  $11^\circ$ . En los cálculos se ha tenido esto en cuenta, además de que este vector gira solidario con la Tierra. Aunque el viento solar distorsiona la magnetosfera, su modelización es muy complicada y no se ha tenido en cuenta en este proyecto.

El campo magnético no crearía pares si el satélite no tuviera un dipolo residual generado por las corrientes de aparatos eléctricos de a bordo. Este dipolo residual se considera un dato del problema  $\mathbf{D}$ .

El momento de la perturbación magnética es igual a:

$$\mathbf{T}_m = \mathbf{D} \times \mathbf{B}.$$

### 12.4.2. Modelo Simulink

El modelo se describe en la figura 12.8.

#### ■ Input

- Variables en función del tiempo

$\mathbf{r}$  Vector de posición del satélite en ejes inerciales

**Actitud** Cuaternión de actitud del satélite (q en otros módulos)

- Constantes

**wt** Velocidad angular de la Tierra

**m0** Momento dipolar de la Tierra

**B0**  $\mu_0/(4\pi)$

- Output

**Par\_magnetico** Par de perturbación magnético en ejes cuerpo

En el modelo de la figura 12.8 se convierte el vector campo magnético en ejes inerciales a ejes cuerpo gracias al cuaternión de actitud y la librería *Aerospace*. Se multiplica vectorialmente por **D** y se obtiene el par magnético en ejes cuerpo.

### 12.4.3. Cálculo del campo magnético terrestre

Para hacer el cálculo del campo magnético terrestre se ha desarrollado el módulo de la figura 12.9.

Se calcula el ángulo girado por la Tierra ( $\varphi = wt \cdot \text{tiempo}$ ) y se construye el cuaternión correspondiente a ese giro:

$$q_T = \begin{Bmatrix} \cos \frac{\varphi}{2} \\ 0 \\ 0 \\ \sin \frac{\varphi}{2} \end{Bmatrix}.$$

Se gira el vector  $m0$  (momento dipolar de la Tierra) con este cuaternión y la librería correspondiente, obteniéndose así el momento dipolar de la Tierra (**m**) en cualquier instante de tiempo.

Con esto se procede a hacer los cálculos de la ecuación (12.2).

En la parte inferior del diagrama se hace el cálculo de  $r^5$  y  $r^2$ . El vector **r** se asigna a un *goto* para emplearlo múltiples veces.

Se calcula el numerador de la ecuación multiplicando escalarmente **m** y **r**. El resultado se multiplica por 3 y el vector **r**. Siendo el minuendo del restador. El sustraendo se consigue multiplicando el vector **m** por el escalar  $r^2$ . La diferencia es el numerador, se multiplica por la constante **B0** y se divide por  $r^5$ . El resultado es el campo magnético terrestre.

#### 12.4.4. Simulación

En la figura 12.10 se ha graficado el par magnético con las condiciones de la órbita de los apartados anteriores. Como el satélite está girando, el módulo del par cambia en cada revolución y el promedio disminuye cuando se acerca al ecuador porque allí es menor el campo magnético.

### 12.5. Perturbación por gradiente de gravedad

#### 12.5.1. Conceptos teóricos

La ecuación vectorial que da el par de gradiente de gravedad es[13]:

$$\mathbf{T}_g = -3\frac{\mu}{r^3}\mathbf{u}_r \times (I\mathbf{u}_r). \quad (12.3)$$

Siendo

$\mu$  Parámetro gravitacional de la Tierra ( $\mu = GM_T$ )

$r$  Módulo radio vector del satélite

$\mathbf{u}_r$  Vector unitario del radio vector terrestre en ejes cuerpo

#### 12.5.2. Modelo Simulink para el control de actitud

El subsistema que calcula la ecuación (12.3) es el que aparece en la figura (12.11).

##### ■ Input

- Variables en función del tiempo

$\mathbf{r}$  Vector de posición del satélite en ejes inerciales

**Actitud** Cuaternión de actitud del satélite (q en otros módulos)

- Constantes

**mu\_t** Parámetro gravitacional terrestre ( $\mu_T$ )

**Inercia** Tensor de inercia del satélite respecto a ejes cuerpo

##### ■ Output

**Par\_grad\_grav** Par de perturbación de gradiente de gravedad en ejes cuerpo

Dado el vector de posición en ejes inerciales se calcula su vector unitario y su módulo al cubo. El vector unitario se traduce a ejes cuerpo con el cuaternión de actitud y la librería *Aerospace* de Simulink. Con eso ya se tiene el vector unitario  $\mathbf{u}_r$ . Se hace el producto del tensor de inercia por este vector y al resultado se hace el producto vectorial por  $\mathbf{u}_r$ . Se multiplica por las constantes  $-3$  y  $\mu_T$  y se divide por  $r^3$ .

#### 12.5.2.1. Cálculo del vector unitario y el módulo al cubo

El subsistema para calcular un vector unitario se muestra en la figura 12.12.

En este subsistema lo primero que se hace es elevar las componentes del vector al cuadrado para después sumarlas, con lo que se obtiene el módulo del vector al cuadrado. A continuación se extrae la raíz cuadrada, con lo que se obtiene el módulo. El módulo al cubo es el producto de los dos últimos resultados, y el vector unitario es el cociente del vector original por su propio módulo.

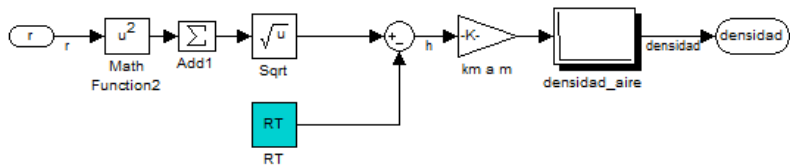
#### 12.5.3. Simulación

En la figura 12.13 se ha graficado la simulación con las condiciones de órbita de los apartados anteriores.

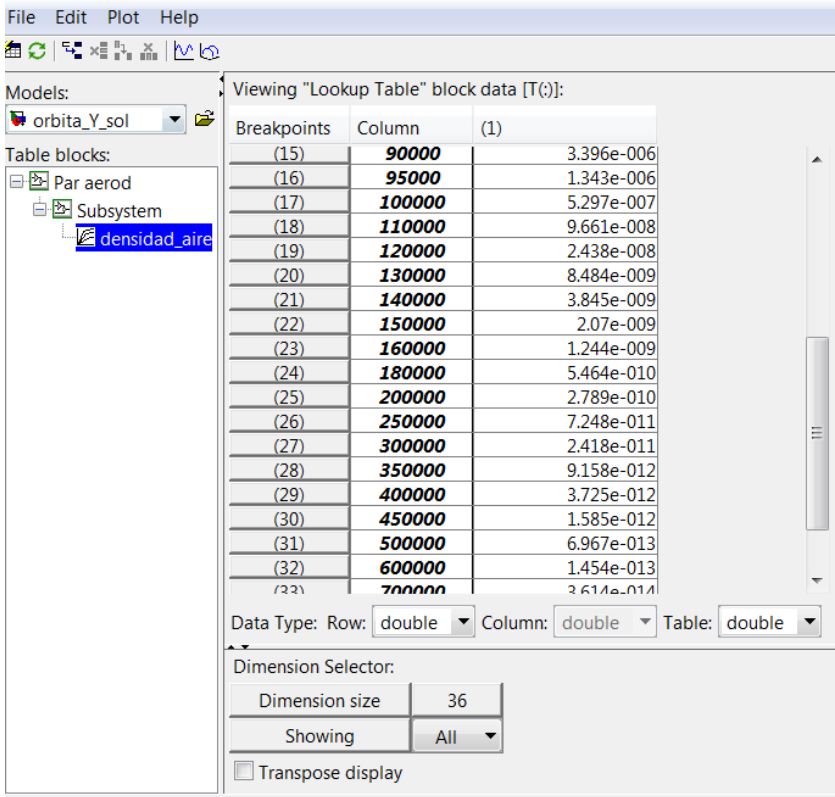
### 12.6. Integración de los pares de perturbación al modelo global

Los pares calculados por independiente se suman y se aplican al modelo dinámico del sistema (subsección 10.2.4). Se observa que todos los pares comparten la entrada del radio vector en ejes inerciales ( $\mathbf{r}$ ) y el cuaternión de actitud ( $\mathbf{q}$ ). El par aerodinámico precisa de la velocidad y el par solar el radio vector del Sol y la condición lógica de eclipse. Ver figura 12.14.

12.6. INTEGRACIÓN DE LOS PARES DE PERTURBACIÓN AL MODELO GLOBAL157



(a) Módulo para el cálculo de la densidad del aire en función de la altura



(b) Tabla de valores para la densidad en función de la altura

Figura 12.3: Cálculo de la densidad del aire en función de la altura

Altitud	Densidad
0	1.225
25000	0.0390
30000	0.0177
35000	0.0083
40000	0.0040
45000	0.0020
50000	0.0011
55000	5.8210e-4
60000	3.2060e-4
65000	1.7180e-4
70000	8.7700e-5
75000	4.1780e-5
80000	1.9050e-5
85000	8.3370e-6
90000	3.3960e-6
95000	1.3430e-6
100000	5.2970e-7
110000	9.6610e-8
120000	2.4380e-8
130000	8.4840e-9
140000	3.8450e-9
150000	2.0700e-9
160000	1.2440e-9
180000	5.4640e-10
200000	2.7890e-10
250000	7.2480e-11
300000	2.4180e-11
350000	9.1580e-12
400000	3.7250e-12
450000	1.5850e-12
500000	6.9670e-13
600000	1.4540e-13
700000	3.6140e-14
800000	1.1700e-14
900000	5.2450e-15
1000000	3.0190e-15

Cuadro 12.1: Tabla de valores de densidad del aire

## 12.6. INTEGRACIÓN DE LOS PARES DE PERTURBACIÓN AL MODELO GLOBAL159

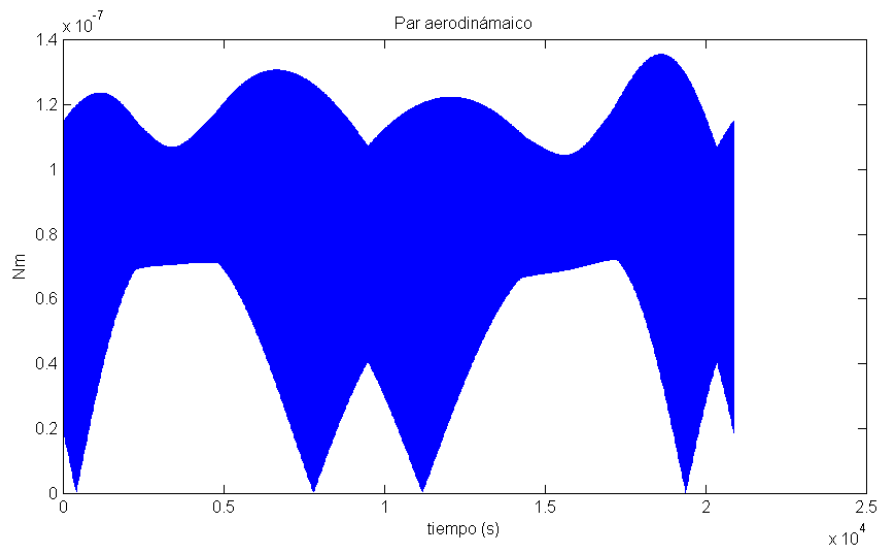


Figura 12.4: Perturbación aerodinámica en módulo

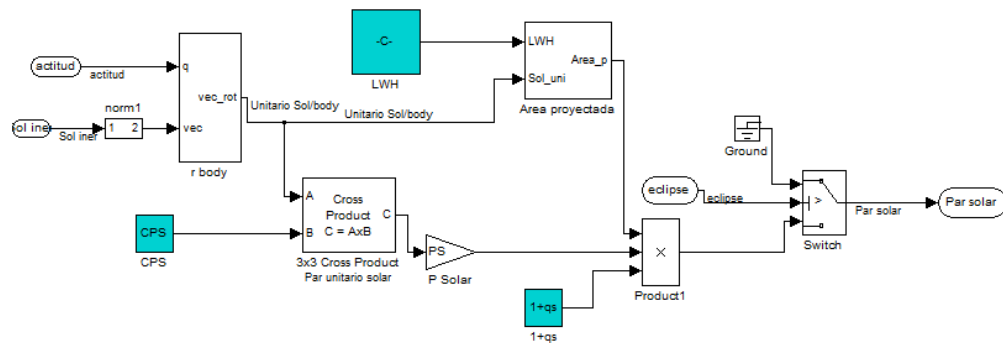


Figura 12.5: Subsistema para el cálculo del par Solar

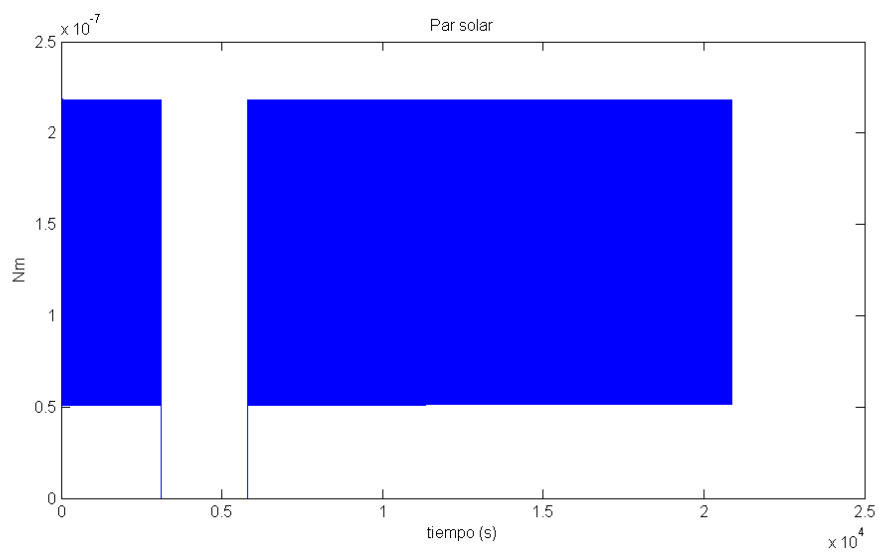


Figura 12.6: Perturbación Solar en módulo

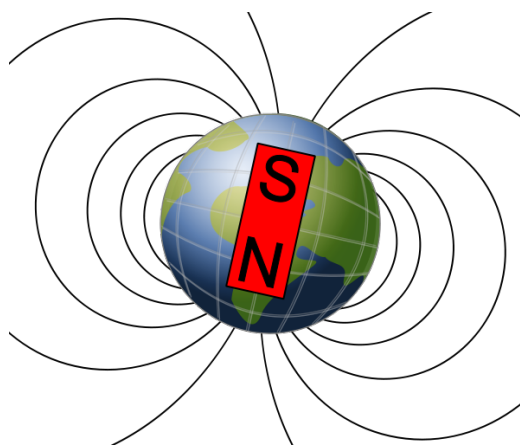


Figura 12.7: Representación del dipolo terrestre





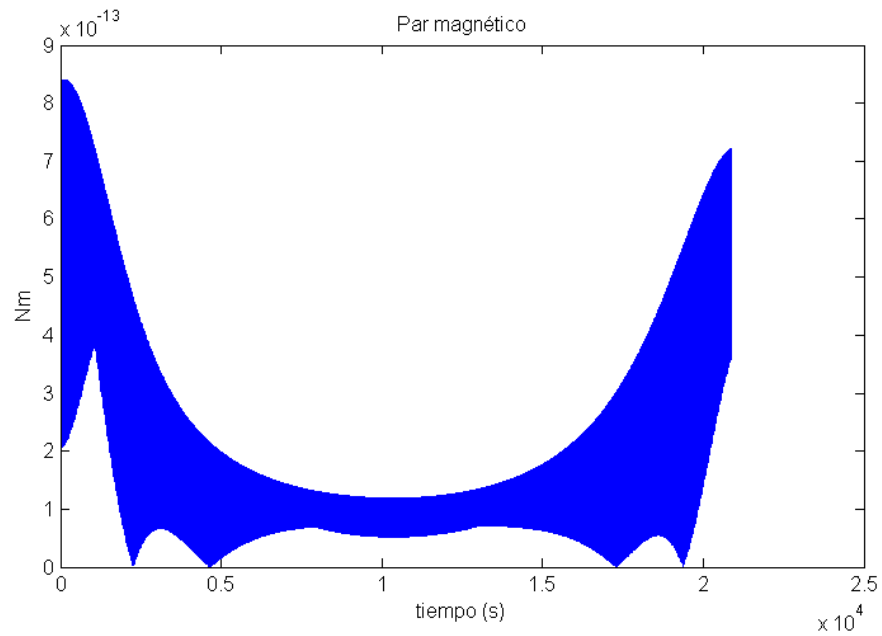


Figura 12.10: Perturbación debida al campo magnético terrestre en módulo

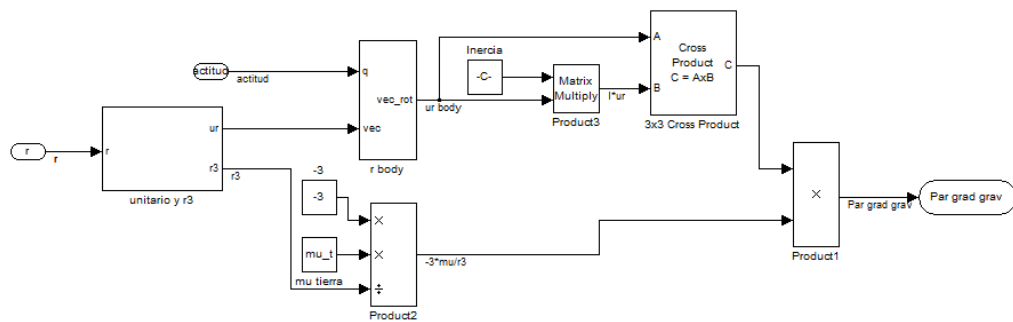


Figura 12.11: Subsistema que calcula el par de gradiente de gravedad

## 12.6. INTEGRACIÓN DE LOS PARES DE PERTURBACIÓN AL MODELO GLOBAL163

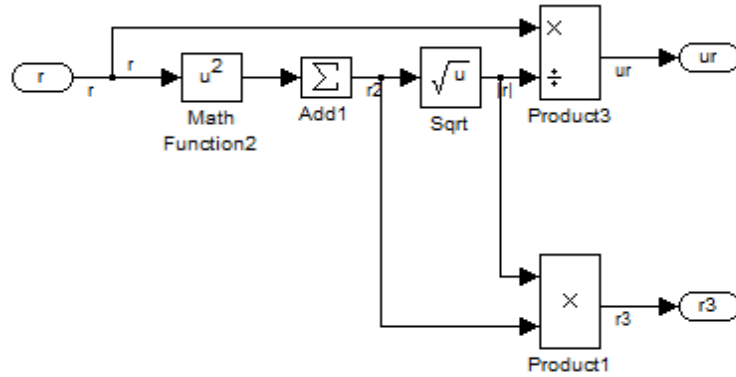


Figura 12.12: Cálculo del vector unitario y el módulo al cubo

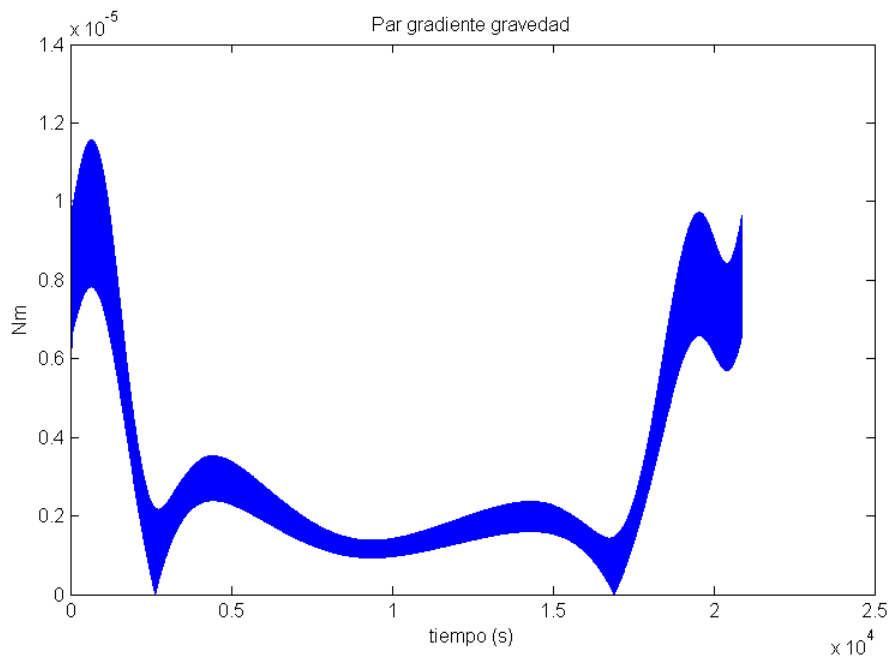


Figura 12.13: Perturbación debida al gradiente de gravedad en módulo

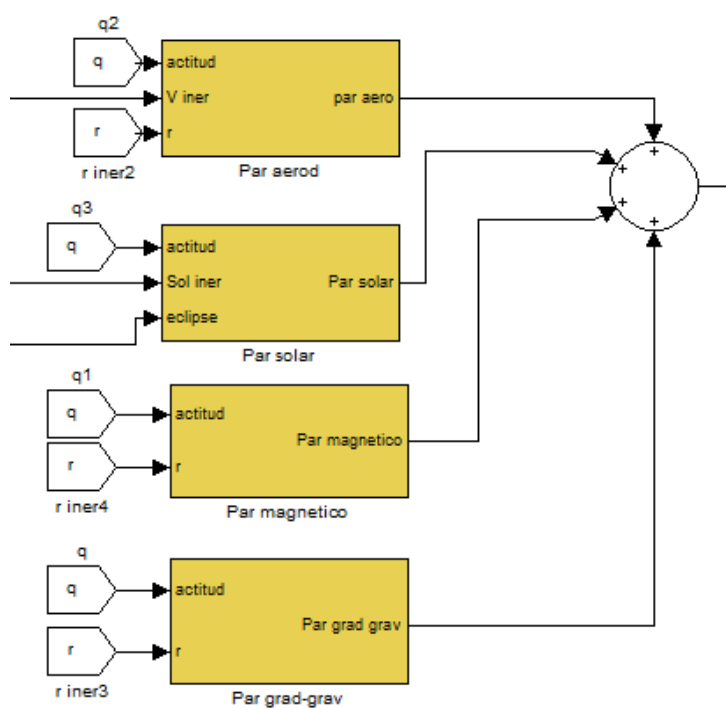


Figura 12.14: Integración de los cuatro pares de perturbación al modelo global

# Capítulo 13

## Control de actitud

### 13.1. Introducción

Hasta el momento, se ha explicado el cálculo de las órbitas del satélite y del Sol respecto al sistema de referencia inercial geocéntrico (capítulo 11). Con ello se ha calculado los pares de perturbación en el capítulo 12. Por otra parte, se ha calculado la respuesta dinámica del satélite y el cuaternión que da la actitud del mismo (capítulo 10). El conjunto que comprende este sistema se muestra en la figura 13.1. Con el cuaternión de actitud se consigue hacer la conversión de todas la magnitudes de ejes inerciales a ejes cuerpo. El objeto de este capítulo (y al fin y al cabo de esta parte del proyecto) es conseguir que el satélite apunte al objetivo que se le indique. Para ello se ha hecho un pequeño módulo de control que se ha incorporado al sistema y se ha experimentado simulando su funcionamiento.

### 13.2. Subsistema de control

#### 13.2.1. Estrategia seguida

El control de actitud que se ha diseñado para este satélite es tal que se le pide que el eje  $z$  del mismo (ejes cuerpo) apunte a una dirección seleccionada. Esta dirección la puede elegir la persona que ejecute la simulación (por supuesto la elige en ejes inerciales).

La estrategia que se ha seguido es comparar la dirección del eje  $z$  del satélite y la dirección de la consigna traducida a ejes cuerpo. El objetivo es

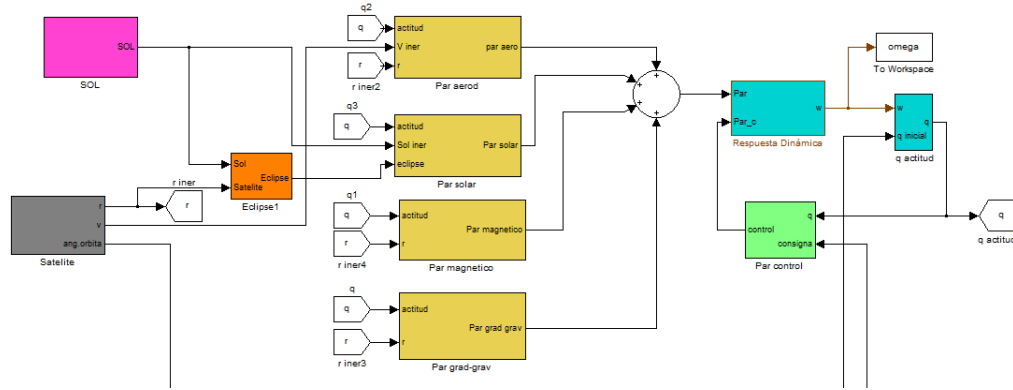


Figura 13.1: Integración de los subsistemas de cálculo de pares, modelo dinámico, cálculo de órbitas y control

que coincidan. Para ello se trabaja con vectores unitarios. Se calcula el producto vectorial del eje unitario  $z$  y el vector unitario consigna y el resultado es proporcional al seno del ángulo que se quiere reducir a cero y su dirección la que debería tener la velocidad angular eliminando la componente  $z$  para acercarse a ambos vectores.

$$\mathbf{u}_z \times \mathbf{u}_{obj} = \sin \varepsilon \mathbf{u}_{\omega xy}.$$

Siendo

$\mathbf{u}_z$  Unitario en la dirección  $z$  (ejes cuerpo)

$\mathbf{u}_{obj}$  Unitario en la dirección de la consigna en ejes cuerpo (a elegir por el usuario)

$\varepsilon$  Ángulo de error entre la consigna y el vector  $z$

$\mathbf{u}_{\omega xy}$  Vector unitario que indica la dirección de la  $\omega$  que corregiría el error

Para conseguir reducir  $\varepsilon$  se aplica un par de control que depende :

1. En magnitud, de  $\varepsilon$  .
2. En dirección, del producto  $I \mathbf{u}_{\omega xy}$ . Si el satélite tiene un momento de inercia  $I_x > I_y$ , los pares a aplicar para conseguir aceleraciones (o deceleraciones) correctoras tendrán que ser mayores en el eje  $x$  que en el eje  $y$ .

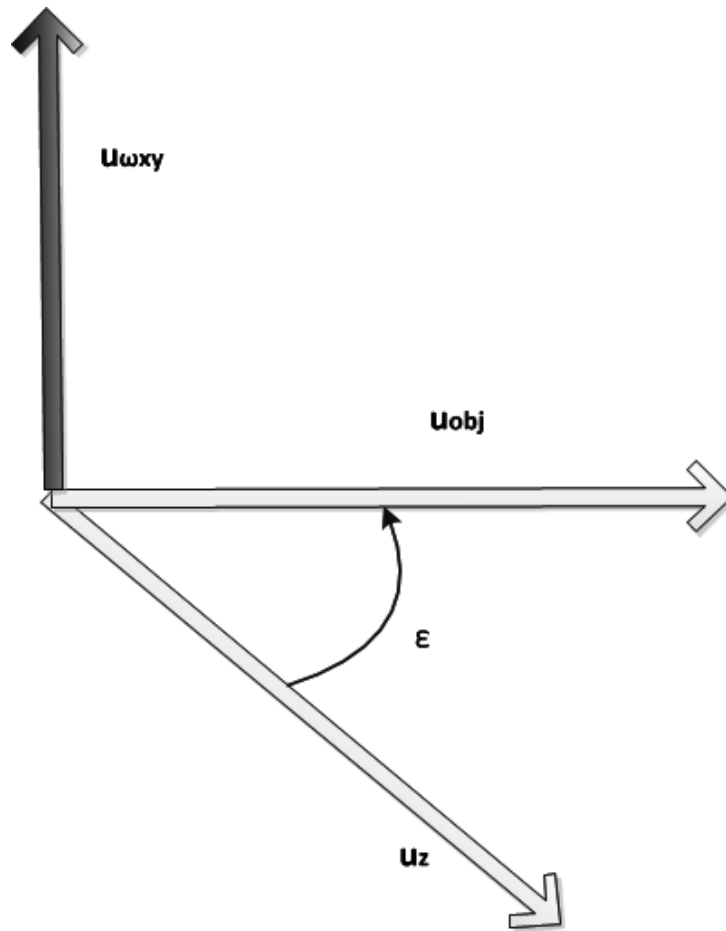


Figura 13.2: Determinación del error

Para aplicar el par corrector se ha instalado un control proporcional integral derivativo (PID). Las constantes de proporcionalidad  $k_P$ , integral  $k_I$  y derivativa  $k_D$  las elige el usuario. Son parámetros del bloque de control (figura 13.3 ).

### 13.2.2. Conceptos teóricos

El concepto de control es el aproximar el valor de una magnitud a una consigna dada y mantener dicho valor dentro de unos límites a lo largo del tiempo. Los métodos de control más empleados son:

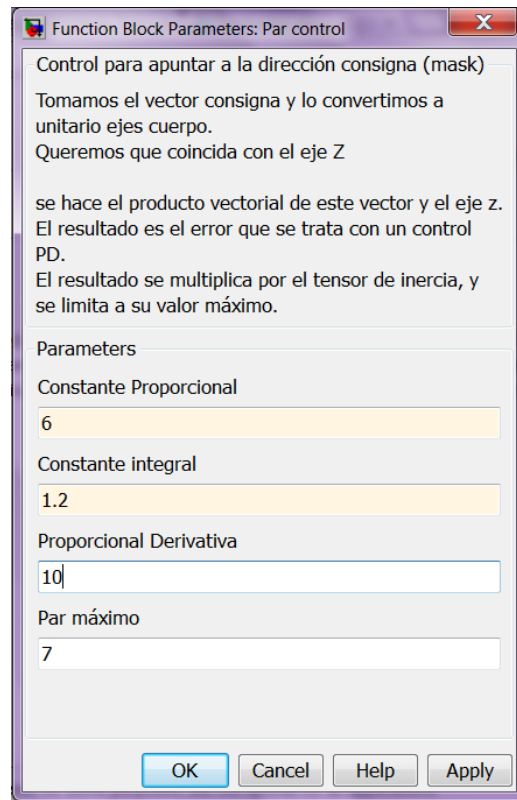


Figura 13.3: Parámetros del bloque de control

**On/Off** El dispositivo encargado de corregir la magnitud tiene dos estados. Por ejemplo el termostato de una calefacción enciende la caldera si la temperatura de la habitación está por debajo de la consigna y la apaga si está por encima. Este control no consigue valores de la consigna con mucha precisión.

**Proporcional** La señal de salida del controlador es proporcional al error o diferencia del valor de la consigna y el estado actual. Por ejemplo una caldera puede aumentar la potencia de combustión si detecta que el agua de retorno de los radiadores está mucho más fría que la que impulsa.

**Integral** Un control proporcional puede causar oscilaciones alrededor del punto de consigna y que a tiempos largos el valor de la variable a controlar no coincide con el de la consigna, manteniéndose un error per-



manente (steady state). El control integral consigue eliminar este error de estado estacionario porque integra este error a lo largo del tiempo añadiéndose la corrección correspondiente al control proporcional.

**Derivativo** El control no solo contempla el valor del error, sino la variación de éste con el tiempo. De modo que si la aproximación a la consigna es demasiado rápida corrige la acción del control proporcional para que éste no sobrepase la consigna, lo que se denomina overshoot.

**Control Combinado** Existen los controles PD, PI y PID, que son combinaciones de los tres últimos.

Cada control se caracteriza por la constante de proporcionalidad que aplica. Se denominan  $k_P$ ,  $k_I$  y  $k_D$  según se trate de control proporcional, integral o derivativo. Los efectos de la variación de las constantes de los controles PID sobre la respuesta del sistema se resumen en la tabla 13.1.

Incrementar parámetro	$k_P$	$k_I$	$k_D$
Tiempo de subida	Decrece	Decrece	Cambio pequeño
Overshoot	Crece	Crece	Decrece
Tiempo de estabilización	Cambio pequeño	Crece	Decrece
Error estacionario	Decrece	Elimina	En teoría no tiene efecto
Estabilidad	Degrada	Degrada	Mejora si $k_D$ pequeño

Cuadro 13.1: Efectos al incrementar cada parámetro por separado [8]

Para sintonizar un control PID la literatura encontrada recomienda usar el método Ziegler-Nichols [17]. Es un método que suele funcionar para múltiples casos (concretamente para sistemas con mucha inercia y amortiguados) y es sencillo de implementar. Consiste en seguir los siguientes pasos:

1. Establecer las ganancias  $k_I$  y  $k_D$  a 0.
2. Incrementar el valor de  $k_P$  hasta el sistema oscile con amplitud constante. El valor de  $k_P$  que consigue esto se denomina ganancia última  $k_U$  (ultimate gain) y el período de esta oscilación  $T_U$ .
3. Según el control que se vaya a usar cargar las tres variables  $k_P$ ,  $k_I$  y  $k_D$  con los valores de la tabla 13.2.

Cuando este método no da buenos resultados se debe sintonizar de forma manual atendiendo a los criterios de la tabla 13.1.

Método Ziegler-Nichols			
Tipo de control	$k_P$	$k_I$	$k_D$
P	$0.5 k_U$	-	-
PI	$0.45 k_U$	$1.2 k_P/T_U$	-
PD	$0.8 k_U$	-	$K_P T_U/8$
PID clásico	$0.60 k_U$	$2 k_P/T_U$	$K_P T_U/8$
Regla de integral de Pessel	$0.7 k_U$	$0.4 k_P/T_U$	$0,15 K_P T_U$
Ligero overshoot	$0.33 k_U$	$2 k_P/T_U$	$K_P T_U/3$
Sin overshoot	$0.2 k_U$	$2 k_P/T_U$	$K_P T_U/3$

Cuadro 13.2: Tabla de valores para sintonizar un PID según el método Ziegler-Nichols

### 13.2.3. Modelo Simulink

El subsistema para determinar el control de actitud del satélite se muestra en la figura 13.4.

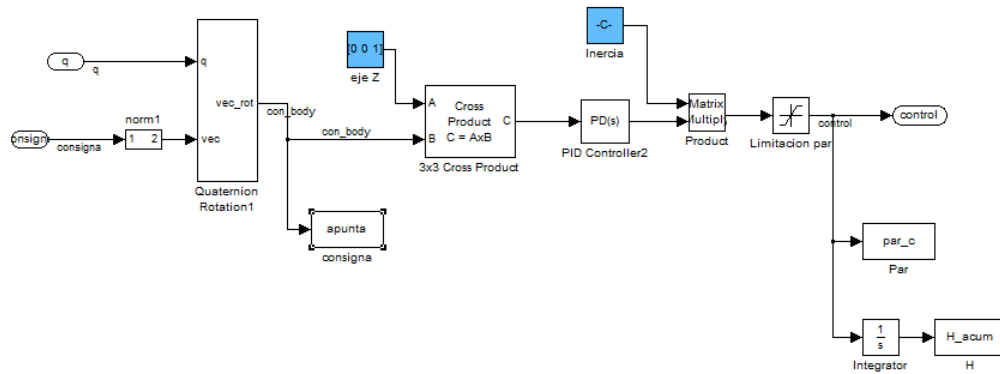


Figura 13.4: Subsistema para el control de actitud

#### ■ Input

- Variables en función del tiempo

**consigna** Vector al que se desea que apunte el eje z cuerpo (en ejes inerciales). Aunque en ejes inerciales esta dirección puede ser fija, no lo es en ejes cuerpo.

- q** Cuaternión de actitud del satélite
  - Constantes
    - eje\_z** Eje z del satélite en ejes cuerpo  $\mathbf{z} = (0\ 0\ 1)^T$
    - Inercia** Tensor de inercia del satélite respecto a ejes cuerpo
  - Parámetros
    - KP** Constante de control proporcional
    - KI** Constante de control integral
    - KD** Constante de control derivativa
    - Nmax** Par máximo que pueden proporcionar los actuadores
- Output
  - control** Par de control de actitud (Par\_c en otros módulos)
  - apunta** Variable que registra la consigna en ejes cuerpo
  - par\_c** Variable que registra el control de actitud
  - H\_acum** Variable que registra el momento acumulado por las ruedas de inercia.

Lo primero que hace el módulo es convertir la consigna en vector unitario y traducirlo a ejes cuerpo con el cuaternión de actitud y la librería de *Aerospac* de Simulink. Este vector se guarda en la variable *apunta* para su posterior tratamiento. Se multiplica vectorialmente el vector  $[0\ 0\ 1]^T$  por esta consigna. El resultado se trata por el controlador PID (de la biblioteca de Simulink). Este controlador lo sintoniza el usuario según se explica en los conceptos teóricos. Se multiplica el tensor de inercia por la salida del controlador y el resultado es el par de control. Se antepone un saturador para limitar el máximo y el mínimo del par aplicado. Se registra el par de control en la variable *Par\_c* y su acumulado en el tiempo en *H\_acum*. El *par\_c* de control es el que se aplica en el modelo dinámico explicado en 10.2.4

#### 13.2.4. Sintonización PID

Para sintonizar el control PID del satélite se ha hecho la simulación eliminando los pares perturbadores y el movimiento orbital alrededor de la órbita. Sólo se ha tenido en cuenta el modelo dinámico con su cálculo de actitud y

el subsistema de control como de muestra en la sección 10.2.4. Se comienza situando el satélite en una situación estable, el eje z del satélite apunta al eje z inercial y su velocidad angular es:

$$\begin{aligned}\omega &= \begin{Bmatrix} 0 \\ 0 \\ 0,5 \end{Bmatrix}, \\ q_0 &= \begin{Bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{Bmatrix}.\end{aligned}\tag{13.1}$$

Con la ecuación 13.1 se indica que se realiza un giro de  $0^\circ$  alrededor de cualquier eje, lo que es lo mismo, que en el instante inicial coinciden los ejes cuerpo con los ejes inerciales.

A continuación se le impone una consigna :

$$consigna = \begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix}.$$

Y por último se observan los resultados.

Se ha intentado el sintonizar el PID por el método Ziegler-Nichols y no ha dado resultado. El sistema es inestable incluso para constantes  $k_P$  muy pequeñas. Ver figura 13.5. Es probable que sea porque este método es bueno para sistemas retardados (con 'lag'). Sin embargo, el sistema del satélite se encuentra en el espacio sin rozamientos y cualquier cambio que se le aplique generará la aceleración angular correspondiente de forma inmediata.

Las gráficas denominadas Eje x, Eje y y Eje z muestran las tres componentes del vector consigna en ejes cuerpo en función del tiempo. La cuarta gráfica titulada Convergencia XY muestra la trayectoria del vector consigna en ejes cuerpo visto en la dirección z. Es una imagen que da una idea de cómo se apunta al objetivo.

Se procede a realizar la sintonización por el método prueba-error. Se comienza con una situación estable

1. Se obtiene una respuesta en bucle abierto y se determina lo que se debe mejorar
2. Se añade control proporcional para acortar el tiempo de subida.

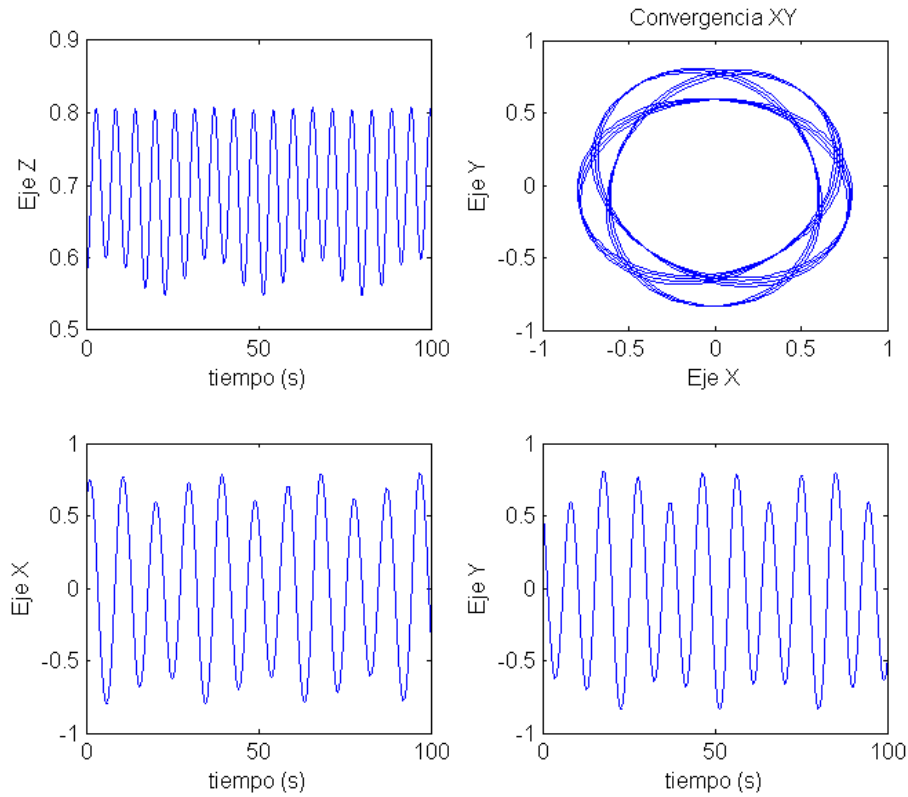


Figura 13.5: Orientación del satélite con control PID para el método de Ziegler-Nichols

Los valores de las constantes son  $k_P = 0,2$ ,  $k_I = 0$  y  $k_D = 0$

3. Se añade control derivativo para disminuir el overshoot.
4. Se añade control integral para eliminar el error de estado estacionario.
5. Se ajusta cada una de las tres constantes hasta obtener la respuesta deseada.

#### 13.2.4.1. Bucle abierto

Se obtienen los resultados que se muestran en la figura 13.6. Como era de esperar, el satélite apunta a una dirección fija del espacio distinta de la

consigna. Como el satélite gira sobre sí mismo, parece que la consigna describe un cono alrededor del eje  $z$  (Convergencia XY)

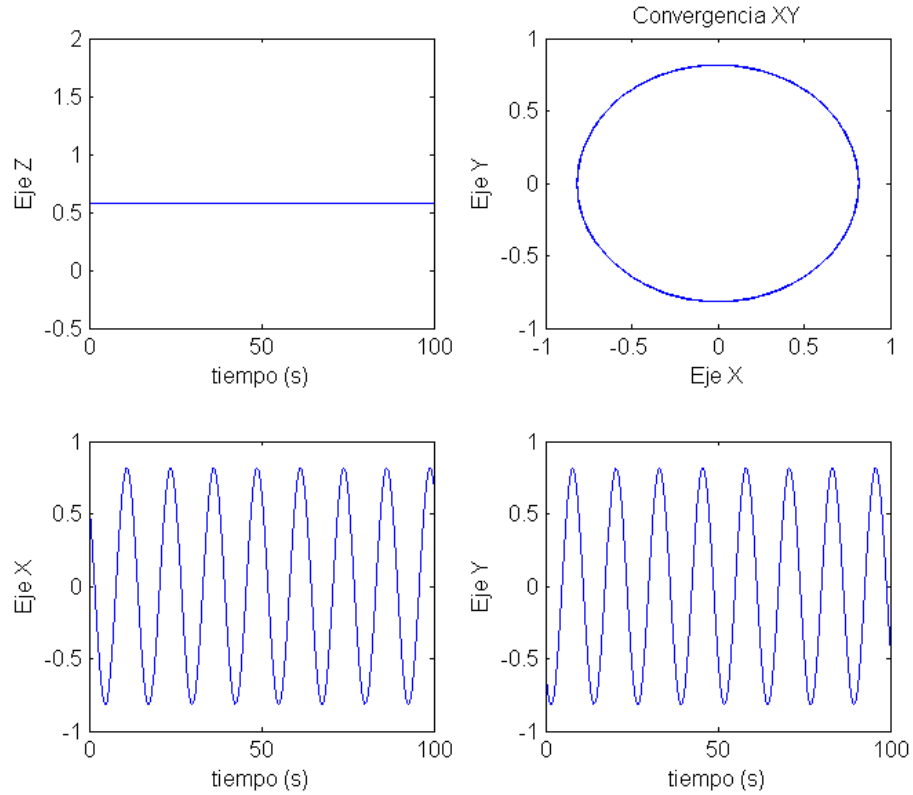


Figura 13.6: Orientación del sistema en bucle abierto  
Los valores de las constantes son  $k_P = 0$ ,  $k_I = 0$  y  $k_D = 0$

#### 13.2.4.2. Control proporcional

Se obtienen los resultados que se muestran en la figura 13.7. El sistema oscila alrededor de la consigna con un movimiento de precesión, pero no converge.

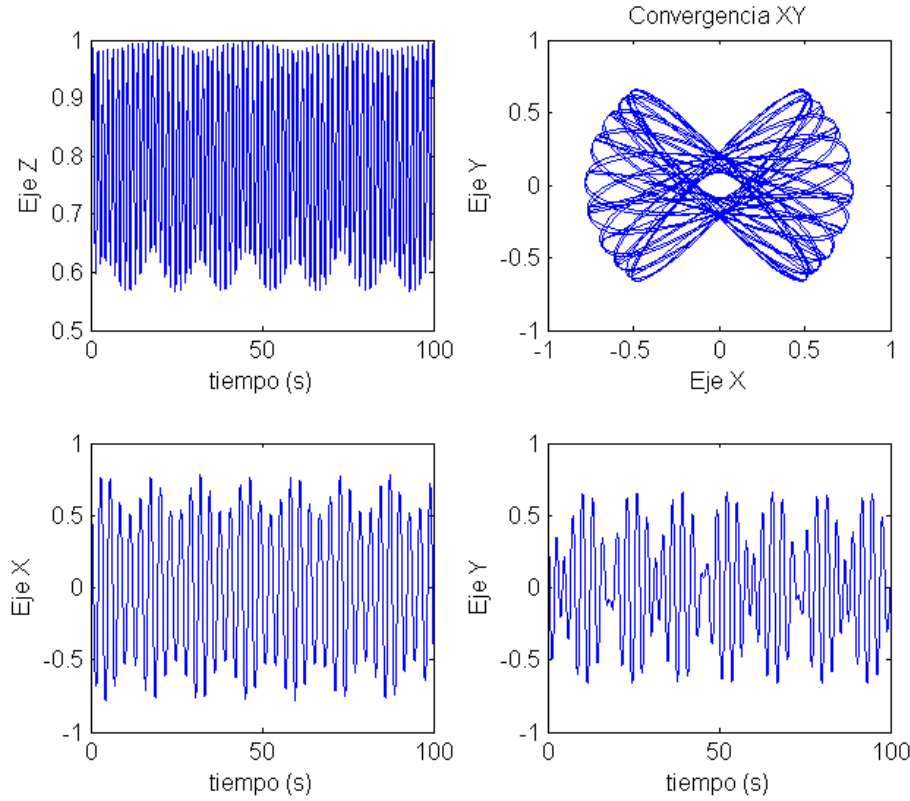


Figura 13.7: Orientación del sistema con control proporcional  
 Los valores de las constantes son  $k_P = 5$ ,  $k_I = 0$  y  $k_D = 0$

#### 13.2.4.3. Control proporcional derivativo

Los resultados de introducir añadir control derivativo al proporcional se muestran en la figuras 13.8 y 13.9. Se puede comprobar que el sistema converge en unos 18 segundos. Sigue habiendo overshoot apreciable pero se puede reducir cambiando los parámetros. Observando los valores numéricos de la variable *apunta* (ver subsección 13.2.3) en la figura 13.10 se comprueba que el error estacionario es del orden de  $10^{-6}$  tanto en el eje y como en el eje x. En ambos ejes el error no cambia de signo, lo que indica que hay un error steady-state.

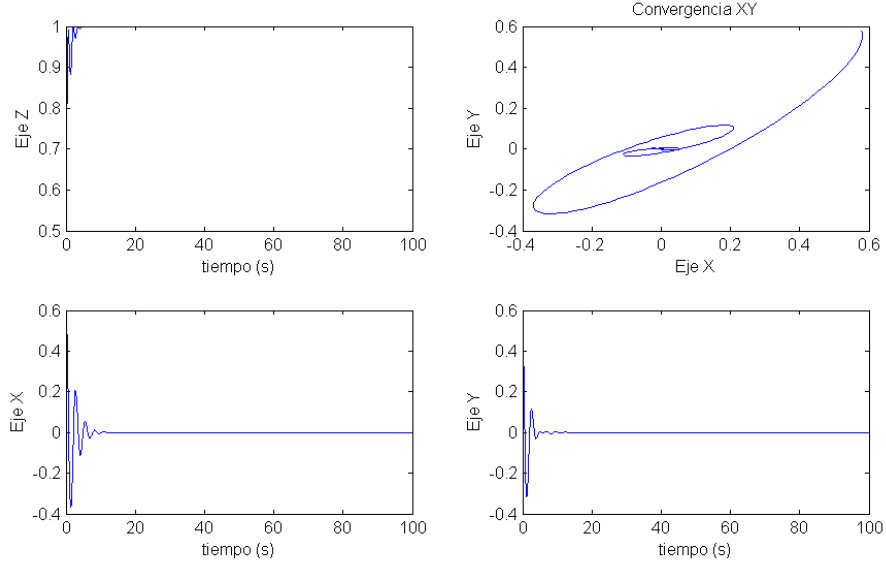


Figura 13.8: Orientación del sistema con control proporcional derivativo. Los valores de las constantes son  $k_P = 5$ ,  $k_I = 0$  y  $k_D = 1$ .

#### 13.2.4.4. Control proporcional derivativo integral

Se muestran los resultados obtenidos en las figuras 13.11, 13.12 y 13.13. El sistema sigue convergiendo, aumentando el tiempo de convergencia y la dispersión aumenta ligeramente debido al control integral. Se observa que el error de estado estacionario registrado en la variable *apunta* disminuye hasta  $10^{-8}$  en el eje y y  $10^{-7}$  en el eje x. Se observa que en el eje y oscila alrededor del 0, mientras que en el control proporcional se mantenía con signo constante. Estos resultados coinciden con lo esperado según la tabla 13.1.

#### 13.2.4.5. Control PID optimizado

Para este satélite en particular, que tiene el tensor de inercia:

$$Inercia = \begin{bmatrix} 40 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & 0 & 71 \end{bmatrix}.$$

Se ha sintonizado el control PID con  $k_P = 5$ ,  $k_D = 1.5$  y  $k_I = 2$ , obteniéndose las gráficas de las figuras 13.14 y 13.15. El error de estado estacionario se



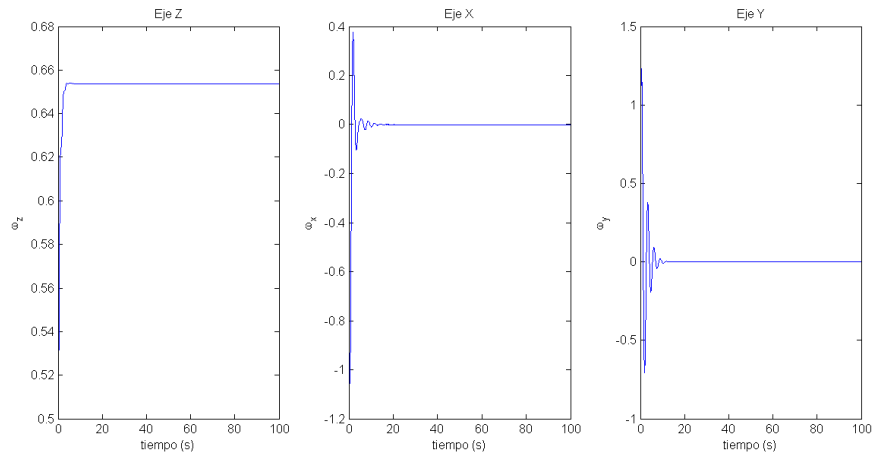
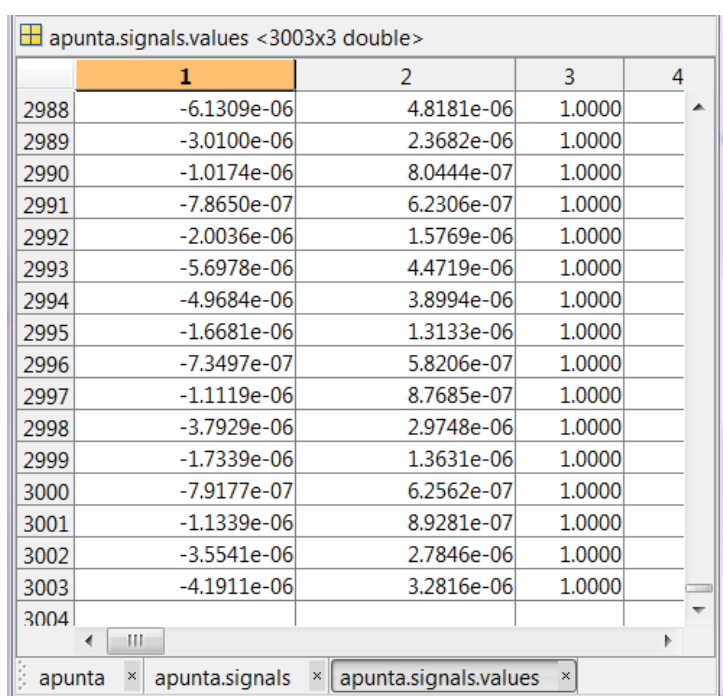


Figura 13.9: Velocidad angular con control proporcional derivativo  
 Los valores de las constantes son  $k_P = 5$ ,  $k_I = 0$  y  $k_D = 1$

encuentra en el valor de  $10^{-6}$ . La convergencia es en 7 segundos y el overshoot es muy pequeño.



	1	2	3	4
2988	-6.1309e-06	4.8181e-06	1.0000	
2989	-3.0100e-06	2.3682e-06	1.0000	
2990	-1.0174e-06	8.0444e-07	1.0000	
2991	-7.8650e-07	6.2306e-07	1.0000	
2992	-2.0036e-06	1.5769e-06	1.0000	
2993	-5.6978e-06	4.4719e-06	1.0000	
2994	-4.9684e-06	3.8994e-06	1.0000	
2995	-1.6681e-06	1.3133e-06	1.0000	
2996	-7.3497e-07	5.8206e-07	1.0000	
2997	-1.1119e-06	8.7685e-07	1.0000	
2998	-3.7929e-06	2.9748e-06	1.0000	
2999	-1.7339e-06	1.3631e-06	1.0000	
3000	-7.9177e-07	6.2562e-07	1.0000	
3001	-1.1339e-06	8.9281e-07	1.0000	
3002	-3.5541e-06	2.7846e-06	1.0000	
3003	-4.1911e-06	3.2816e-06	1.0000	
3004				

Figura 13.10: Error estacionario con control PD

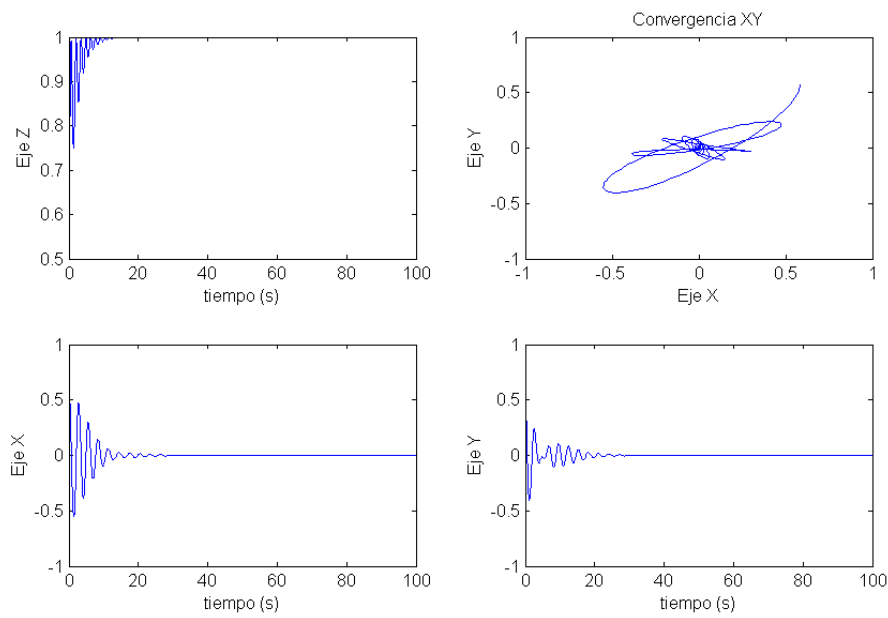


Figura 13.11: Orientación del sistema con control proporcional integral derivativo

Los valores de las constantes son  $k_P = 5$ ,  $3$  y  $k_D = 1$

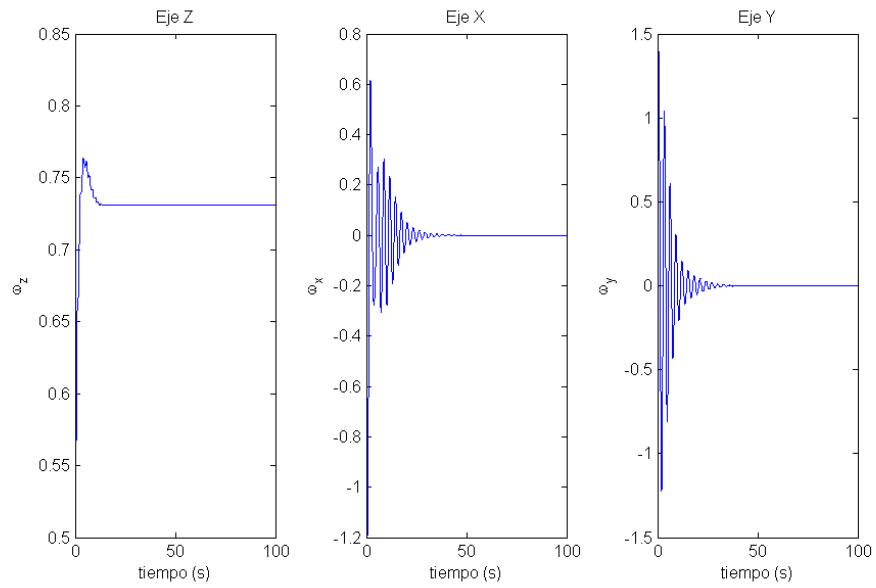


Figura 13.12: Velocidad angular con control proporcional integral derivativo  
 Los valores de las constantes son  $k_P = 5$ ,  $k_I = 3$  y  $k_D = 1$

	1	2	3
2991	6.6271e-07	-7.3823e-09	1.0000
2992	1.0404e-06	-6.7252e-09	1.0000
2993	3.7187e-06	3.5278e-08	1.0000
2994	1.6658e-06	-6.7747e-09	1.0000
2995	7.2761e-07	-2.8498e-08	1.0000
2996	1.0720e-06	-2.7296e-08	1.0000
2997	3.4914e-06	1.2565e-08	1.0000
2998	6.0671e-06	5.6808e-08	1.0000
2999	2.9557e-06	-5.4881e-09	1.0000
3000	9.6999e-07	-4.6375e-08	1.0000
3001	7.4339e-07	-5.3974e-08	1.0000
3002	1.9630e-06	-3.3790e-08	1.0000
3003	5.6555e-06	3.5560e-08	1.0000
3004	4.9325e-06	1.9574e-08	1.0000
3005	1.6422e-06	-4.6901e-08	1.0000
3006	7.1476e-07	-6.6648e-08	1.0000
3007	1.9886e-07	-7.7933e-08	1.0000
3008			

Figura 13.13: Error estacionario con control PID  
 Los valores de las constantes son  $k_P = 5$ ,  $k_I 3$  y  $k_D = 1$

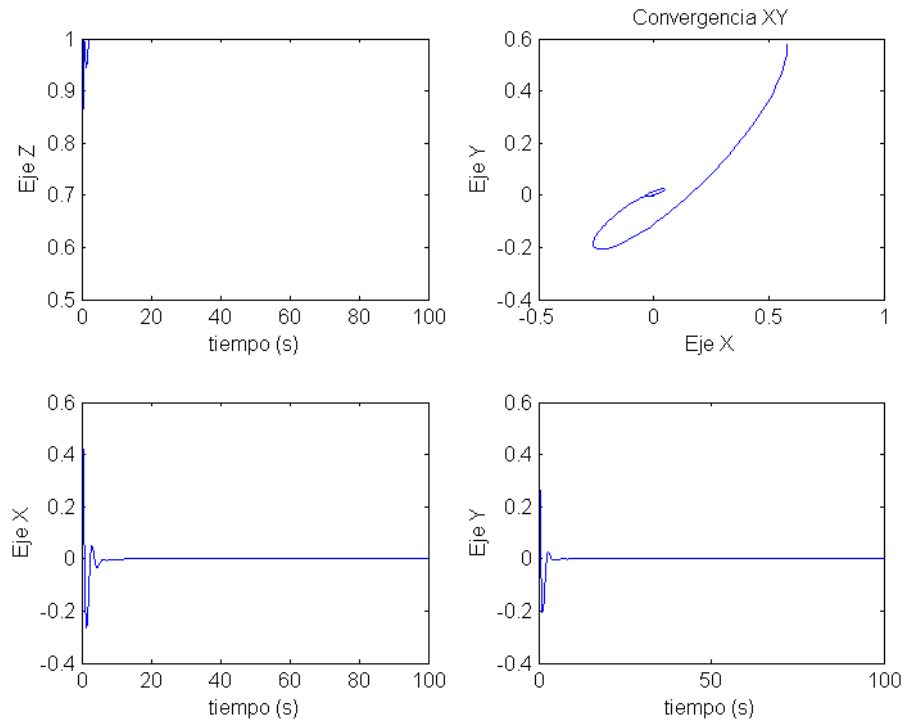


Figura 13.14: Orientación del sistema con control proporcional derivativo  
 Los valores de las constantes son  $k_P = 5$ ,  $k_I = 1,5$  y  $k_D = 2$

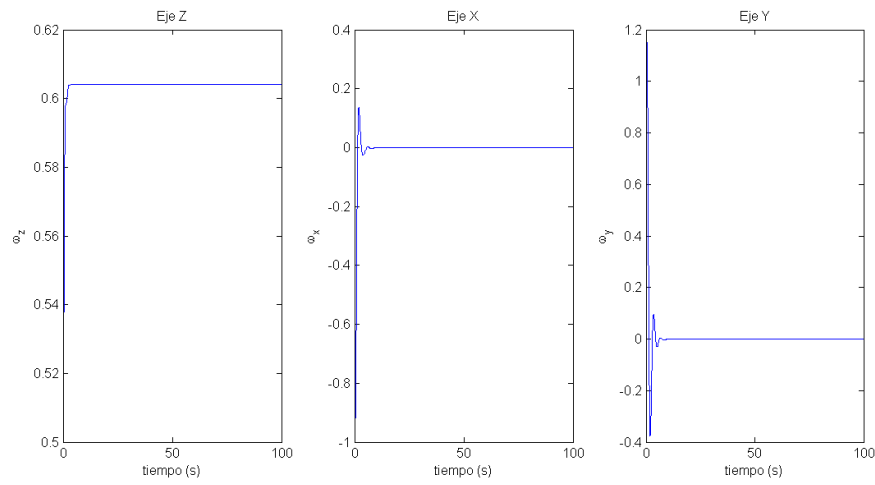


Figura 13.15: Velocidad angular con control proporcional derivativo  
 Los valores de las constantes son  $k_P = 5$ ,  $k_I = 1,5$  y  $k_D = 2$





# Capítulo 14

## Ejecución del modelo

### 14.1. Introducción

En este capítulo se va a proceder a observar el comportamiento del modelo completo, es decir, con el cálculo de órbita y de pares perturbadores aplicados a la dinámica del satélite. Además se harán experimentos para comprobar el funcionamiento del sistema de control frente a cambios en los parámetros del sistema.

### 14.2. Establecimiento de consigna

Se ha diseñado un pequeño módulo para establecer una consigna. Se puede observar en la figura 14.1.

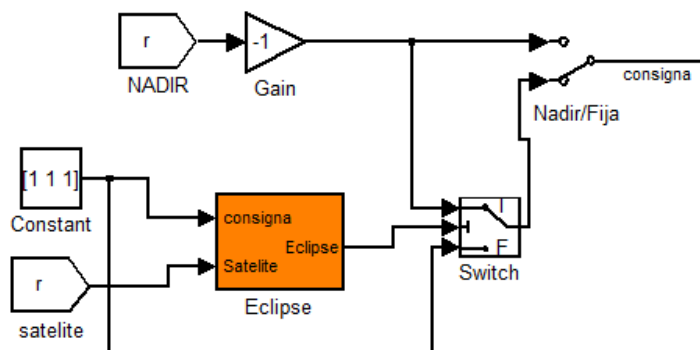


Figura 14.1: Subsistema para la asignación de consigna

El funcionamiento es sencillo. El módulo tiene un interruptor manual y uno automático. El interruptor manual permite al usuario elegir entre la consigna nadir y otra. En la imagen se ve que la consigna alternativa es  $[1 \ 1 \ 1]$ . Esto quiere decir que apunta a esa dirección en los ejes inerciales. Se evalúa si la Tierra eclipsa esta dirección del mismo modo que se explica en el apartado 11.4. El interruptor automático da la consigna establecida si no está eclipsada por la Tierra y envía el nadir en caso de que sí esté eclipsada. Así durante la simulación de la órbita se podrá apreciar las dos maniobras de entrada y salida de zona de sombra.

### 14.3. Comportamiento dinámico

Se estudia la órbita completa de un satélite y con las siguientes características:

- Datos de la órbita
  - Semieje mayor  $a = 16378 \text{ km}$
  - Excentricidad  $e = 0,3$
  - Longitud del nodo ascendente  $\Omega = 20^\circ$
  - Inclinação de la órbita  $i = 25^\circ$
  - Argumento del perigeo  $\omega = 50^\circ$
  - Período de la órbita  $T = \frac{2\pi}{\sqrt{\mu_T/a^3}} = 348 \text{ min}$
- Datos del satélite
  - Tensor de inercia y ajuste de control correspondientes a los definidos en la sección 13.2.4.5
  - Centro de presión solar y aerodinámica  $c_{pa} = c_{ps} = [0,01 \ 0,005 \ -0,02]^T$ .
  - Largo, ancho y alto  $LWH = [0,4 \ 0,5 \ 0,7]^T$ .
  - Coeficiente aerodinámico  $c_D = 2$
  - Coeficiente de reflexión solar  $q = 0,6$
  - Dipolo magnético residual.  $D = [1 \ 0,5 \ 0,7]^T$

- Velocidad angular inicial  $\omega = [0,1 \ 0,1 \ 0,5]^T$ . Realmente esta velocidad se estabiliza a los pocos segundos en una  $\omega_z$  constante (debido al sistema de control que no tiende a eliminar la componente z porque esta componente sola no desvía la orientación).

### 14.3.1. Orientación hacia una dirección fija en ejes inerciales

Las gráficas obtenidas son las que se muestran en las figuras 14.2, 14.3, 14.4, 14.5 y 14.6.

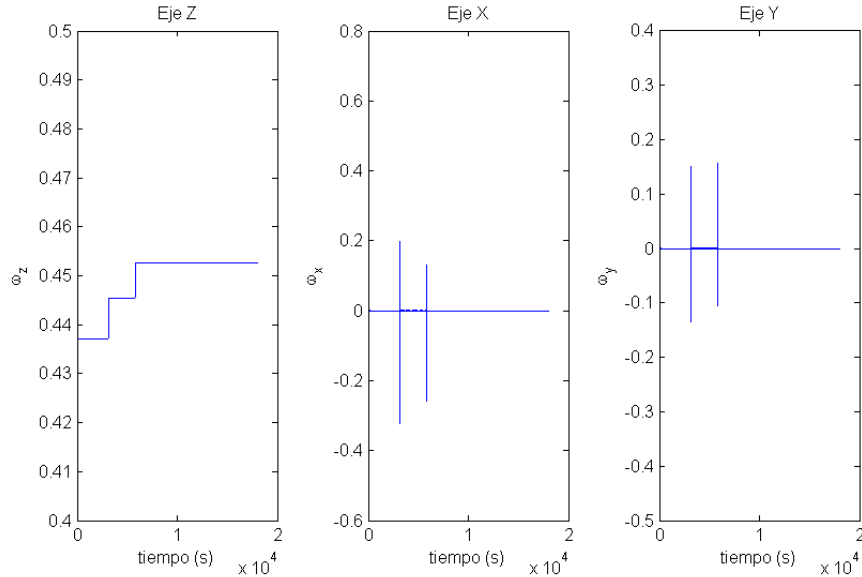


Figura 14.2: Velocidad angular  $\omega$  en los tres ejes

En esta gráfica se observa que en el eje x y en el eje y se efectúan dos maniobras. Corresponden a la entrada en la sombra ya que en este momento se cambia la consigna a nadir y la salida de la zona de sombra cuando vuelve a establecerse la consigna original. A este nivel de escala no se aprecia el tiempo de maniobra. Resulta curioso el comportamiento de la rotación en torno al eje z. El valor de  $\omega_z$  al inicio es 0.5, siendo  $\omega_x = \omega_y = 0,1$ . En el primer apuntamiento a consigna impone la reducción a 0 de  $\omega_x$  y  $\omega_y$ . No se aprecia en la figura pero son los parámetros iniciales y se ha verificado

observando las tablas numéricas de resultados. En esta primera maniobra el valor de  $\omega_z$  se ha decrementado de 0.5 a 0.437, a la vez que  $\omega_x$  y  $\omega_y$  se han reducido a 0. Sin embargo, en las dos maniobras de entrada y salida de zona eclipsada, el valor de  $\omega_z$  se ha vuelto a incrementar (a 0.445 y 0.452).

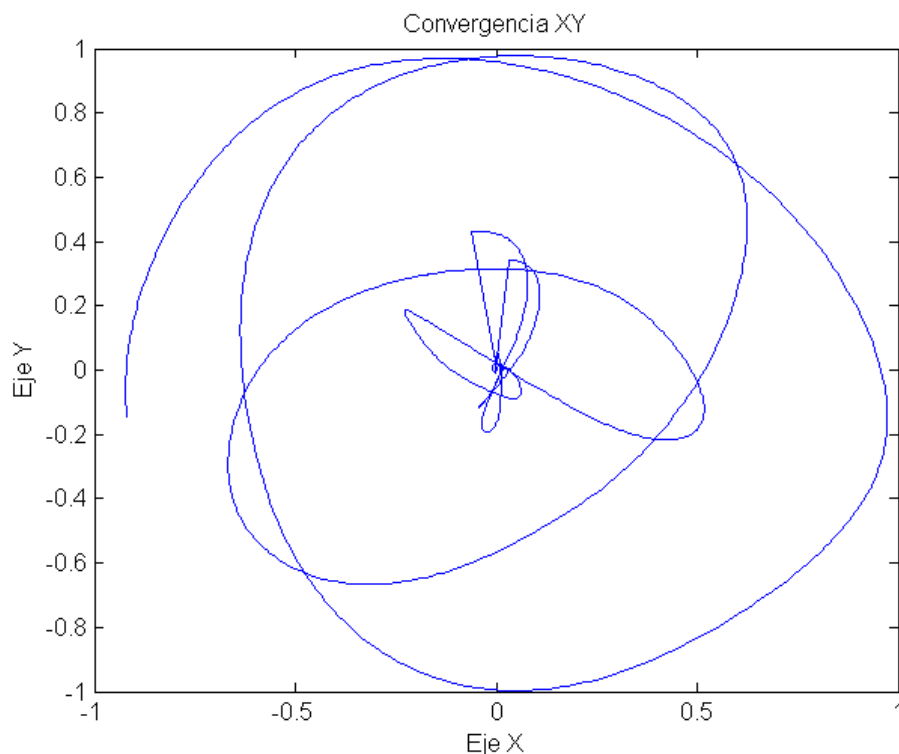


Figura 14.3: Orientación del satélite, convergencia XY

En la gráfica 14.3 se observa la trayectoria del vector consigna en ejes cuerpo proyectada sobre el plano xy. Realmente en la gráfica se ven tres maniobras. La primera comienza con el extremo de la traza (cerca del punto  $(-1 \ 0)$ ). En la primera maniobra la trayectoria da menos de tres vueltas para centrar el objetivo. Las otras dos maniobras se observan como salidas rectilíneas del centro y vuelta al centro de forma curvilínea. Se ve que estas dos maniobras convergen muy rápido y con muy poca dispersión ya que se parte de una situación en la que  $\omega_x = \omega_y = 0$ .

Se ha hecho un zoom en la figura 14.4 para observar la zona central. Se ve que hay dos zonas de convergencia, una de ellas es la corona circular

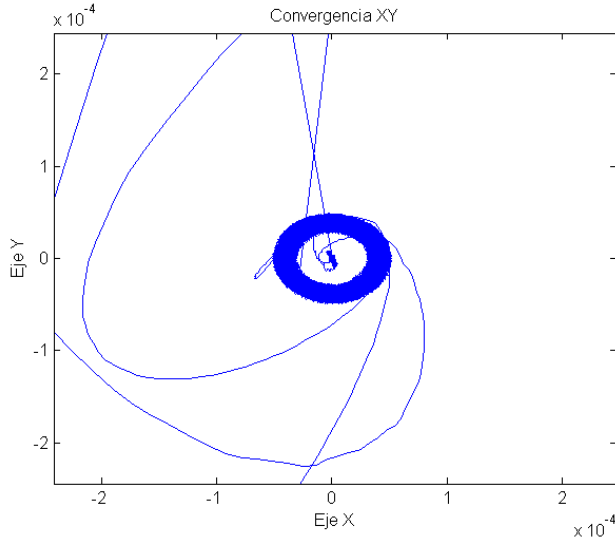


Figura 14.4: Detalle de la figura 14.3

(téngase en cuenta que la escala no es la misma) de diámetro del orden de  $10^{-4}$ . La segunda es mucho más pequeña y está en su centro. Estudiando las siguientes gráficas se deduce que la zona más precisa corresponde a la primera convergencia, donde la  $\omega_z$  es menor.

En la figura 14.5 se representa el par proporcionado por el controlador PID para realizar las maniobras requeridas. La primera observación es que sobre el eje  $z$  no se ha actuado. Una rotación sobre este eje no desvía la consigna, por lo que el control decide no alterarla. Los errores proyectados en el eje  $z$  son muy pequeños comparados a los proyectados en los ejes  $x$  y  $y$ . Lo segundo que se observa son los extremos de los pares ( $\pm 7$ ). Es el parámetro que se ha definido para el limitador de par colocado después del control PID (figura 13.4). Lo tercero y más interesante es que desde la primera hasta la segunda maniobra el par es muy pequeño, mientras que a partir de la segunda maniobra el par oscila considerablemente ( $\pm 2$  en el eje  $x$  y  $\pm 1$  en el eje  $y$ ). No se puede culpar a los pares perturbadores esta oscilación porque después de la primera maniobra las perturbaciones son las mismas que después de la tercera. Lo que ha sucedido es que se ha incrementado  $\omega_z$ , lo que supone que corregir una actitud cuesta más par si el satélite es estabilizado por autospin, como era de esperar. Probablemente ajustando el parámetro  $k_D$  se reduzca la inestabilidad en el par.

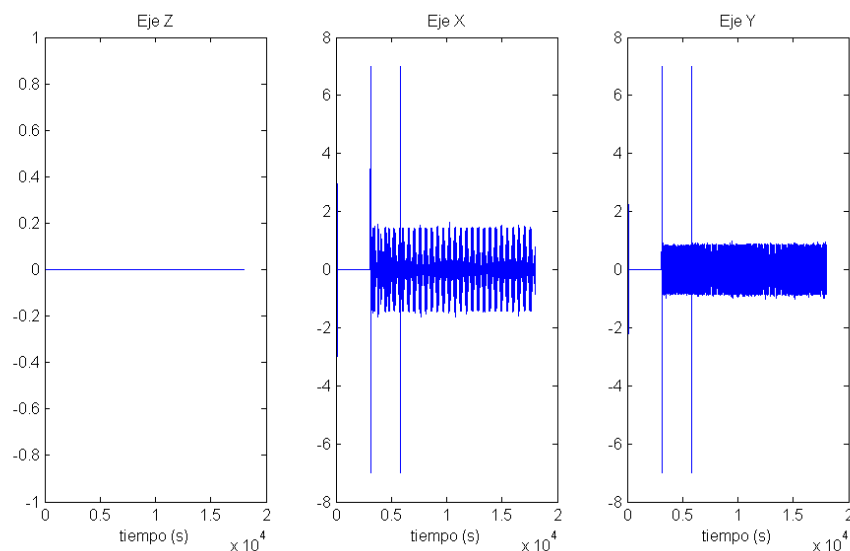


Figura 14.5: Par de control en los tres ejes

En la figura 14.6 se muestra la zona ampliada de la tercera maniobra. La maniobra dura alrededor de 7 u 8 segundos. Las amplitudes de los pares aplicados son similares antes y después de la maniobra, con lo que se concluye que la radiación solar no es la responsable de estas oscilaciones.

## 14.4. Acumulación de momento

Para diseñar el tamaño de las ruedas de inercia y de reacción es necesario tener una idea del valor del momento acumulado a lo largo una órbita o en una maniobra tipo. La simulación efectuada incluye ambas situaciones, que se muestran en la figura 14.7, donde se observa el momento acumulado. Por estar el satélite girando sobre su eje z el momento acumulado debido a perturbaciones es nulo. Las maniobras sí que provocan acumulación de momento en las ruedas que deberán ser descargadas con otros actuadores (microcohetes o magnetopares)

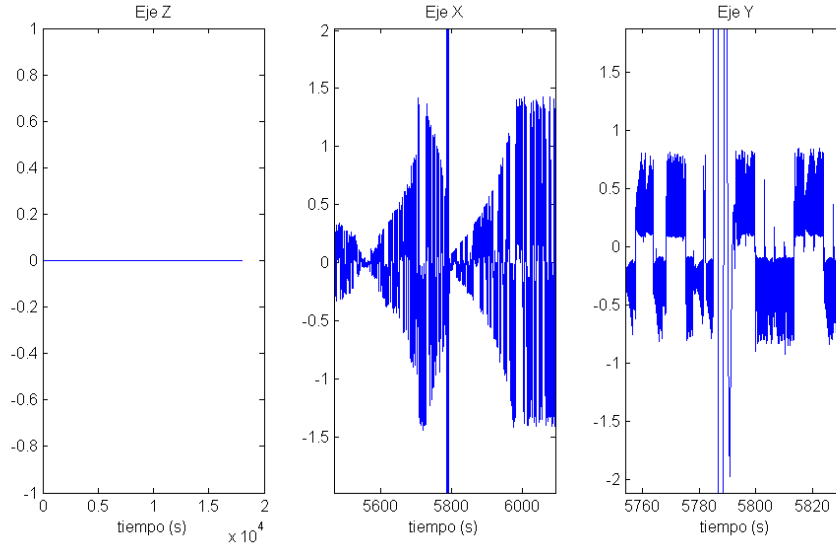


Figura 14.6: Detalle de la figura 14.5

## 14.5. Variaciones del comportamiento del modelo dinámico

En este apartado se va a estudiar el comportamiento del sistema dinámico y su control cuando se varían los parámetros másicos. En particular, se va a modificar los momentos de inercia de modo que los ejes cuerpo dejan de ser principales.

### 14.5.1. Escenario 1: $J_{xy} \neq 0$

Se ha modificado el tensor de inercia de modo que la simulación se realizará con:

$$Inercia = \begin{bmatrix} 40 & 20 & 0 \\ 20 & 50 & 0 \\ 0 & 0 & 71 \end{bmatrix}.$$

Se estudia la eficiencia del control sin perturbaciones, y con los mismos parámetros que en la subsección 13.2.4.5.

Los resultados se muestran en las figuras 14.8 y 14.9.

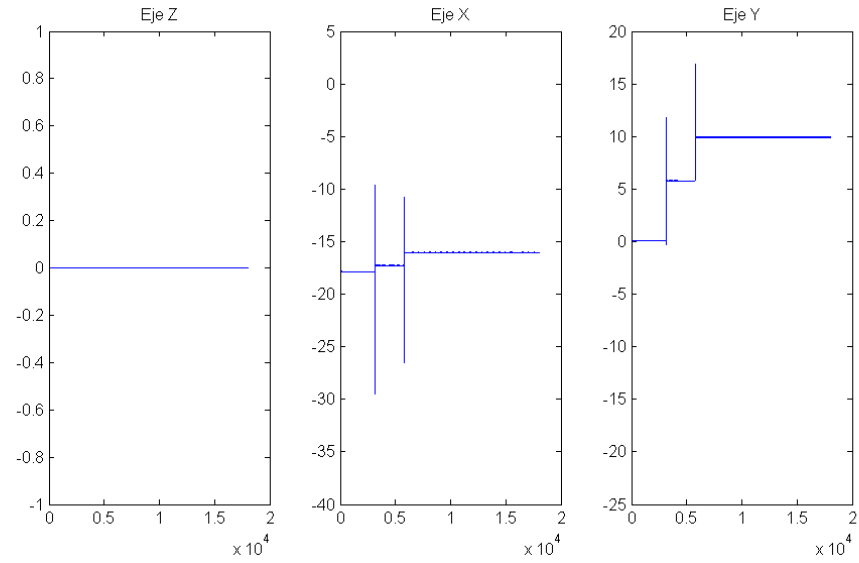


Figura 14.7: Momento acumulado en las tres direcciones.

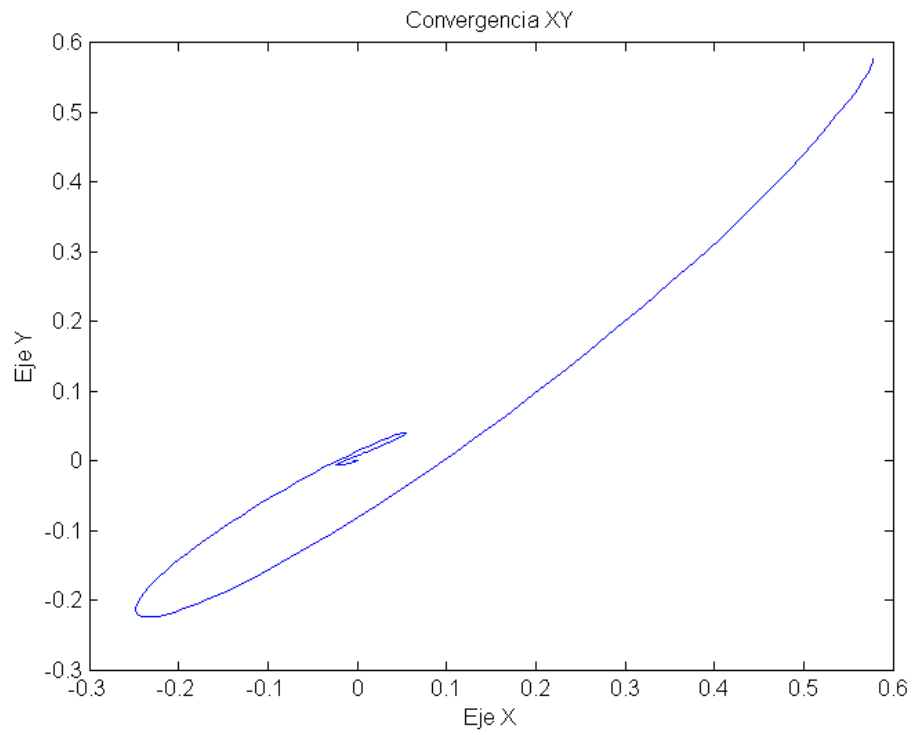


Figura 14.8: Convergencia del apuntamiento cuando  $J_{xy} \neq 0$



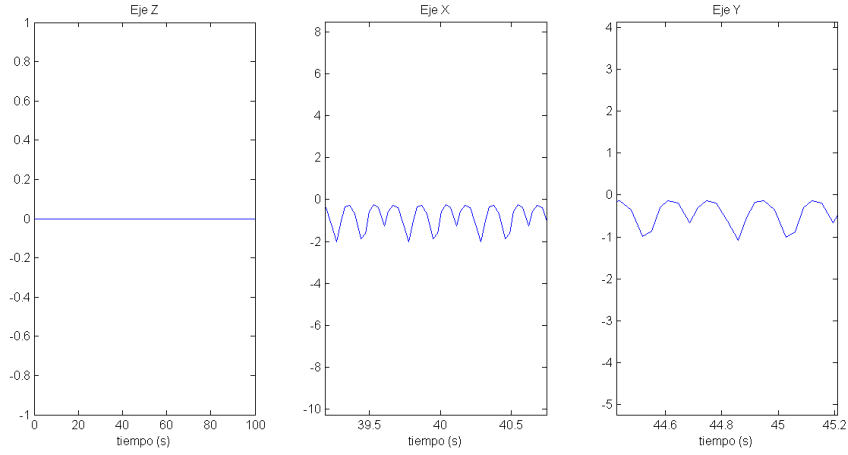


Figura 14.9: Detalle ampliado del par de control con  $J_{xy} \neq 0$  una vez estabilizado

Se observa que el comportamiento es muy parecido a cuando los ejes x e y son principales de inercia.

#### 14.5.2. Escenario 2: $J_{yz} \neq 0$

En este escenario el tensor de inercia que se usa en la simulación es:

$$Inercia = \begin{bmatrix} 40 & 0 & 0 \\ 0 & 50 & 10 \\ 0 & 10 & 71 \end{bmatrix}.$$

Al igual que en el escenario anterior, se simula sin perturbaciones y con el control PID sintonizado con los mismos parámetros.

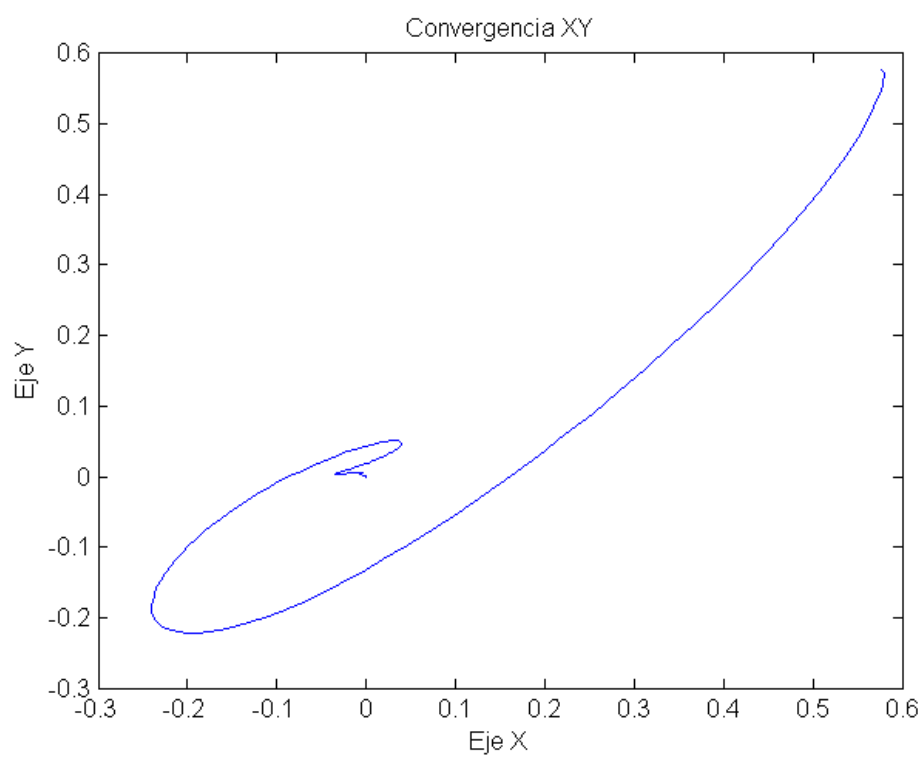


Figura 14.10: Convergencia del apuntamiento cuando  $J_{yz} \neq 0$

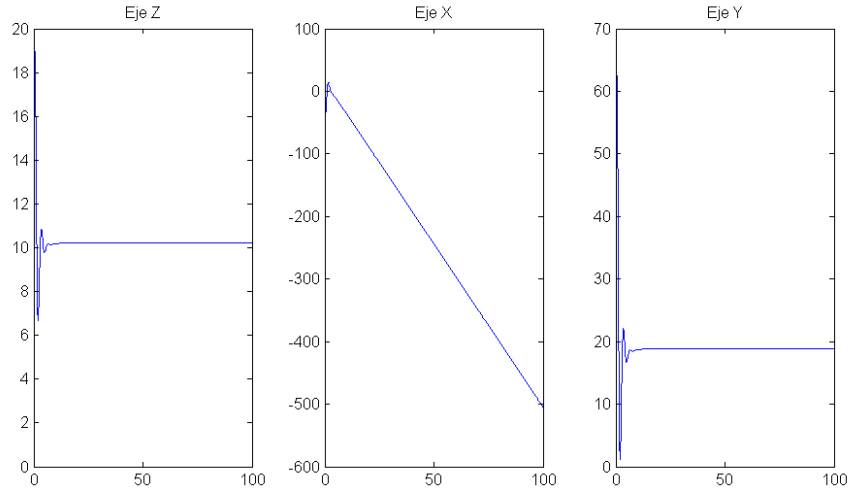


Figura 14.11: Momento acumulado cuando  $J_{yz} \neq 0$

En las figuras 14.10 y 14.11 se observa que la convergencia es buena y rápida pero para mantenerla es necesario aplicar continuamente un par neto según el eje x.

### 14.5.3. Escenario 3: $J_{xz} \neq 0$

En este escenario el tensor de inercia que se usa en la simulación es:

$$Inercia = \begin{bmatrix} 40 & 0 & 10 \\ 0 & 50 & 0 \\ 10 & 0 & 71 \end{bmatrix}.$$

Al igual que en el escenario anterior, se simula sin perturbaciones y con el control PID sintonizado con los mismos parámetros.

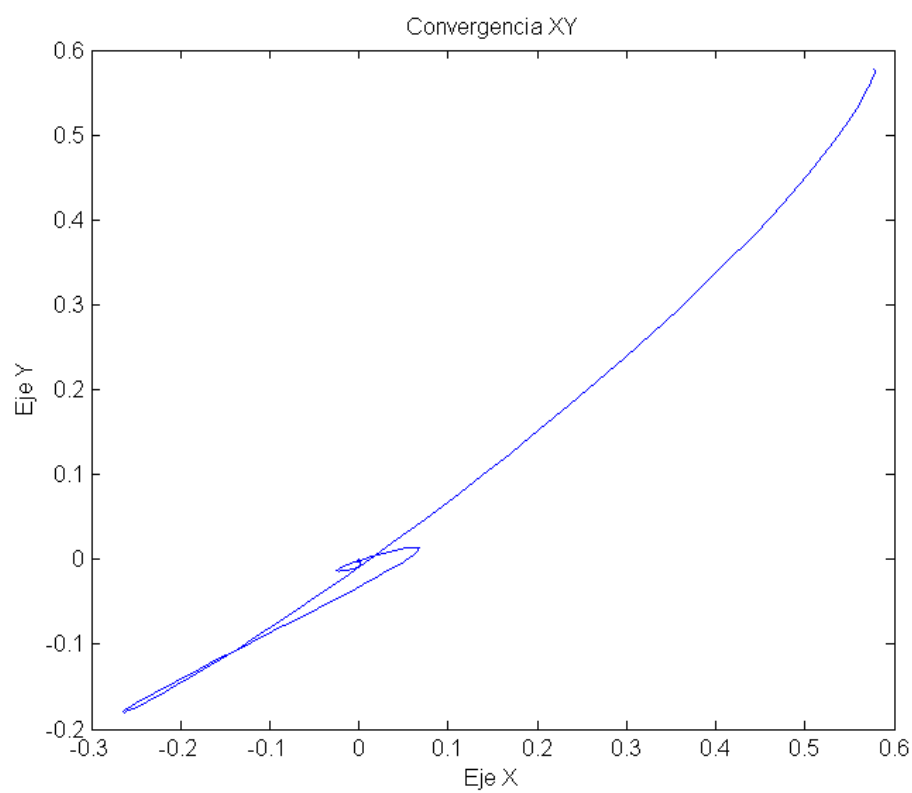


Figura 14.12: Convergencia de apuntamiento cuando  $J_{xz} \neq 0$

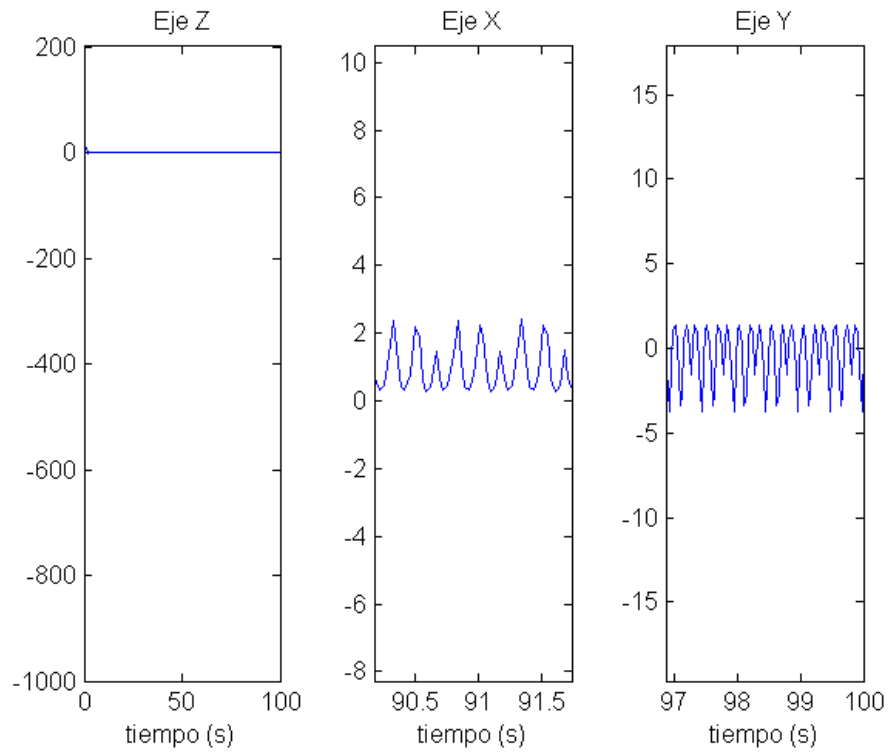


Figura 14.13: Pares de control para el caso de  $J_{xz} \neq 0$

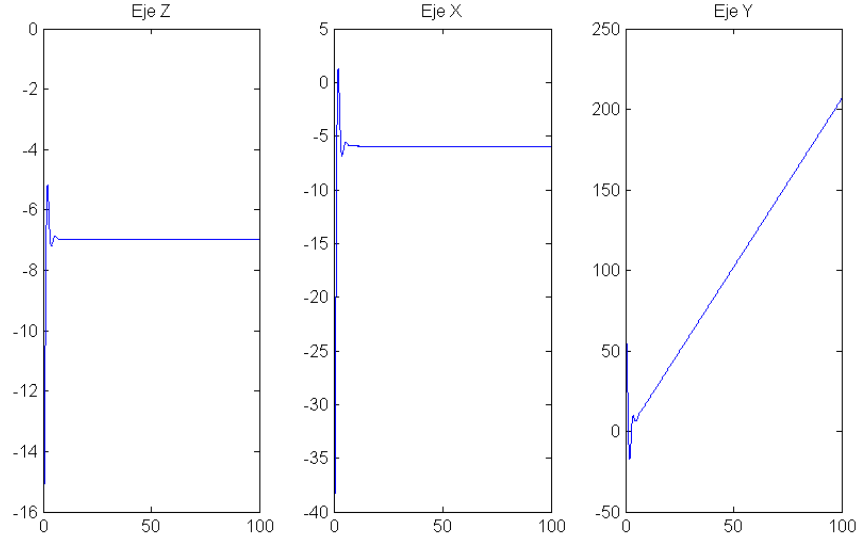


Figura 14.14: Momentos acumulados cuando  $J_{xz} \neq 0$

En las figuras 14.12, 14.13 y 14.14 se ve que el comportamiento es parecido al escenario anterior pero es el eje y el que acumula el momento. Las amplitudes de las oscilaciones de los pares según los ejes x e y son de 2 y 5 Nm respectivamente.

#### 14.5.4. Escenario 4: $J_{xz} \neq 0$ y $J_{yz} \neq 0$

En este escenario el tensor de inercia que se usa en la simulación es:

$$Inercia = \begin{bmatrix} 40 & 0 & 10 \\ 0 & 50 & 10 \\ 10 & 10 & 71 \end{bmatrix}.$$

Al igual que en el escenario anterior, se simula sin perturbaciones y con el control PID sintonizado con los mismos parámetros.

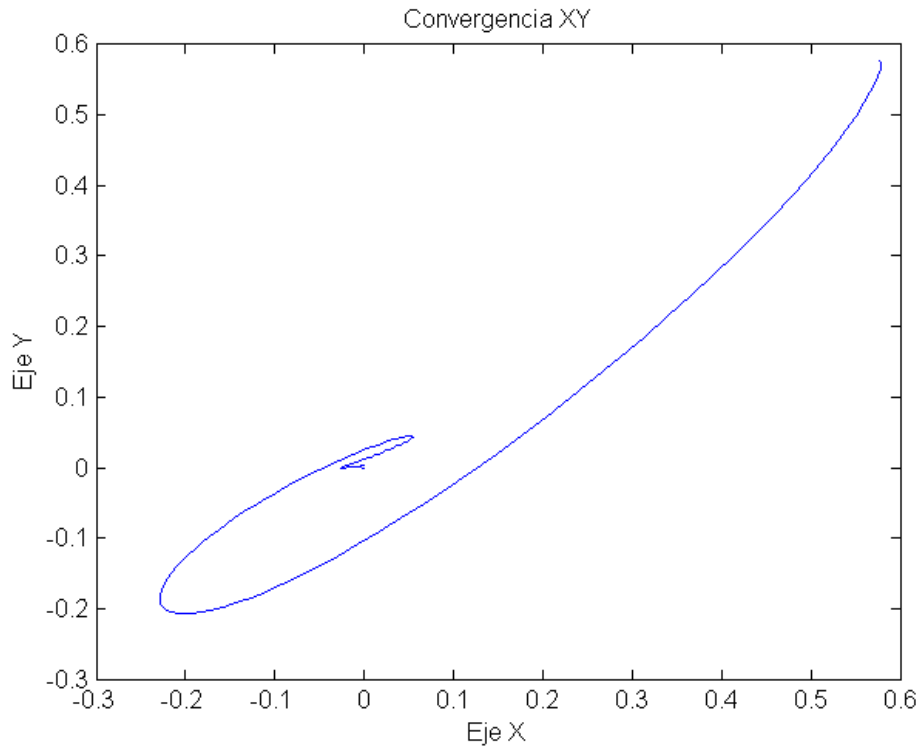


Figura 14.15: Convergencia cuando  $J_{xz} \neq 0$  y  $J_{yz} \neq 0$

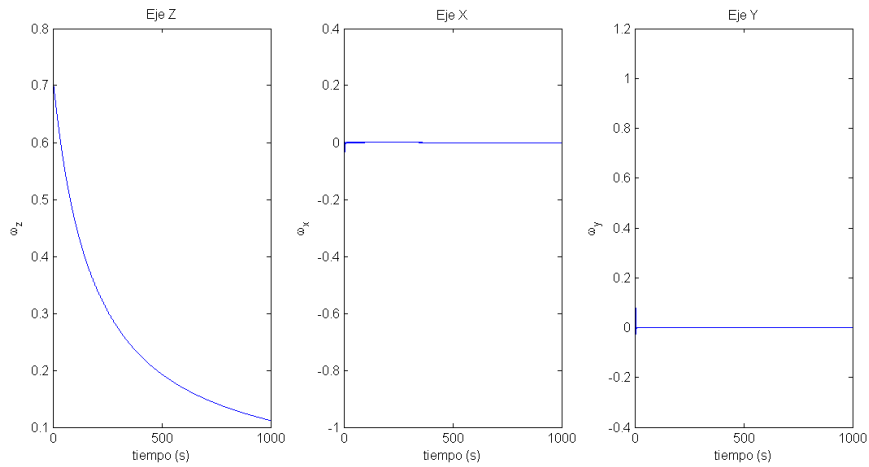


Figura 14.16: Velocidad angular cuando  $J_{xz} \neq 0$  y  $J_{yz} \neq 0$

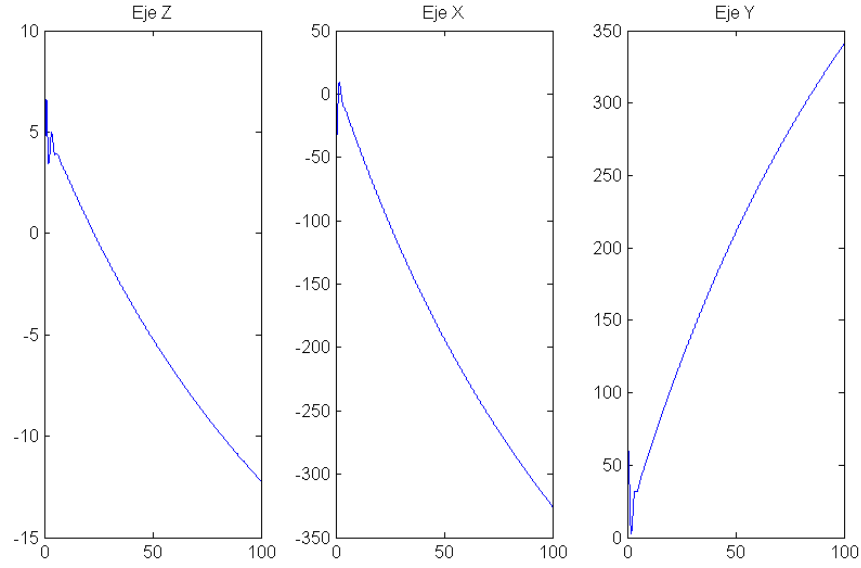


Figura 14.17: Momento acumulado con  $J_{xz} \neq 0$  y  $J_{yz} \neq 0$

En las figuras 14.15, 14.16 y 14.17 se observa que para tener el satélite apuntando a una dirección fija el control se ve obligado a disminuir las tres componentes de la velocidad angular. La convergencia es rápida pero costosa porque el momento acumulado se incrementa continuamente y obligará a descargar las ruedas con mucha frecuencia.

#### 14.5.5. Escenario 5: $J_{yz} \neq 0$ con perturbaciones

En este escenario no sólo se contempla la estabilidad del control sino también como responde a las perturbaciones. La simulación no se extiende a una órbita sino sólo a una maniobra.

El tensor de inercia usado es

$$Inercia = \begin{bmatrix} 40 & 0 & 0 \\ 0 & 50 & 5 \\ 0 & 5 & 71 \end{bmatrix}.$$



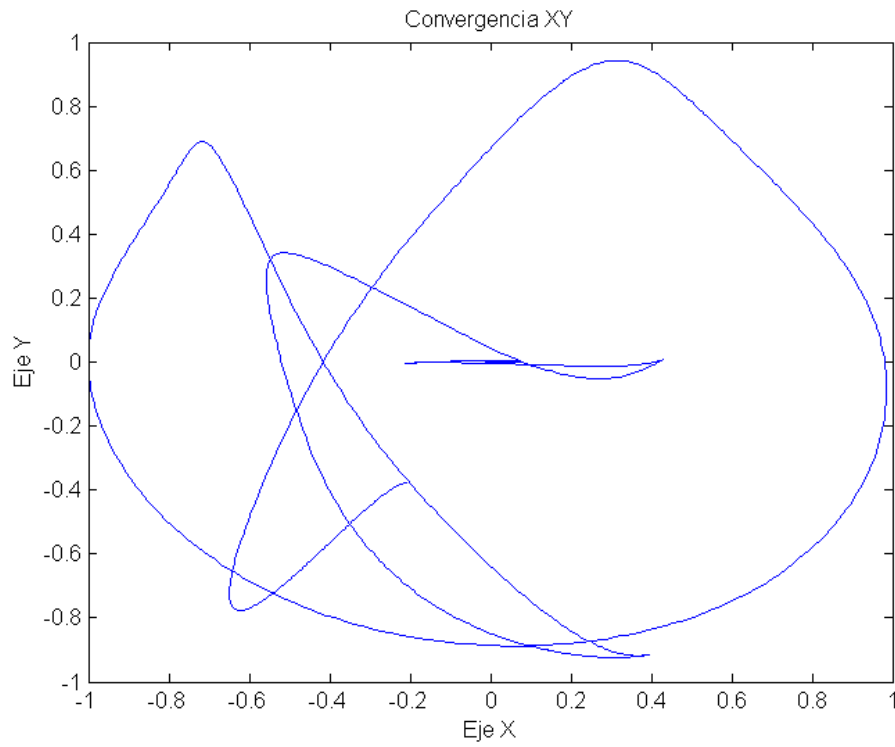


Figura 14.18: Convergencia cuando  $J_{yz} \neq 0$  con perturbaciones

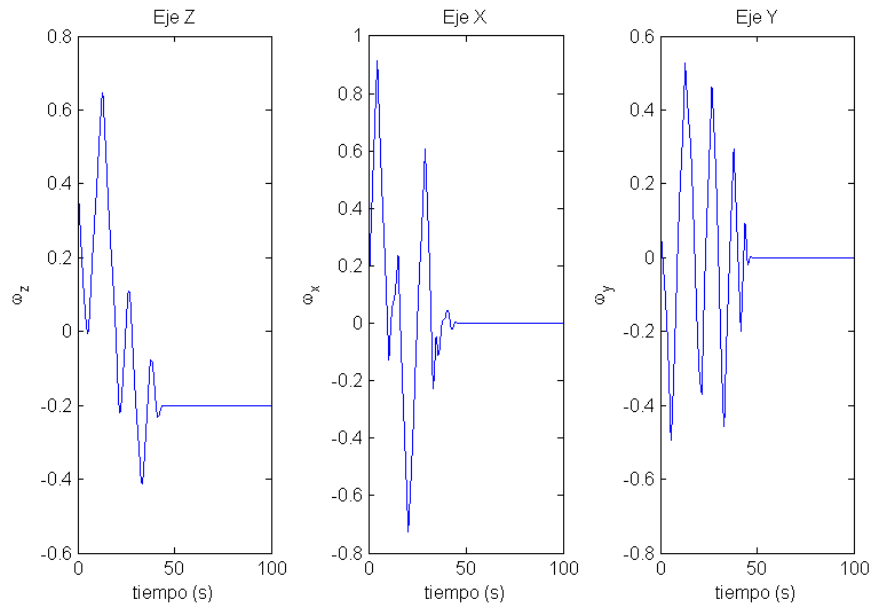


Figura 14.19: Velocidad angular cuando  $J_{yz} \neq 0$  con perturbaciones

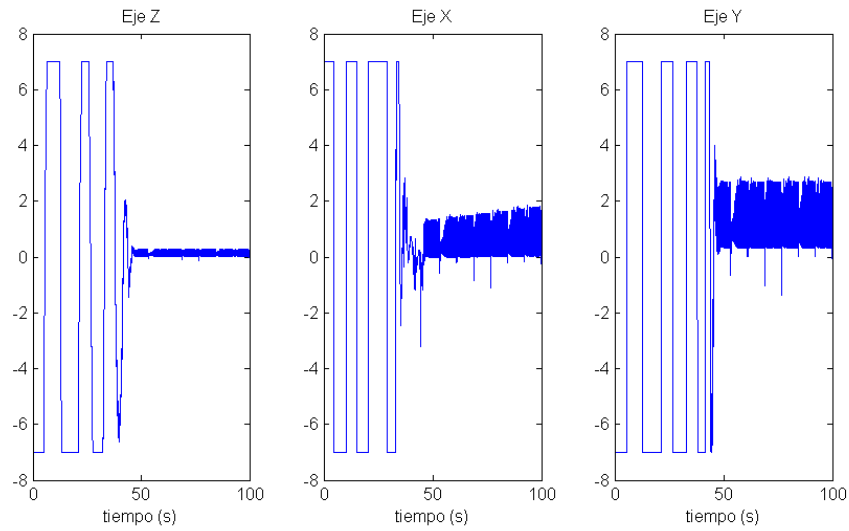


Figura 14.20: Pares de control cuando  $J_{yz} \neq 0$  con perturbaciones

En las figuras 14.18, 14.19 y 14.20 se observa que se tarda bastante más

en conseguir el apuntamiento. Para ello los actuadores han oscilado entre los valores máximos impuestos durante casi un minuto. Este es el primer caso en que la rotación del satélite se invierte en el eje  $z$ . Parece que las perturbaciones inestabilizan al control pero éste finalmente consigue dominar al satélite. Aunque no se muestra el par acumulado de la figura 14.20, se deduce que continúan incrementándose en el régimen estacionario en los ejes  $x$  e  $y$  por tener valores medios diferentes de cero.

#### 14.5.6. Escenario 6: $J_{yz} \neq 0$ y $J_{xz} \neq 0$ con perturbaciones

En este último escenario se ha estudiado el satélite con tensor de inercia

$$Inercia = \begin{bmatrix} 40 & 0 & 5 \\ 0 & 50 & 5 \\ 5 & 5 & 71 \end{bmatrix}.$$

Se imponen los parámetros de control de los apartados anteriores y se somete a perturbaciones.

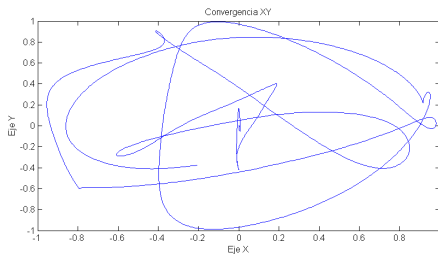


Figura 14.21: Convergencia cuando  $J_{xz} \neq 0$  y  $J_{yz} \neq 0$  con perturbaciones

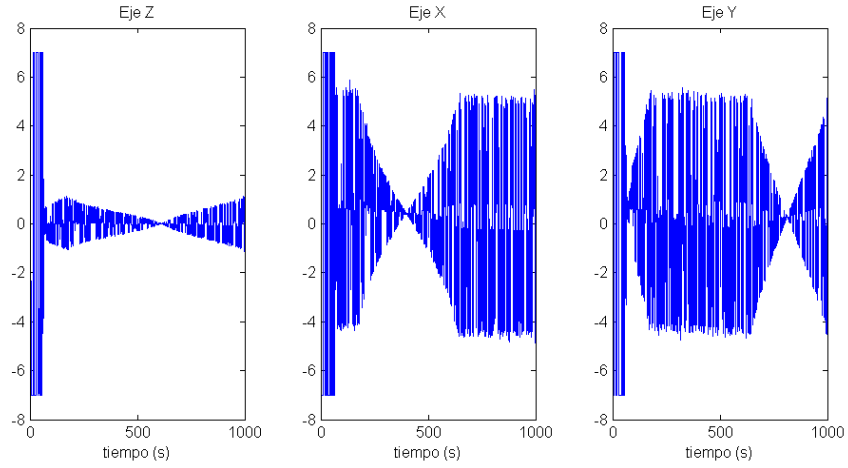


Figura 14.22: Pares de control cuando  $J_{xz} \neq 0$  y  $J_{yz} \neq 0$  con perturbaciones

En las figuras 14.21 y 14.22 se observa que el satélite tarda mucho en estabilizarse (alrededor de 50 seg) y que para mantenerlo los pares empleados son muy grandes. El satélite en esta situación consumirá mucha energía para mantener su actitud. Las ruedas estarán continuamente saturándose y se acortará su vida. Se ve que este es el escenario más desfavorable.

# Capítulo 15

## Conclusiones de la Parte III

### 15.1. Cumplimiento de objetivos

Los objetivos principales de este proyecto fueron simular las perturbaciones principales que afectan a la actitud de un satélite y diseñar un sistema de control de actitud.

Para ello se han seguido los siguientes pasos:

- Se realizó el modelo matemático que simula el movimiento de traslación de un satélite en una órbita estacionaria y el movimiento del Sol en la suya.
- Se diseñó un sistema de control para conseguir que un eje del satélite apunte a una dirección dada sin importar que éste gire sobre este eje
- Se sintonizó un control PID que consigue de una forma eficiente el objetivo de la misión cuando no hay perturbaciones
- Se comprobó que el mismo control es eficiente cuando el satélite está sometido a las condiciones de perturbación y cuando el satélite está desequilibrado másicamente

Habiendo seguido estos pasos, se puede concluir que el control diseñado es eficiente y cumple su misión.

### 15.2. Comentarios

Se puede extraer una serie de comentarios:

El tratamiento de sistemas de referencia con cuaterniones es eficiente y cómodo para este tipo de problemas.

En principio el cambio de sistemas de referencia se diseñó con matrices de cosenos directores, pero la complejidad conceptual de tener que integrar una matriz en el tiempo (la de actitud) y observando lo cómodo que es integrar un cuaternión (ecuaciones (9.5) y (9.6)) al final se decidió aprovechar este concepto lo máximo posible. La existencia de librerías de Simulink que operan con cuaterniones y con matrices permitió comprobar que ambas daban los mismos resultados y son perfectamente intercambiables. En la literatura sobre el tema, se ha visto dos nomenclaturas diferentes para los cuaterniones. En este proyecto se ha empleado la que corresponde a las librerías de Simulink y a la mayor parte de la bibliografía respecto al tema.

Las comprobaciones hechas con el primer modelo diseñado que fue el modelo dinámico del satélite y los buenos resultados obtenidos dan confianza en la herramienta Simulink para hacer simulaciones de sistemas complejos. La posibilidad de ensayar modelos por partes ayuda a corregir errores y tener la seguridad de que el conjunto cumple matemáticamente lo que se quiere diseñar.

Las gráficas obtenidas de convergencia XY obtenidas en el presente proyecto son similares a las que se observan en el capítulo 3 de la referencia [14], lo que refuerza la confianza en el modelo desarrollado.

La estrategia de control elegida (sección 13.2.1), que no se ha extraído de ningún documento en particular, ha demostrado ser lo suficientemente eficiente para el tipo de misión encomendada.

El empleo de control PID único para tres direcciones espaciales ha sido efectivo al multiplicar su salida vectorial por el tensor de inercia. Aunque se ha sintonizado el control PID para un sistema de ejes cuerpo principales de inercia, ha demostrado que para otros ejes sigue siendo igual de eficiente si se mantiene el  $z$  principal de inercia, y siguen siendo efectivos incluso cuando el eje  $z$  no es principal de inercia.

Se observa en la sección 14.5 que aunque el control consigue su objetivo, cuando se desequilibra el eje  $z$  (deja de ser principal de inercia), es mucho más costoso en energía y en duración. Puede compararse a la importancia del equilibrado dinámico de todas las partes rotativas de una máquina de precisión. Por muy robusta que sea una bancada, si la máquina giratoria (compresor, turbina...) no está equilibrada acabará por estropearse debido a las vibraciones.

Por último recordar que los modelos diseñados e interconectados en este

proyecto se encuentran en el anexo II.

### 15.3. Trabajos futuros

Como Simulink es un sistema de diseño modular, es muy fácil mejorar una función añadiendo componentes o modificando bloques.

Una mejora evidente sería permitir diferentes formas de satélite y por lo tanto el cálculo del área proyectada se debería modificar.

El satélite en este modelo es rígido, pero en la realidad puede tener paneles que se despliegan y modificar sus momentos de inercia. Habría que ampliar el modelo dinámico. En el modelo dinámico se podrían incluir amortiguadores (dampers) que permitieran un control más eficiente y ahorro de propulsante/electricidad.

Se puede aumentar el grado de detalle de las perturbaciones, e incluso añadir alguna como la influencia de la Luna o el achatamiento terrestre.

Se puede mejorar el interfaz de usuario creando uno para uno para introducir los datos de la simulación.

El control se puede autosintonizar de forma inteligente y no manualmente como es el caso de este proyecto.

Así mismo se pueden incluir consigna de velocidad de rotación.

Se pueden añadir otros tipos de órbita como parabólica o hiperbólica.

.





# Bibliografía

- [1] Tai L. Chow. *Introduction to electromagnetic theory: a modern perspective*. Jones & Bartlett Learning, 2006. 7.1.5, 12.4.1
- [2] Carlos López de Echezarreta Eduardo Zornoza Carlos Gómez-Calero. Student concurrent design tool (scdt): Attitude & orbit control subsystem (aocs). Technical report, ESAC European Space Astronomy Centre, 2011. 8.1
- [3] Ignacio Fernández Rico, Germán Torralbo. Concurrent design application. PDF, 2014. 2.2.1, 2.3
- [4] Sebastián Franchini. Presentaciones de máster: Space project: Diseño preliminar del subsistema del control de actitud. 8.1
- [5] [http://www.esa.int/SPECIALS/CDF/SEMQOF1P4HD\\_0.html](http://www.esa.int/SPECIALS/CDF/SEMQOF1P4HD_0.html). Esa-cdf-what is the cdf?, Julio 2012. 1.1
- [6] <http://www.idr.upm.es/instalaciones/instalaciones.html>. Idr/upm-instalaciones. 1.1
- [7] J.B. Kuipers. *Quaternions and rotation Sequences: a Primer with Applications to Orbits, Aerospace, and Virtual Reality*. Princeton University Press, 1999. 9.4.1
- [8] K.H. Ang G.C.Y. Chong Y. Li. *PID control system analysis, design, and technology*. University of Glasgow, 2005. (document), 13.1
- [9] Viorel Macdonald, Malcolm Badescu. *The international handbook of Space technology*. Springer Berlin Heidelberg, 2014. 1.2.1
- [10] Vincent L. Pisacane Robert C. Moore. *Fundamentals of Space Systems*. Oxford, 1994. (document), 7.1, 7.1, 7.2, 8.1, 10.2.2

- [11] Fermín Navarro. Proyecto zahori. Technical report, UPM. 4.1, 6.1, 8.1
- [12] Rafael Ramis Abril. *Mecánica Orbital*. EIAE publicaciones, 2007. 11.1.1
- [13] T. Sanz-Aránguez, P. Elices Concha. *Vehículos espaciales y misiles*. Publicaciones EIAE UPM, 1999. 7.1.2, 7.1.4, 7.1.5, 8.1, 9.2.3, 9.1, 9.2, 11.1.1, 11.1.3, 11.4.1, 12.2.1, 12.5.1
- [14] S. Fred Singer. *Torques and attitude sensors in Earth satellites*. Academic Press, 1964. 15.2
- [15] Paul A. Tipler. *Física para la ciencia y la tecnología*. Reverté, 2000. 10.1
- [16] Wiley J. Larson James R. Wertz. *Space Mission Analysis and Design*. Space Technology Library Microcosm Press. 4.1, 8.1, 12.3.1
- [17] N.B. Ziegler, J.G. Nichols. *Optimum Settings for automatic controllers*. ASME 64, 1942. 13.2.2

# Índice alfabético

- ángulos de Euler:, 115
- actitud, 123
- actuadores, 26, 27, 58, 61, 79, 80, 190, 203
- ADCS, 24, 25, 45
- anomalía
  - excéntrica, 127, 130, 135, 136
  - verdadera, 128, 130–132, 136
- base de datos, 25, 27–31, 37, 39, 62, 63, 73, 75
- cell\_array, 84
- Concurrent Design Facility, 23
- control de actitud, 165, 170, 171, 189
- cuaternión, 115, 117, 123
- DSAD, 93
- eclipse, 129, 134
- Export, 35
- giróscopo, 26, 97, 98
- giróscopos, 26
- handles, 35, 49, 50, 58
- Import, 32
- Issue, 28, 30, 37
- magnetómetros, 26
- microcohetes, 26, 79
- Microsoft Access, 27, 28, 32
- momento
  - cero, 80
  - continuo, 80
- MySQL, 28, 31, 32
- Nan, 50
- ODBC, 27, 29, 31, 32
- orientación, 26, 27
- par
  - magnético, 57
- perturbación
  - aerodinámica, 47
  - campo magnético, 46
  - gradiente de gravedad, 48, 155
  - magnética, 153
  - solar, 46, 151
- perturbaciones, 45, 49, 58, 147, 191, 193, 198, 200, 201
- PID, 171–173, 176
- pre-fase A, 24, 25
- prefase A, 109
- primer punto de Aries, 109
- propulsores, 61
- Querybuilder, 31
- ruedas, 61
  - inercia, 55, 56, 79
  - reacción, 56

saturación, 79  
sensor, 91, 93–97  
Simulink, 25, 120, 125, 134, 148  
sistema  
    órbita, 110  
    cuerpo, 111  
    geocéntrico, 109, 110, 115  
    inercial, 109  
    trayectoria, 110  
spin, 26  
version, 28, 30

# Anexo I



## Programas para la conexión a DB

```

function [s1 s2] = ImportDB_3()
%Con esta funcion traemos de la base de datos las variables.
%Devuelve dos estructuras,
% s1 con todos los datos y
% s2 con la misma informacion, pero separadas
%por 'input' y 'output' con los valores de las variables, Issue y version
%De este modo para pasar todas las variables de un programa a otro, solo
%hay que pasar la primera estructura (s1). Cuando se vaya a grabar en la base de
%datos, solo se pasa la rama 'output' de la estructura s2.
%

%quitar estas instrucciones cuando hay conexion de verdad
%load('datos_satelite.mat','s1');
%return

cn = database('IDR','','');

%Provisional mientras me dan las variables
%curs = exec(cn,'select * from var_record where varIssue = 17');

curs = exec(cn,'select * from andrea');
curs = fetch(curs);

datos = curs.Data;

[r c] = size(datos);
close(curs);
close(cn);

for i=1:r
    %input /output
    tipo = datos{i,2};
    %Provisionelmante

    variable = datos{i,4} ;
    valor = str2num(datos{i,5});
    Issue = datos{i,6};
    Version = datos{i,7};

    %variable = datos{i,3} ;
    %valor = str2num(datos{i,4});

    s1.(variable) = valor;
    %
    s2.(tipo).(variable).Valor = valor ;
    s2.(tipo).(variable).Issue = Issue ;
    s2.(tipo).(variable).Version = Version;

    %comando1 = [e1 '.' variable ' = ' valor ';''];
    %comando2 = [e2 '.' tipo '.' variable ' = ' valor ';''];
    %evalin('caller',comando1);
    %evalin('base',comando2);
end

%Variables a la espera de asignación por el SE
tipo = 'input';
variable = 'HeightCube';
valor = .6;
Issue = 1;

```



```
Version = 0;
s1.(variable) = valor;
s2.(tipo).(variable).Valor = valor ;
s2.(tipo).(variable).Issue = Issue ;
s2.(tipo).(variable).Version = Version;
```

```
variable = 'SheetsNumber';
valor = 4;
s1.(variable) = valor;
s2.(tipo).(variable).Valor = valor ;
s2.(tipo).(variable).Issue = Issue ;
s2.(tipo).(variable).Version = Version;
```

```
%Guardo los datos para cuando vaya a ejecutar sin conexion a la base de
%datos.
```

```
save('datos_satelite.mat','s1');
```

```
end
```

```

function [ salida ] = Grabar_BD( p, sc )
%Con esta función comprobamos los datos recibidos de la base de datos con
%los actuales. Si los de tipo input son iguales, grabaremos los output si
%alguno se ha modificado.
% sc es la estructura con los datos leídos al principio

%Volvemos a leer la base de datos
[s scnew] = ImportDB_3();

%Variables de entrada nombres y numero
VariablesIn = fieldnames(sc.input);
nvin = size(VariablesIn,1);

%Comprobamos Issue, Version y valor de las variables Input leídas de la base
%de datos antes y ahora

%Indicador de error
cerror = 0;

for i = 1:nvin
    nom = VariablesIn{i,1};

    if sc.input.(nom).Issue ~= scnew.input.(nom).Issue
        cadena = [nom ' Issues ' num2str(sc.input.(nom).Issue) ' y ' num2str(scnew.
input.(nom).Issue)];
        cerror = 1;
        break;
    end

    if sc.input.(nom).Version ~= scnew.input.(nom).Version
        cadena = [nom ' Versions ' num2str(sc.input.(nom).Version) ' y ' num2str
(scnew.input.(nom).Version)];
        cerror = 2;
        break;
    end

    if sc.input.(nom).Valor ~= scnew.input.(nom).Valor
        cadena = [nom ' Values ' num2str(sc.input.(nom).Valor) ' y ' num2str(scnew.
input.(nom).Valor)];
        cerror = 3;
        break;
    end

end

if cerror ~= 0
    msgbox(cadena,'No se puede grabar','error')
    return
end

%Ahora comprobamos que el usuario no haya cambiado el valor a una variable
%input al hacer sus experimentos

for i = 1:nvin
    %Para cada variable input

```

```

%Busco el nombre de la variable en el programa
nomBD = VariablesIn{i,1};
nomP = NombreV(nomBD);

if strcmp(nomP,'dummy')
    %Variable no cambiable por el usuario (no traducida)
    continue
end

%y comparo sus valores
if sc.input.(nomBD).Valor ~= p.(nomP)
    error = 4;
    cadena = ['Variable ' nomBD ' / ' nomP ' ha cambiado'];
    break
end
end

if error ~= 0
    msgbox(cadena,'No se puede grabar','error')
    return
end

%Ahora vamos a comparar las variables de salida, y si alguna ha cambiado la
%grabamos

VariablesOut = fieldnames(sc.output);
nvout = size(VariablesOut,1);

cn = database('IDR','','');

%Leemos los campos de la tabla total
colnames = columns(cn,','', 'var_record');

for i = 1:nvout
    %Busco el nombre de la variable en el programa
    nomBD = VariablesOut{i,1};
    nomP = NombreV(nomBD);

    if strcmp(nomP,'dummy')
        %Variable no calculada/modificada
        continue
    end

    Issue = scnew.output.(nomBD).Issue;
    Version = scnew.output.(nomBD).Version;

    if sc.output.(nomBD).Valor ~= p.(nomP)
        %Aquí grabamos la variable en la base de datos
        sqlstring = ['select * from var_record where varName = '' ' nomBD ' '' ...
            ' and varIssue = ' num2str(Issue) ' ...
            ' and varVersion = ' num2str(Version)];

        curs = exec(cn,sqlstring);
        curs = fetch(curs,1);
        resultado = curs.Data;

        close(curs);
    end
end

```

```
%copio el registro y cambio el valor y la versión
nuevo = resultado;
nuevo{4} = num2str(p.(nomP));
nuevo{6} = Version + 1;

%inserto el registro sin incluir el campo autonumerado
insert(cn, 'var_record', colnames(2:end),nuevo(2:end));
```

```
end
end
```

```
%Cierro la conexion a la base de Datos
close(cn);
```

```
end
```

```

function [ NombreP ] = NombreV( NombreBD )
%Con esta función se traduce el nombre de una variable en base de datos con
%el nombre utilizado en este programa

switch NombreBD
    case 'OrbitAltitudeCircular'
        NombreP = 'h';
    case 'MaximumDeviation'
        NombreP = 'theta_m';
    case 'Velocity'
        NombreP = 'v';
    case 'ResidualDipole'
        NombreP = 'D1';
    case 'Iz'
        NombreP = 'MIz';
    case 'Imin'
        NombreP = 'MIy';
    case 'SuperficieFrontal'
        NombreP = 'A';
    case 'Cd'
        NombreP = 'Cda';
    case 'GravityCenter'
        %NombreP = 'cg';
        NombreP = 'dummy';

    case 'PointingAccuracy'
        NombreP = 'theta_a';
    case 'Period'
        NombreP = 'P';
    case 'Burntime'
        NombreP = 'tb';
    case 'Isp'
        NombreP = 'Isp_v';

    case 'AttitudeControlWeight'
        NombreP = 'masa';
    otherwise
        %msgbox(['Variable ' NombreBD ' no modificable en el programa'],'Variable
OK','help');
        NombreP = 'dummy';
end

end

```



## Anexo II





Modelo global

```
%Programa para inicializar el modelo "Orbita_y_sol.mdl"

clc;
clear;
%Datos del SOL

mu_sol = 132712440000; %parametro gravitacional
ro_sol = 149598261; %radio de la orbita km
e_sol = 0.01671123; %excentricidad
PS = 4.6e-6; %Presion solar en Pa. Fs = 1358 w/m^2; c = 3e8m/s
%PS = Fs/c

%Datos de la tierra
mu_t = 398600; %parametro gravitacional
RT = 6378; % km

%Para el campo magnetico (Sistema Internacional)
B0 = 1e-7; %mu0/(4*pi)
m = 7.96e15; %momento dipolar de la tierra modulo
bi = 11*pi/180; %inclinacion del campo magnetico terrestre
wt = 2*pi/(24*3600); % rotacion de la tierra
m0 = m*[sin(bi) 0 cos(bi)]; %momento dipolar de la tierra vector

%comprobaciones orbita SOL
ns = sqrt(mu_sol/ro_sol^3);
T = 2*pi/ns;

%Datos de la orbita
a = 10000+RT ; %km
e = .3 ;

rp = a*(1-e);
if rp-RT < 20
    msgbox(['El perigeo esta a ' num2str(rp-RT) ' km'], 'Orbita muy baja', 'error');
end

nt = sqrt(mu_t/a^3);
T = 2*pi/nt;

msgbox(['El periodo de la orbita es ' num2str(T/60) ' minutos'], 'Periodo', 'help');

%Datos el satelite
cp = [.04 .03 -.02]; %Centro de Presion solar/aerodinamica
LWH = [.4 .6 .7] ; %Largo ancho y alto
CD = 2; %Coeficiente aerodinamico
qs = 0.6 ;%Reflexión solar

D = [1 .5 .7]; %Dipolo residual (A*m^2)

inercia = [40 0 0; 0 50 0; 0 0 71]; %Tensor de Inercia
omega_ini = [0.1 0.1 0.4] ;%Omega inicial
```

```
%Densidad de la atmosfera según altura
```

```
air_density = [1.2250e+000  3.8990e-002  1.7740e-002  8.2790e-003  3.9720e-003  1.9950e-003  1.0570e-003  ...
               5.8210e-004  3.2060e-004  1.7180e-004  8.7700e-005  4.1780e-005  1.9050e-005  8.3370e-006  ...
               3.3960e-006  1.3430e-006  5.2970e-007  9.6610e-008  2.4380e-008  8.4840e-009  3.8450e-009  ...
               2.0700e-009  1.2440e-009  5.4640e-010  2.7890e-010  7.2480e-011  2.4180e-011  9.1580e-012  ...
               3.7250e-012  1.5850e-012  6.9670e-013  1.4540e-013  3.6140e-014  1.1700e-014  5.2450e-015  ...
               3.0190e-015]; %kg/m^3
```

```
altitude = [ 0      25000      30000      35000      40000      45000 50000 55000 ...
             60000      65000      70000      75000      80000      85000 90000 95000 ...
             100000     110000     120000     130000     140000     150000 160000 180000 ...
             200000     250000     300000     350000     400000     450000 500000 600000 ...
             700000     800000     900000     1000000]; %m
```

```
%sim('orbita_Y_sol')
```

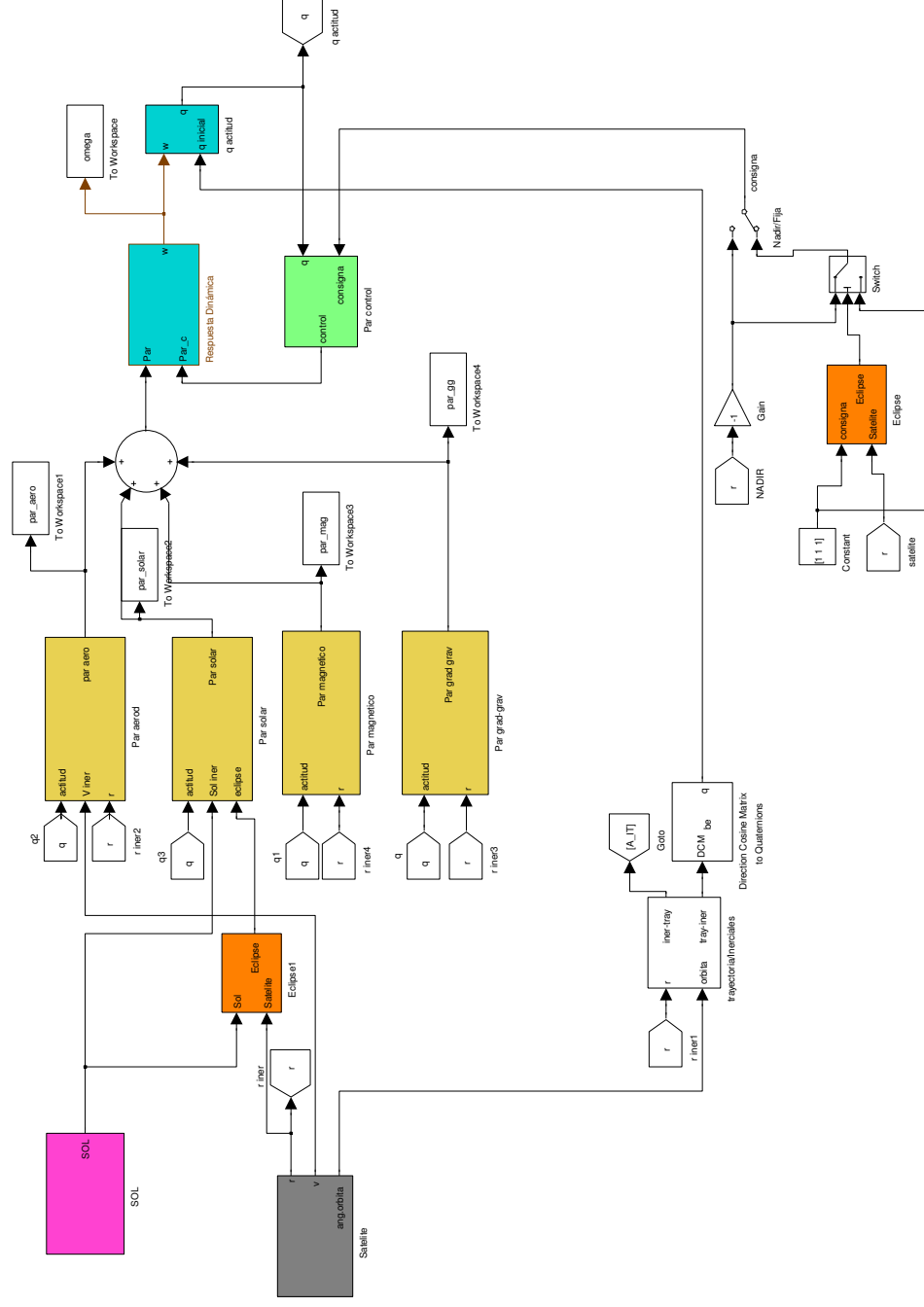
```
%graficas
```

```
%Control PID
```

```
KP = 5;
KI = 1.5;
KD = 2;
```

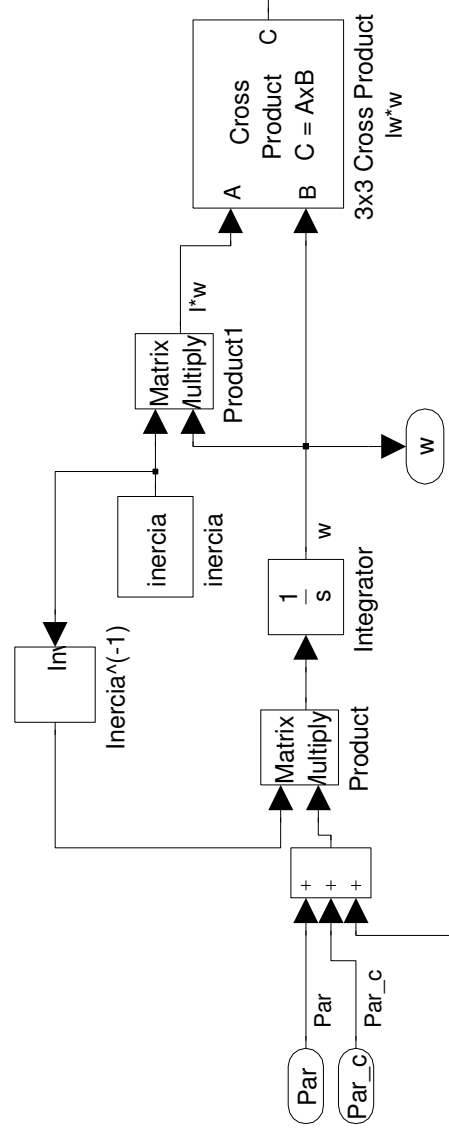
```
Nmax = 7; %Par máximo
```

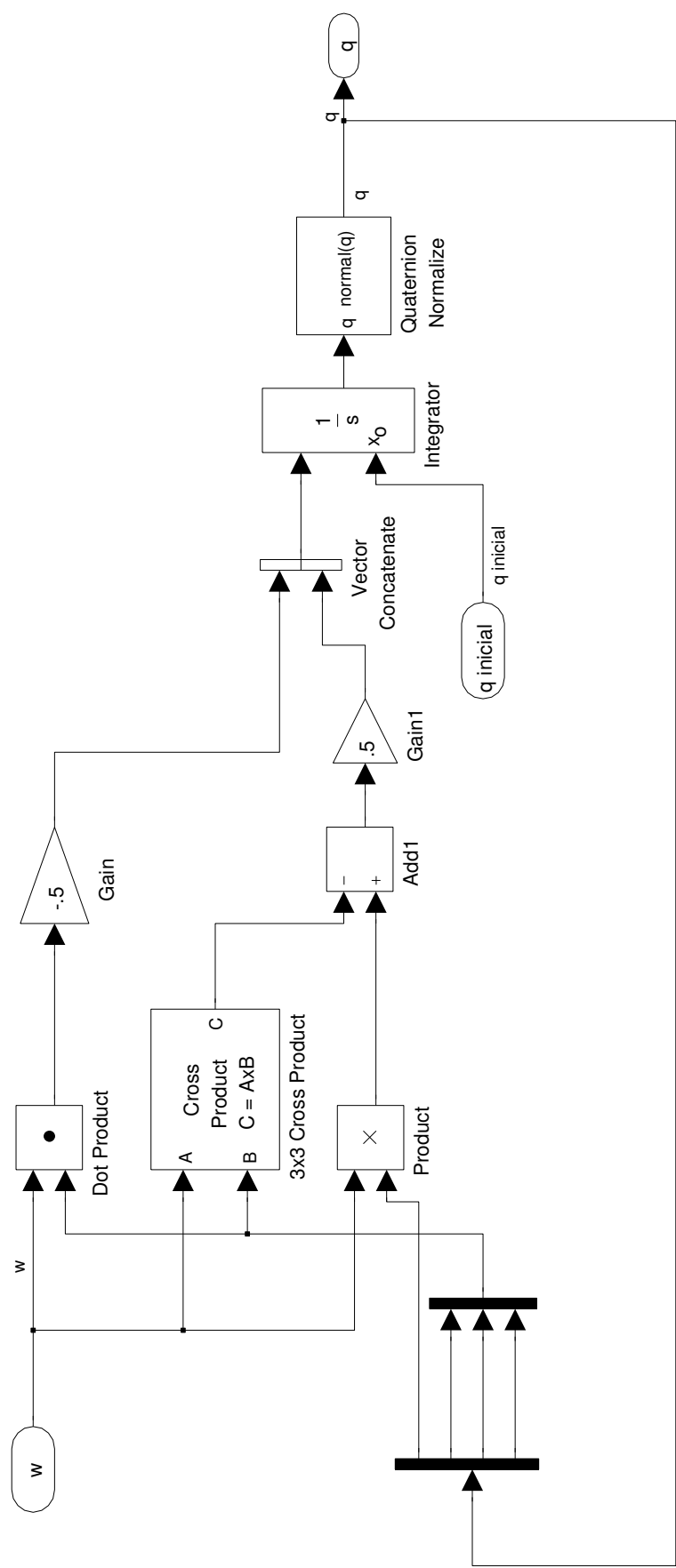
```
WMAX = 4.2 ;% control de maxima wz
```

**orbita\_Y\_sol**

C:\Users\Andrea\Dropbox\IDR\Simulink\orbita\_Y\_sol.mdl

## Respuesta dinámica y actitud

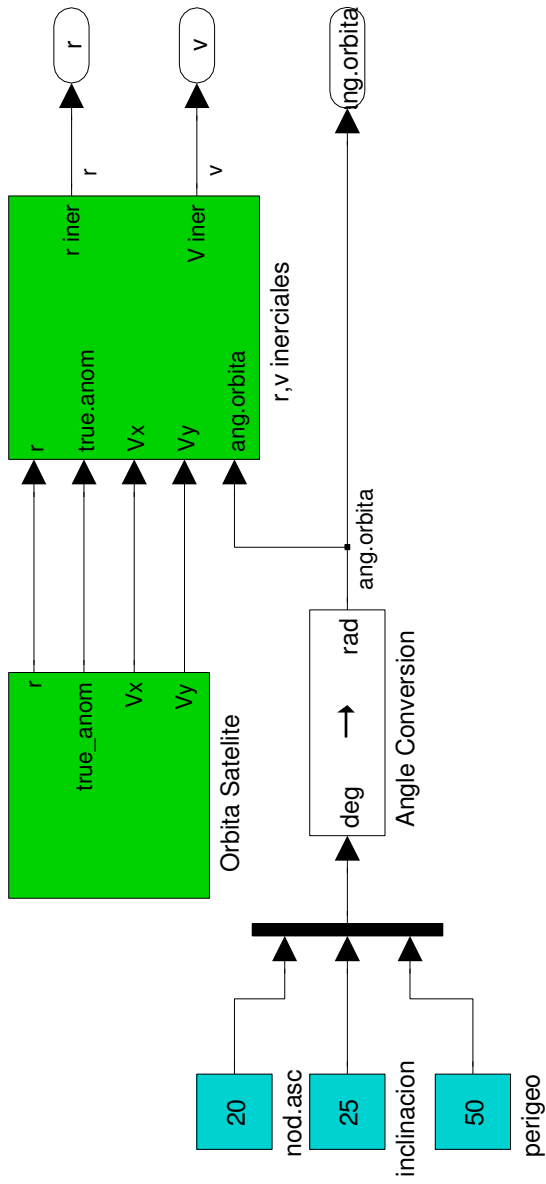




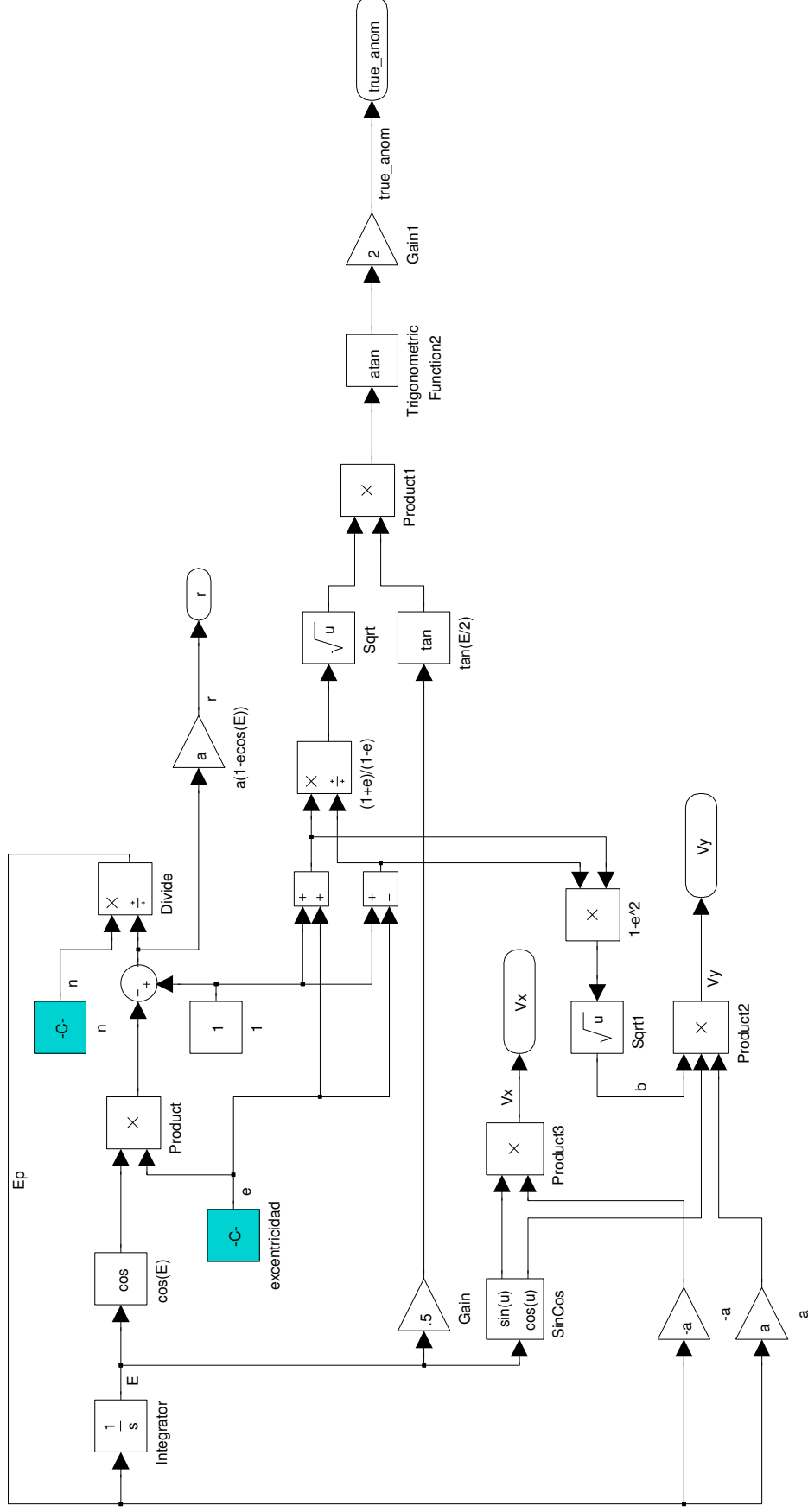




# Órbitas y eclipse

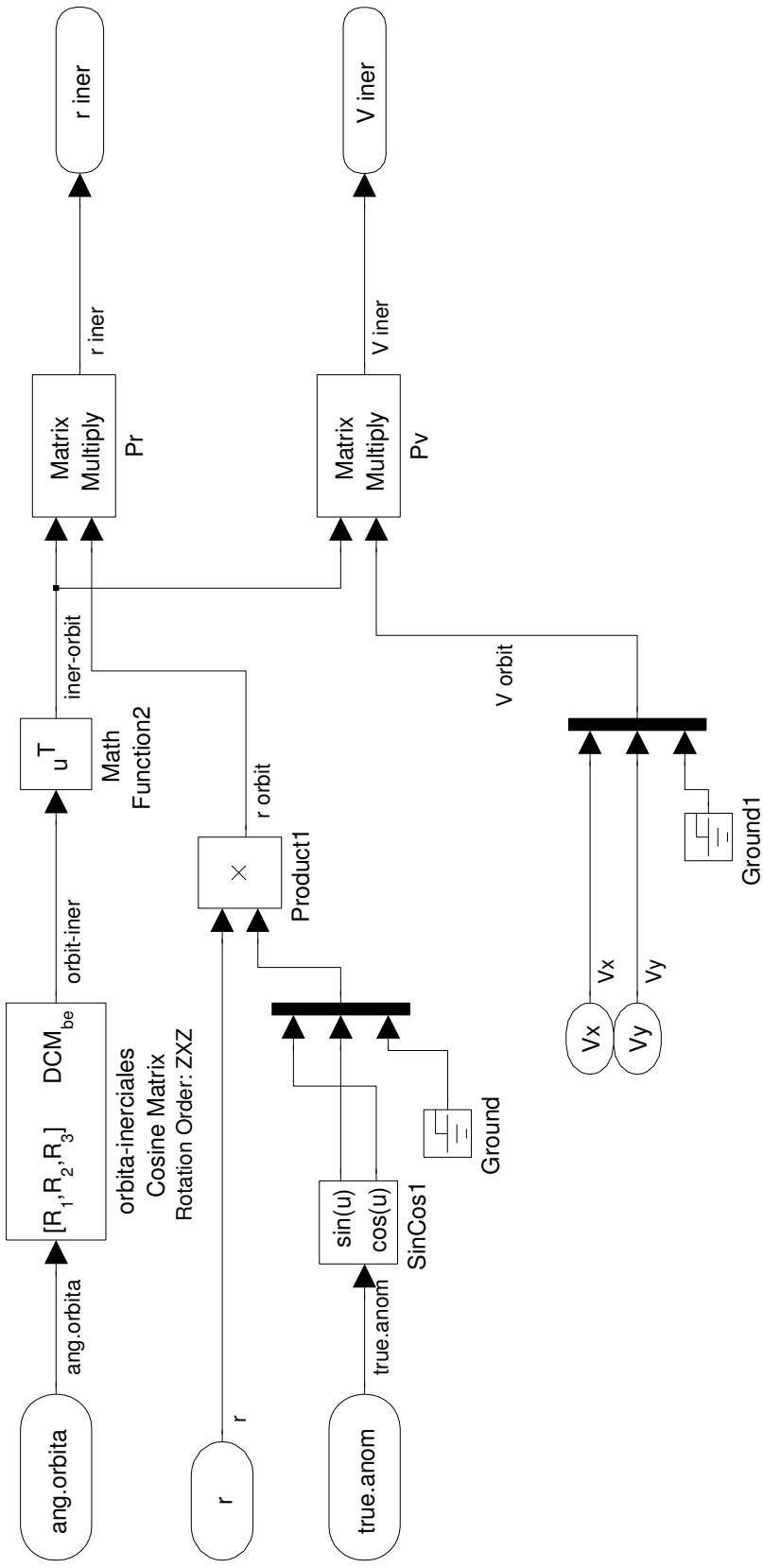


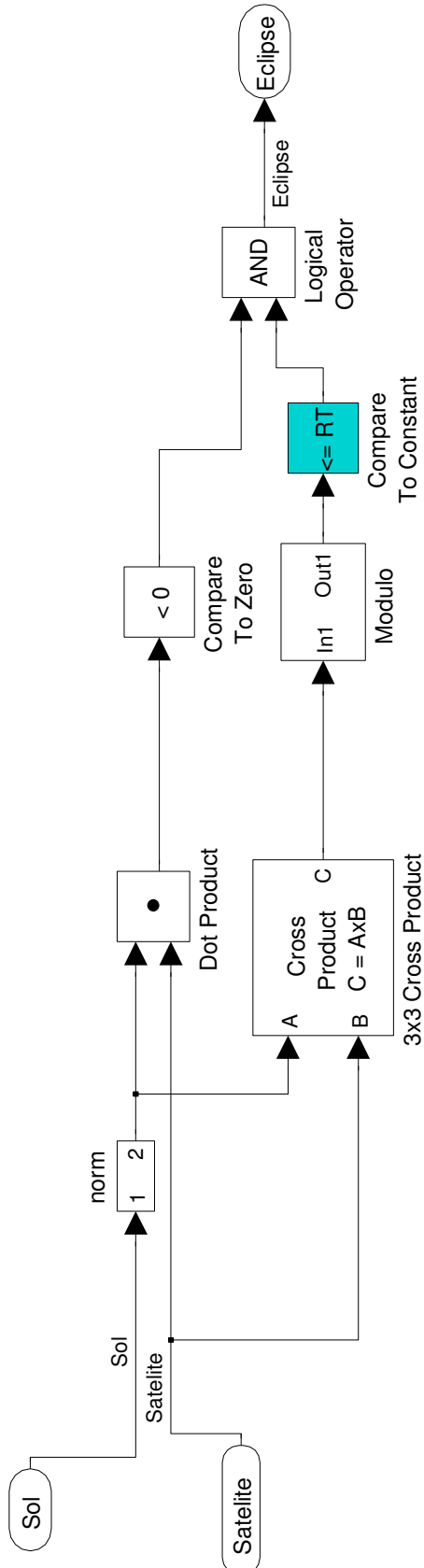
orbita\_Y\_sol/Satelite/Orbita Satellite

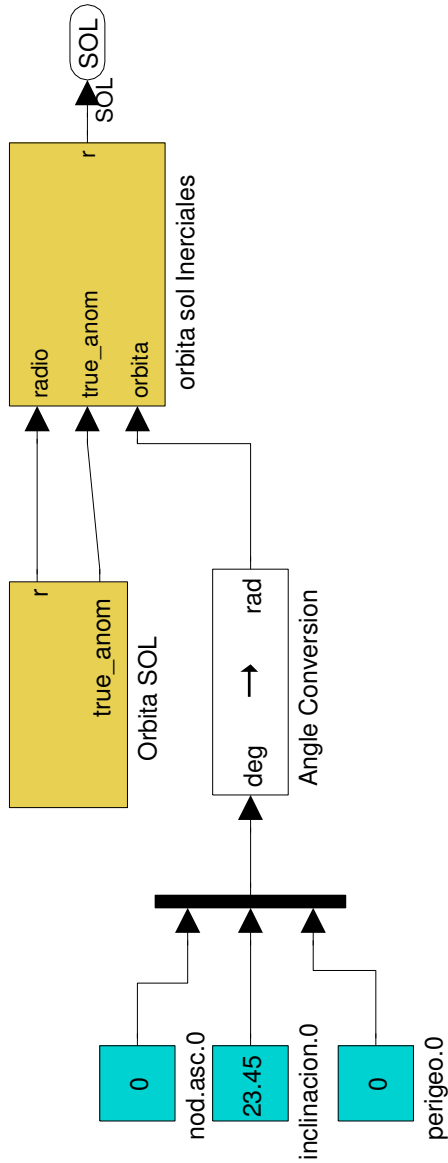


C:\Users\Andrea\Dropbox\IDR\Simulink\orbita\_Y\_sol.mdl

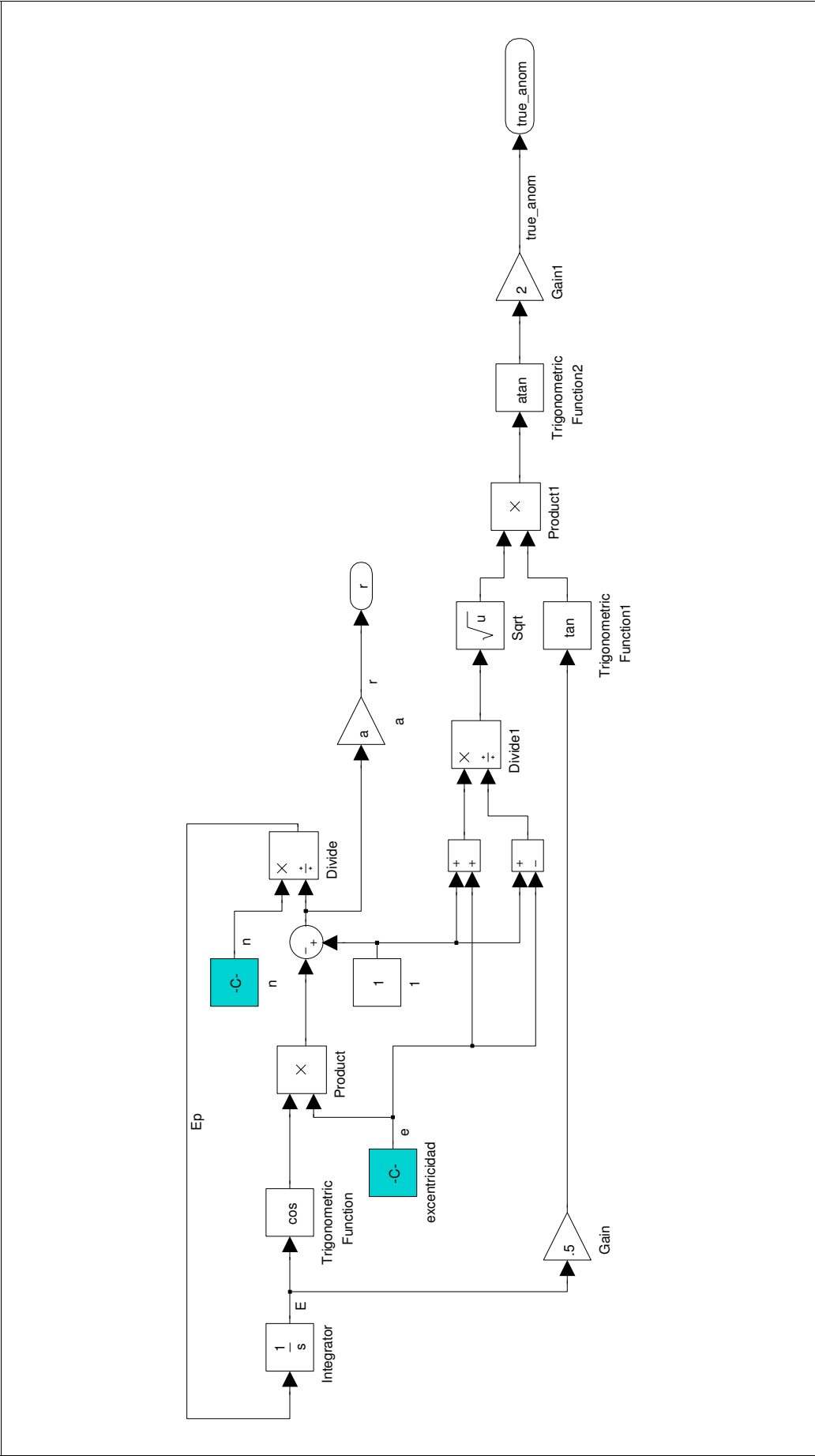
orbita\_Y\_sol/Satelite/r,v inerciales



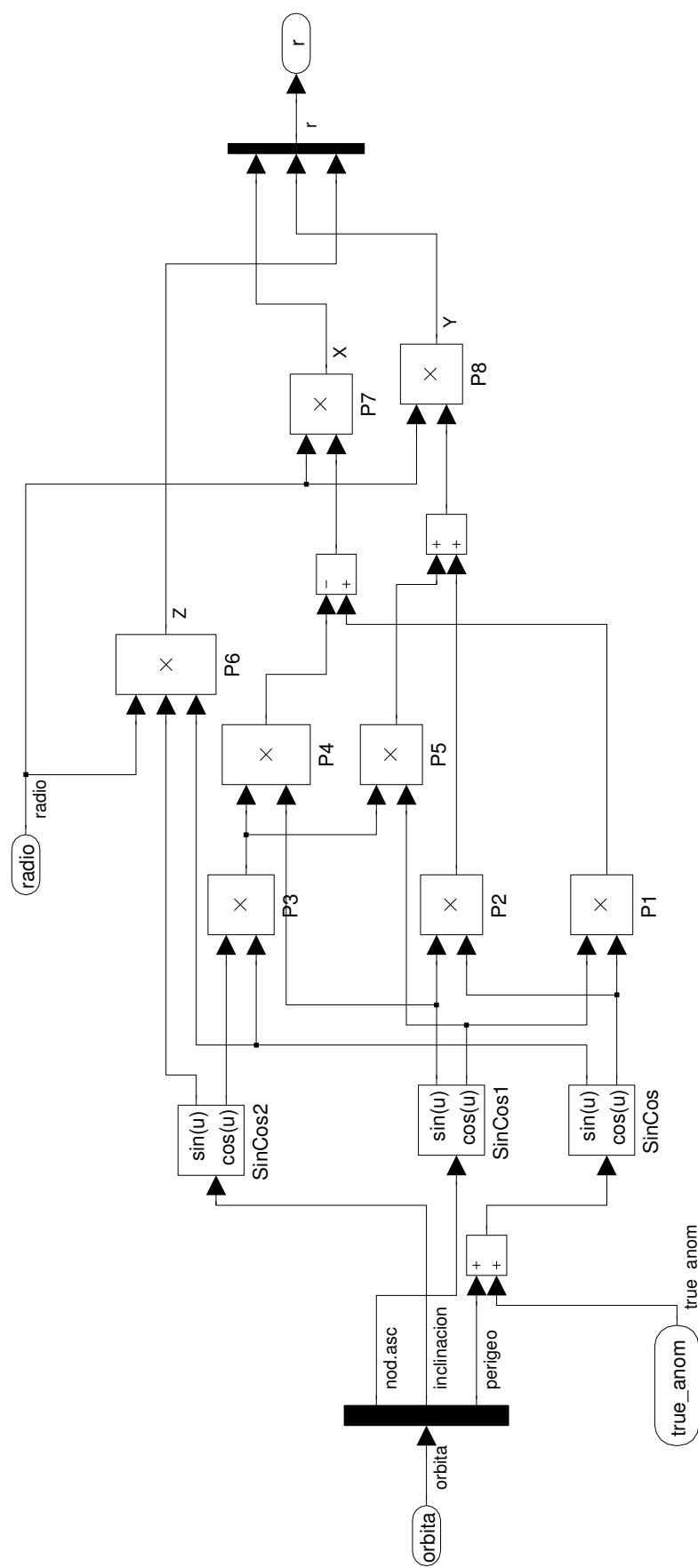




<b>orbita_Y_sol/SOL/Orbita SOL</b>

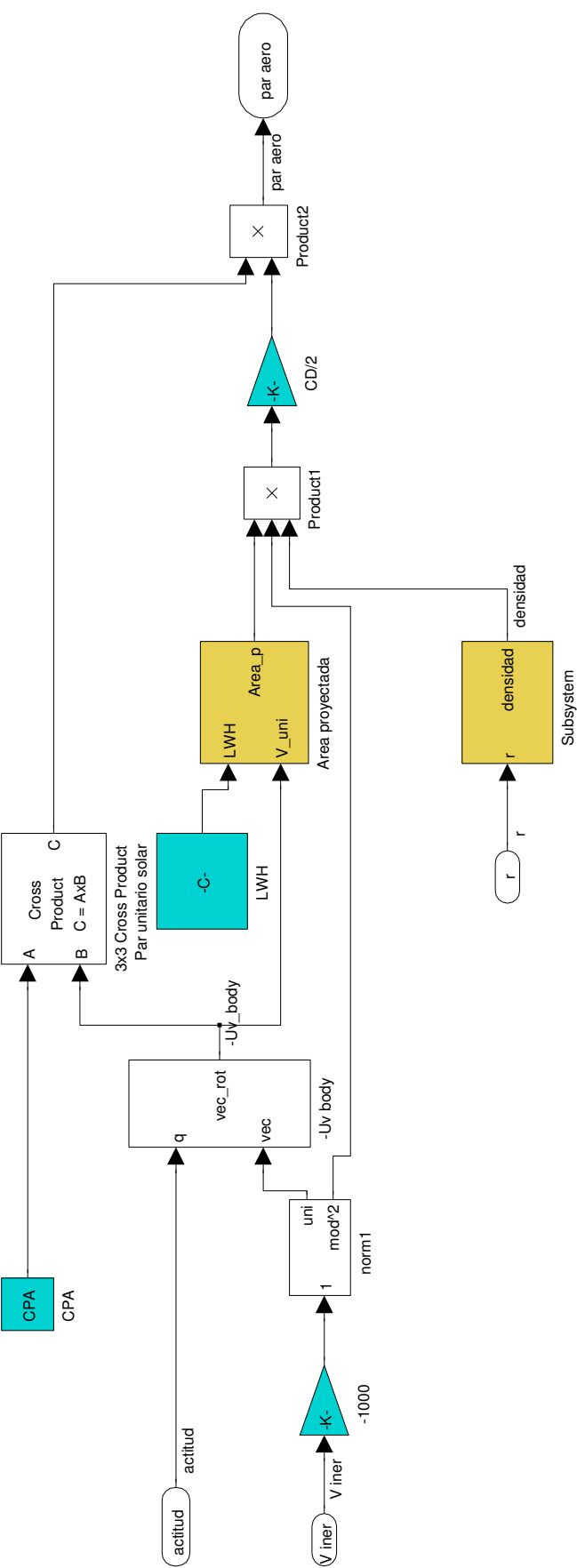


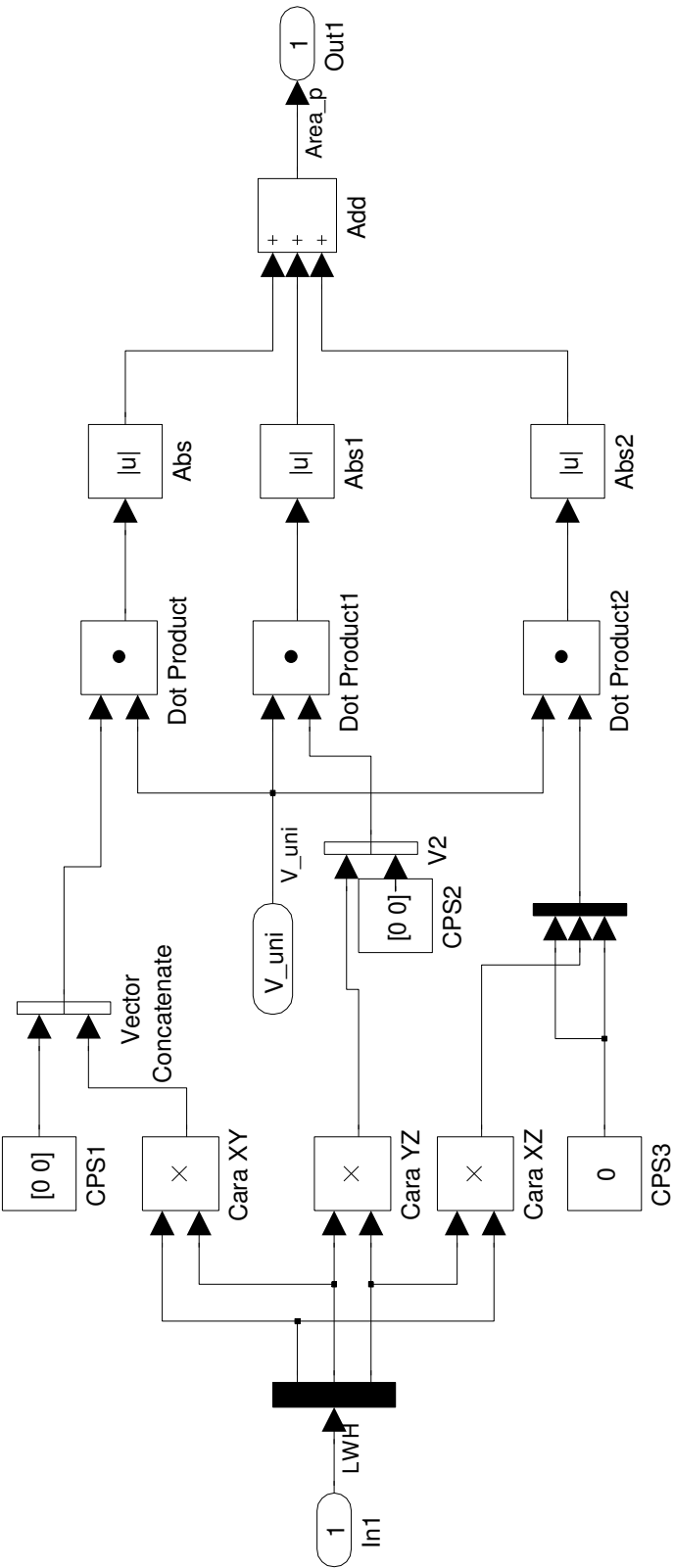
C:\Users\Andrea\Dropbox\IDR\Simulink\orbita\_Y\_sol.mdl

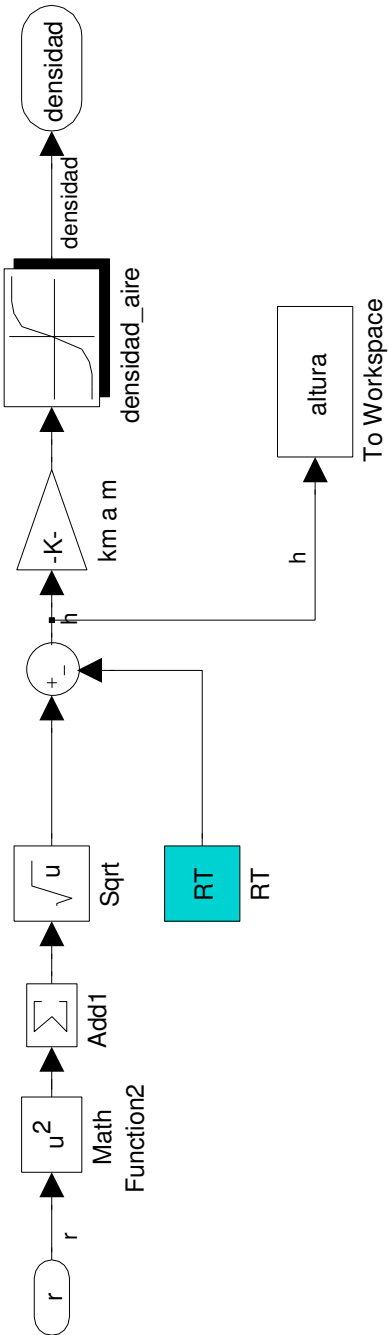


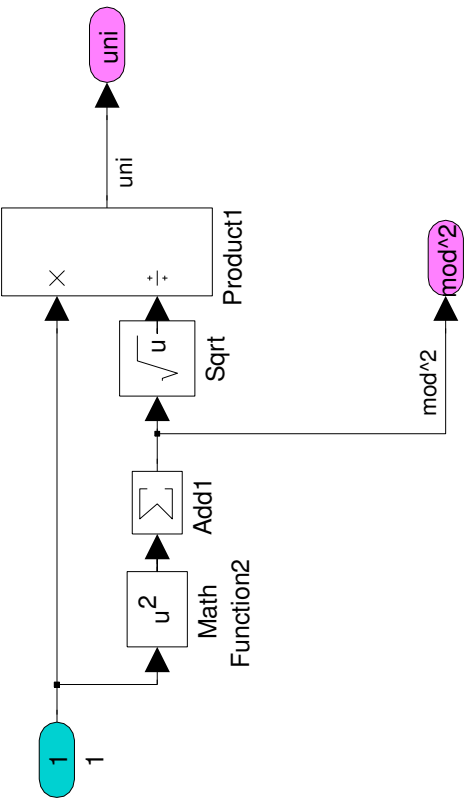


## Pares de perturbación

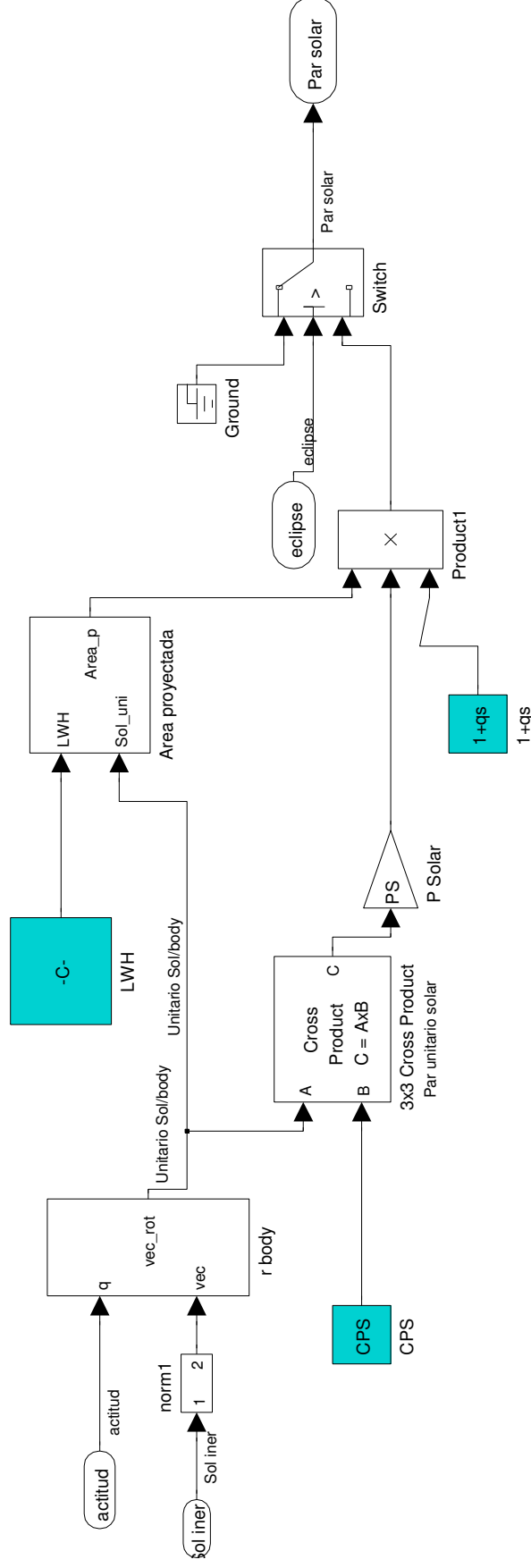




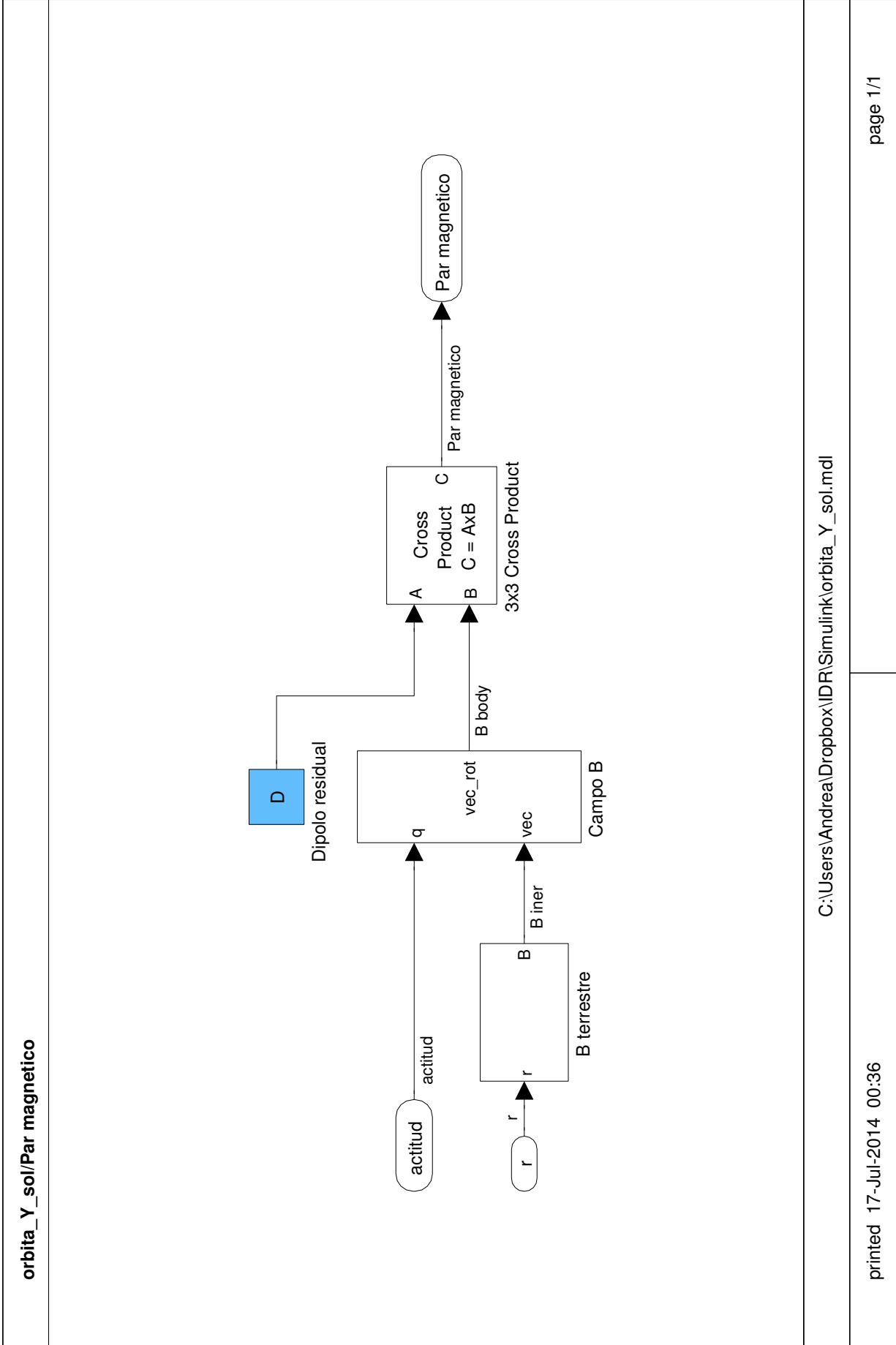




# orbita\_Y\_sol/Par solar



C:\Users\Andrea\Dropbox\IDR\Simulink\orbita\_Y\_sol.mdl

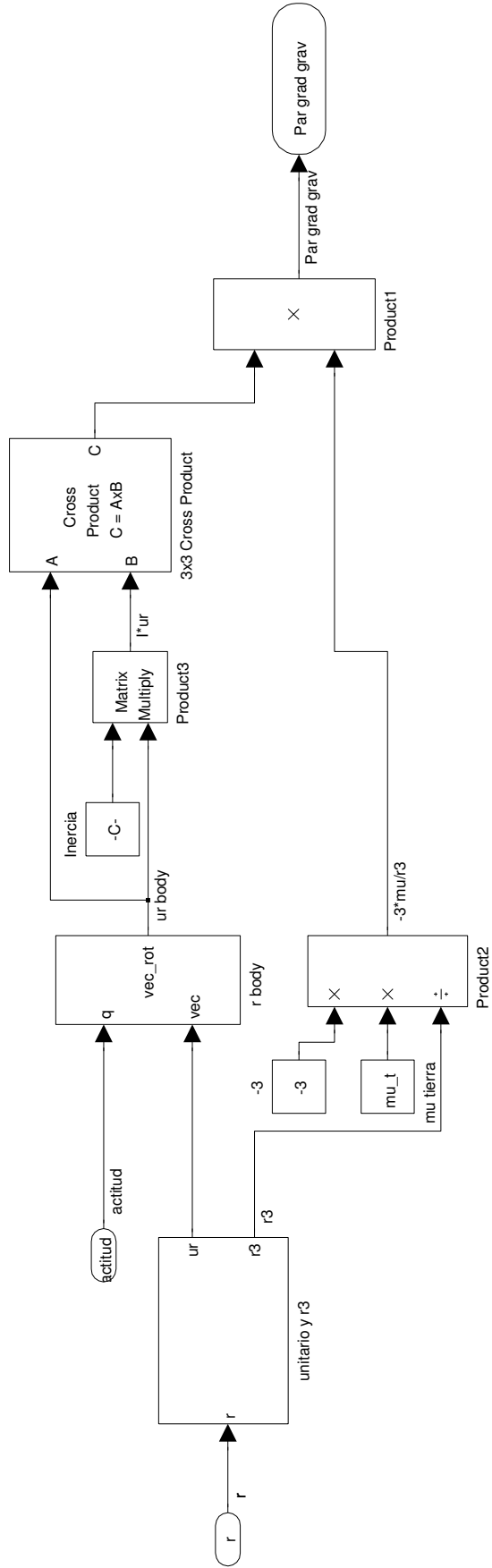


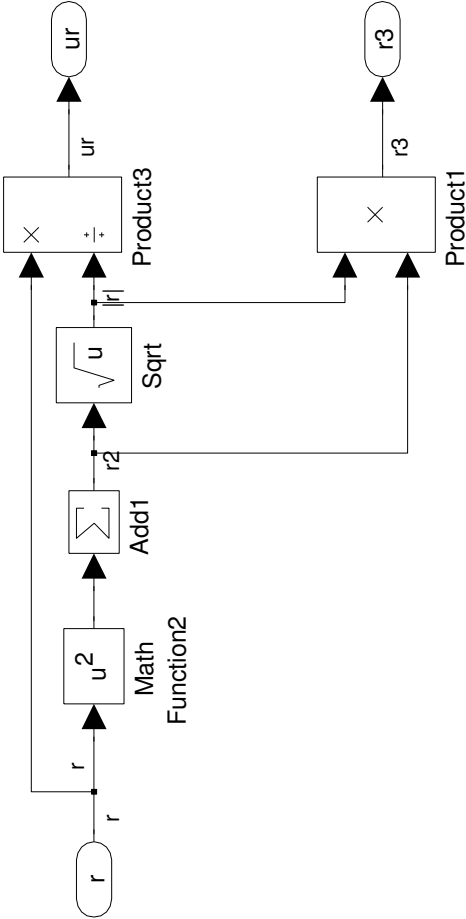
orbital\_Y\_sol/Par magnetico/B terrestre

printed 17-Jul-2014 00:36

page 1/1







## Control y consigna



