Satellite Dynamics and Control in a Quaternion Formulation (2nd edition)

Blanke, Mogens; Larsen, Martin Birkelund

Publication date:
2010

Document Version
Publisher's PDF, also known as Version of record

Satellite Dynamics and Control in a Quaternion Formulation

-

Lecture note for course 31365 Spacecraft Dynamics and Control at DTU

Mogens Blanke and Martin Birkelund Larsen

Automation and Control Group
DTU Electrical Engineering Department
Technical University of Denmark

Version 2f - September, 2010

Abstract

This lecture note treats modelling and attitude control design using a quaternion description of attitude for a rigid body in space. Dynamics and kinematics of a satellite is formulated as a non-linear model from Euler's moment equations and a description of kinematics using the attitude quaternion to represent rotation. A general linearised model is derived such that the user can specify an arbitrary point of operation in angular velocity and wheel angular momentum, specifying the a inertia matrix for a rigid satellite. A set of Simulink® models that simulate the satellite's nonlinear behaviour are described and a Matlab® [1] function is described that has been written to calculate the linear model in an arbitrary point of operation.

[1] Matlab and Simulink are trademarks of MathWorks Inc. USA.

Contents

1 Introduction **3**

2 Models in State-space Form **5**
- 2.0.1 Solution of an LTI state-space equation 6
- 2.0.2 Laplace transform of an LTI system 6
- 2.0.3 Stability of an LTI model 7
- 2.0.4 Dealing with LTI models in Matlab 7

2.1 Nonlinear models and linearization 8
- 2.1.1 Scalar case 8
- 2.1.2 Linearisation in scalar case - example 8
- 2.1.3 Vector case 9
- 2.1.4 Linearisation vector case - example 10

3 Satellite Equations of Motion **11**

3.1 Dynamics of satellite with reaction wheels 11

3.2 Kinematics 12

3.3 Reaction Wheels as Actuators 14
- 3.3.1 Wheel Aligned with Satellite Coordinate System . . . 14
- 3.3.2 Wheel Aligned in Other Direction 14

3.4 Actuator Dynamics - the Wheel Control Loop 15
- 3.4.1 Combined Model 15

3.5 Nonlinear model 15

4 Nonlinear Simulation Model **19**

5 Linear Model for Analysis **23**

5.1 Linear Satellite Model with Actuator Dynamics 26

6 Matlab Function to get Linearised Model **27**

6.1 Linear model without wheel dynamics 27

6.2 Linear Models with wheel dynamics 28

6.3 Linear model for sampled system 28

7 Attitude Control **29**
7.1 Reference in both angular rate and attitude 30
7.2 Reference in Attitude 32
7.3 Stability Analysis and Simple Controller Design 33

A Programme Listings and Contents CD **39**
A.1 linSatModNogg 40
A.2 romerEx 43
A.3 SATsimstart 46
A.4 RWAsim 46
A.5 SATsim 48

Chapter 1

Introduction

The purpose of this lecture note is to derive and provide the complete nonlinear model of a satellite with reaction wheels, using quaternion representation of attitude, and to derive the linearised equations of motion of such satellite for use in linear analysis and control design. The note extends and supplements the treatment of attitude models that are provided in section 4.8 and later in the book by Marcel J. Sidi [5], which is based on Euler angle description of attitude. Sidi analyse a number of feedback schemes in chapter 6, based on an Euler angle model and assuming small deviations from a reference. In chapter 7, he introduces attitude control based on quaternion error feedback. However, the complete quaternion based model was not derived by [5]. A linearised model was needed in the quaternion formulation as well.

When we wish to make an analysis of the quaternion feedback scheme similar to that done in chapter 6, a small signal model is needed for the satellite described with attitude represented as a quaternion.

This note is aimed at students at a first course in spacecraft dynamics and control, at the first year of the MSc level, where participants have different backgrounds. Therefore, this note starts with some prerequisites. In chapter 2, we define models of linear time invariant (LTI) systems as a set of coupled first-order differential equations, referred to as the state-space formulation in the automatic control literature, and analysis of stability properties are discussed. It is then shown how a continuous-time nonlinear system is linearised, using the Jacobian to obtain the state-space model in vector-matrix notation.

Chapter 3 considers the dynamics of a spacecraft with reaction wheels and derives a model of spacecraft attitude dynamics and kinematics in a quaternion representation. The dynamics associated with torque control of reaction wheels[1] are included in the model. Gravity gradient effects are not included in this version.

[1]The terms reaction wheel and momentum wheel are used as synonyms in this note.

Chapter 4 derives a linearised model for analysis in an arbitrary point of operation, i.e. angular momentum from reaction wheels and angular velocity vector.

Chapter 5 introduces ways to design and implement feedback control and discusses the issue of how to command reference pointing in various control configurations. Finally, stability analysis is demonstrated on an example.

The appendix comprises Matlab® code used in examples in the text.

A comprehensive introduction to rotation sequences and quaternions is given by Kuipers [4]. Some advanced topics in attitude control are presented by Wie [7]. A comprehensive coverage of satellite dynamics is also available in the now classical text by Huges [2].

Chapter 2

Models in State-space Form

Systems described in continuous-time by differential equations are often re-written to a set of first-order coupled differential equations because these are more easily solved. A differential equation in the variable $x(t)$ and input $u(t)$

$$\dddot{x}\left(t\right)+a_3\ddot{x}\left(t\right)+a_2\dot{x}\left(t\right)+a_1x\left(t\right)=b_1u\left(t\right), \tag{2.1}$$

can be re-written if we introduce

$$x_1=x\left(t\right), \quad x_2=\dot{x}\left(t\right), \quad x_3=\ddot{x}\left(t\right),$$

to

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}=\begin{bmatrix} x_2 \\ x_3 \\ -a_1x_1-a_2x_2-a_3x_3+b_1u \end{bmatrix}$$

In a matrix-vector notation, when we write in component form,

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}=\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_1 & -a_2 & -a_3 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}+\begin{bmatrix} 0 \\ 0 \\ b_1 \end{bmatrix}u$$

In condensed matrix-vector notation, and using $\dot{x}=\frac{dx}{dt}$,

$$\dot{\mathbf{x}}(t)=\mathbf{A}(t)\mathbf{x}+\mathbf{B}(t)\mathbf{u}.$$

If an output is included, we get the general form, referred to as a state-space model

$$\begin{aligned} \dot{\mathbf{x}}(t)&=\mathbf{A}(t)\mathbf{x}(t)+\mathbf{B}(t)\mathbf{u}(t), \\ \mathbf{y}(t)&=\mathbf{C}(t)\mathbf{x}(t)+\mathbf{D}(t)\mathbf{u}(t). \end{aligned} \tag{2.2}$$

A particular and important case is when the matrices $\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}$ are time-invariant. We then have a linear time-invariant (LTI) system:

$$\begin{aligned} \dot{\mathbf{x}}(t)&=\mathbf{A}\mathbf{x}(t)+\mathbf{B}\mathbf{u}(t), \\ \mathbf{y}(t)&=\mathbf{C}\mathbf{x}(t)+\mathbf{D}\mathbf{u}(t). \end{aligned} \tag{2.3}$$

The LTI system is used throughout classical control for analysis of various properties, most important stability, and for design of linear control systems.

2.0.1 Solution of an LTI state-space equation

The Laplace transform is a transform from the time domain to a domain in the complex frequency s. The Laplace transform $\mathcal{L}$ of an LTI system has the following properties,

$$\begin{aligned} \mathcal{L}(x(t)) &= x(s), \\ \mathcal{L}(\dot{x}(t)) &= sx(s) - x(t_0). \end{aligned} \tag{2.4}$$

The final value theorem Eq. (2.5) shows how one finds the steady state value of a bounded function $x(t)$, from its Laplace transform $x(s)$ as:

$$\lim_{t\to\infty}(x(t)) = \lim_{s\to 0}(sx(s)). \tag{2.5}$$

As an example, let $x(t)$ be a unit step at time zero,

$$x(t) = \begin{cases} 1 & t \geq 0 \\ 0 & t < 0 \end{cases} \Rightarrow \mathcal{L}(x(t)) = \frac{1}{s}. \tag{2.6}$$

For a vector, $\mathbf{x}(t)$ and a time-invariant matrix $\mathbf{A}$,

$$\begin{aligned} \mathcal{L}(\mathbf{x}(t)) &= \mathbf{x}(s), \\ \mathcal{L}(\mathbf{A}\mathbf{x}(t)) &= \mathbf{A}\mathbf{x}(s). \end{aligned} \tag{2.7}$$

Rigorous treatments of the Laplace transform is available in most introductory texts in automatic control and signal processing.

2.0.2 Laplace transform of an LTI system

Applying the Laplace transform to the LTI state-space model, Eq. (2.3), gives:

$$\begin{aligned} s\mathbf{x}(s) &= \mathbf{A}\mathbf{x}(s) + \mathbf{B}\mathbf{u}(s) + \mathbf{x}(t_0) \\ \mathbf{y}(s) &= \mathbf{C}\mathbf{x}(s) + \mathbf{D}\mathbf{u}(s). \end{aligned} \tag{2.8}$$

Let $\mathbf{I} \in \mathbb{R}^{n\times n}$, be a diagonal unity matrix of dimension $n\times n$, this is rewritten as follows:

$$\begin{aligned} s\mathbf{I}\mathbf{x}(s) &= \mathbf{A}\mathbf{x}(s) + \mathbf{B}\mathbf{u}(s) + \mathbf{x}(t_0) \\ &\Leftrightarrow (s\mathbf{I} - \mathbf{A})\,\mathbf{x}(s) = \mathbf{B}\mathbf{u}(s) + \mathbf{x}(t_0) \\ &\Leftrightarrow \mathbf{x}(s) = (s\mathbf{I} - \mathbf{A})^{-1}\,(\mathbf{B}\mathbf{u}(s) + \mathbf{x}(t_0)) \\ \mathbf{y}(s) &= \left(\mathbf{C}\,(s\mathbf{I} - \mathbf{A})^{-1}\,\mathbf{B} + \mathbf{D}\right)\mathbf{u}(s) + \mathbf{C}\,(s\mathbf{I} - \mathbf{A})^{-1}\,\mathbf{B}\mathbf{x}(t_0). \end{aligned} \tag{2.9}$$

The transfer function $\mathbf{H}_{\mathbf{yu}}(s)$ from input $\mathbf{u}(s)$ to output $\mathbf{y}(s)$ is

$$\mathbf{y}(s) = \left(\mathbf{C}\,(s\mathbf{I} - \mathbf{A})^{-1}\,\mathbf{B} + \mathbf{D}\right)\mathbf{u}(s) \equiv \mathbf{H}_{\mathbf{yu}}(s)\mathbf{u}(s), \tag{2.10}$$

where $\mathbf{H_{yu}}(s)$ has k rows (same as number of elements in $\mathbf{y}$) and m columns (same as number of inputs), $\mathbf{H_{yu}}(s) \in \mathbb{C}^{k \times m}$. Each element in $\mathbf{H_{yu}}(s)$ is a single-input single-output (SISO) transfer function from a particular input to a particular output.

Since $\mathbf{H_{yu}}(s) = \mathbf{C}\left(s\mathbf{I} - \mathbf{A}\right)^{-1}\mathbf{B} + \mathbf{D}$, all transfer functions in $\mathbf{H_{yu}}(s)$ have the same denominator, namely the polynomial $\det\left(s\mathbf{I} - \mathbf{A}\right)$.The numerators will in general be different from each other.

2.0.3 Stability of an LTI model

The stability of a transfer function is determined by the roots of its denominator, also referred to as the poles of the system. These poles are the roots of the equation

$$\det\left(s\mathbf{I} - \mathbf{A}\right) = 0, \tag{2.11}$$

which are also the eigenvalues of the matrix $\mathbf{A}$.

All eigenvalues of $\mathbf{A}$ must have a strictly negative real part for the system to be asymptotically stable. Eigenvalues with zero real and zero imaginary parts indicate pure integrations in the system. Pairs of eigenvalues symmetric on the positive and negative imaginary axis indicate that an undamped oscillating mode may exist in the system. Eigenvalues with positive real part show that the linear system is unstable.

Eigenvalues of a system in state-space form are generally found using numerical methods. Analytic solutions may be achieved in special cases.

2.0.4 Dealing with LTI models in Matlab

Matlab has commands dedicated to show features of LTI systems.

One need first to define a LTI system in state space form for Matlab:
`sys = ss(A,B,C,D)`
`[Wn,z,p] = damp(sys)` returns vectors $\mathbf{W}n$, $\mathbf{z}$, $\mathbf{p}$ containing the natural frequencies, damping factors and poles of the LTI model sys.
`p = pole(sys)` computes the poles $\mathbf{p}$ of the LTI model sys

`pzmap(sys)` computes the poles and (transmission) zeros of the LTI model sys and plots them in the complex plane. The poles are plotted as x's and the zeros are plotted as o's. Transmission zeros are the complex frequencies $s = z$ at which the rank of $\mathbf{H}_{yu}(z)$ is less than the normal rank of $\mathbf{H}_{yu}(s)$. These are outside the scope of this note and will not be treated further in this context [6, (section 4.5)].

The Matlab LTI toolbox documentation should be consulted for further details.

2.1 Nonlinear models and linearization

2.1.1 Scalar case

Let a system be described by the nonlinear scalar function

$$\begin{aligned} \dot{x} &= g(\dot{x}, u), \\ y &= h(x, y), \end{aligned} \tag{2.12}$$

where $x \in \mathbb{R}^1, u \in \mathbb{R}^1, y \in \mathbb{R}^1$. We wish to obtain a model that describes operation around a trajectory $\dot{\bar{x}}, \bar{x}, \bar{u}$. Introduce deviations from the trajectory by

$$\begin{aligned} \tilde{x} &= x - \bar{x}, \\ \tilde{u} &= u - \bar{u}, \\ \tilde{y} &= y - \bar{y}. \end{aligned} \tag{2.13}$$

Then a Taylor expansion of Eq. (2.12) gives

$$\begin{aligned} \dot{\tilde{x}} + \dot{\bar{x}} &= g(\bar{x}, \bar{u}) + \tfrac{\partial g(\bar{x},\bar{u})}{\partial \mathbf{x}} \tilde{x} + \tfrac{\partial g(\bar{x},\bar{u})}{\partial \mathbf{u}} \tilde{u} + ... \\ \tilde{y} + \bar{y} &= h(\bar{x}, \bar{u}) + \tfrac{\partial h(\bar{x},\bar{u})}{\partial x} \tilde{x} + \tfrac{\partial h(\bar{x},\bar{u})}{\partial u} \tilde{u} + ... \end{aligned} \tag{2.14}$$

Since $\dot{\bar{x}} = g(\bar{x}, \bar{u})$ and $\bar{y} = h(\bar{x}, \bar{u})$ by definition of the "bar" variables, and truncation to first order of the Taylor series gives

$$\begin{aligned} \dot{\tilde{x}} &= \tfrac{\partial g(\bar{x},\bar{u})}{\partial x} \tilde{x} + \tfrac{\partial g(\bar{x},\bar{u})}{\partial u} \tilde{u}, \\ \tilde{y} &= \tfrac{\partial h(\bar{x},\bar{u})}{\partial x} \tilde{x} + \tfrac{\partial h(\bar{x},\bar{u})}{\partial u} \tilde{u}. \end{aligned} \tag{2.15}$$

This expression is linear in the tilted variables, i.e., we have obtained a linear model. The parameters can be functions of the trajectory around which we linearise.

2.1.2 Linearisation in scalar case - example

Let a nonlinear system be

$$\begin{aligned} \dot{x} &= a_0 + a_1 x + a_2 x^2 + b_1 u + b_2 x u, \\ y &= c_1 x + c_3 x^3. \end{aligned} \tag{2.16}$$

The model linearised about the $(\bar{x}, \bar{u})$ trajectory is

$$\begin{aligned} \dot{\tilde{x}} &= (a_1 + 2a_2\bar{x} + b_2\bar{u})\,\tilde{x} + (b_2\bar{x})\,\tilde{u}, \\ \tilde{y} &= \left(c_1 + 3c_3\bar{x}^2\right)\tilde{x}. \end{aligned} \tag{2.17}$$

A special case is that $(\bar{x}, \bar{u})$ is time-invariant. In this case we obtain a linear time-invariant system (LTI).

2.1.3 Vector case

Let a system be described by the nonlinear vector equation

$$\begin{aligned}\dot{\mathbf{x}}(t)&= \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t))\\ \mathbf{y}(t) &= \mathbf{h}(\mathbf{x}(t), \mathbf{y}(t))\end{aligned} \tag{2.18}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state vector, $\mathbf{u} \in \mathbb{R}^m$ the control input, $\mathbf{y} \in \mathbb{R}^k$the output. We wish to obtain a model that describes operation around a trajectory defined by $\dot{\bar{\mathbf{x}}}, \bar{\mathbf{x}}, \bar{\mathbf{u}}$. Introduce therefore the (small) deviation from the trajectory by

$$\begin{aligned}\tilde{\mathbf{x}} &= \mathbf{x} - \bar{\mathbf{x}}\\ \tilde{\mathbf{u}} &= \mathbf{u} - \bar{\mathbf{u}}\\ \tilde{\mathbf{y}} &= \mathbf{y} - \bar{\mathbf{y}}\end{aligned} \tag{2.19}$$

Taylor expansion of Eq. 2.18 gives Eq. 2.20, where the argument (t) has been omitted for brevity,

$$\begin{aligned}\dot{\bar{\mathbf{x}}} + \dot{\tilde{\mathbf{x}}} &= \mathbf{g}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}}) + \frac{\partial \mathbf{g}(\bar{\mathbf{x}},\bar{\mathbf{u}},\bar{\mathbf{d}})}{\partial \mathbf{x}}\tilde{\mathbf{x}} + \frac{\partial \mathbf{g}(\bar{\mathbf{x}},\bar{\mathbf{u}},\bar{\mathbf{d}})}{\partial \mathbf{u}}\tilde{\mathbf{u}} + ...\\ \tilde{\mathbf{y}} + \bar{\mathbf{y}} &= \mathbf{h}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}}) + \frac{\partial \mathbf{h}(\bar{\mathbf{x}},\bar{\mathbf{u}},\bar{\mathbf{d}})}{\partial \mathbf{x}}\tilde{\mathbf{x}} + \frac{\partial \mathbf{h}(\bar{\mathbf{x}},\bar{\mathbf{u}},\bar{\mathbf{d}})}{\partial \mathbf{u}}\tilde{\mathbf{u}} + ...\end{aligned} \tag{2.20}$$

Using that $\dot{\bar{\mathbf{x}}} = \mathbf{g}(\bar{\mathbf{x}}, \bar{\mathbf{u}})$, $\bar{\mathbf{y}} = \mathbf{h}(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ and truncating the Taylor series after the first order,

$$\begin{aligned}\dot{\tilde{\mathbf{x}}} &= \frac{\partial \mathbf{g}(\bar{\mathbf{x}},\bar{\mathbf{u}},\bar{\mathbf{d}})}{\partial \mathbf{x}}\tilde{\mathbf{x}} + \frac{\partial \mathbf{g}(\bar{\mathbf{x}},\bar{\mathbf{u}},\bar{\mathbf{d}})}{\partial \mathbf{u}}\tilde{\mathbf{u}}\\ \tilde{\mathbf{y}} &= \frac{\partial \mathbf{h}(\bar{\mathbf{x}},\bar{\mathbf{u}},\bar{\mathbf{d}})}{\partial \mathbf{x}}\tilde{\mathbf{x}} + \frac{\partial \mathbf{h}(\bar{\mathbf{x}},\bar{\mathbf{u}},\bar{\mathbf{d}})}{\partial \mathbf{u}}\tilde{\mathbf{u}}\end{aligned} \tag{2.21}$$

This defines a linear system

$$\begin{aligned}\dot{\tilde{\mathbf{x}}} &= \mathbf{A}\tilde{\mathbf{x}} + \mathbf{B}_u\tilde{\mathbf{u}}\\ \tilde{\mathbf{y}} &= \mathbf{C}\tilde{\mathbf{x}} + \mathbf{D}_u\tilde{\mathbf{u}}\end{aligned} \tag{2.22}$$

The matrices $\mathbf{A}, \mathbf{B}_u$, are defined by the derivatives of the vector valued function $\mathbf{g}$ with respect to vectors $\mathbf{x}$, and $\mathbf{u}$ (and $\mathbf{C}, \mathbf{D}_u$ equivalently by differentiation of $\mathbf{h}$):

$$\begin{aligned}\mathbf{A} &= \frac{\partial \mathbf{g}(\bar{\mathbf{x}},\bar{\mathbf{u}},\bar{\mathbf{d}})}{\partial \mathbf{x}} \qquad \mathbf{C} = \frac{\partial \mathbf{h}(\bar{\mathbf{x}},\bar{\mathbf{u}},\bar{\mathbf{d}})}{\partial \mathbf{x}}\\ \mathbf{B}_u &= \frac{\partial \mathbf{g}(\bar{\mathbf{x}},\bar{\mathbf{u}},\bar{\mathbf{d}})}{\partial \mathbf{u}} \quad \mathbf{D}_u = \frac{\partial \mathbf{h}(\bar{\mathbf{x}},\bar{\mathbf{u}},\bar{\mathbf{d}})}{\partial \mathbf{u}}\end{aligned} \tag{2.23}$$

The derivative of a vector with respect to a vector is referred to as the Jacobian,

$$\frac{\partial \mathbf{g}(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{d}})}{\partial \mathbf{u}} = \begin{pmatrix} \frac{\partial g_1}{\partial u_1} & \frac{\partial g_1}{\partial u_2} & \cdots & \frac{\partial g_1}{\partial u_m} \\ \frac{\partial g_2}{\partial u_1} & \frac{\partial g_2}{\partial u_2} & \cdots & \frac{\partial g_2}{\partial u_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_n}{\partial u_1} & \frac{\partial g_n}{\partial u_2} & \cdots & \frac{\partial g_n}{\partial u_m} \end{pmatrix} \tag{2.24}$$

is a matrix with n rows and m columns.

2.1.4 Linearisation vector case - example

Let a system be given by

$$\begin{aligned}\dot{\mathbf{x}} = \mathbf{g}(\mathbf{x},\mathbf{u}) &= \begin{bmatrix} a_0 + a_1x_1x_2 + a_2x_1^2 + b_1u_1 + b_2x_2u_1 \\ a_3 + a_4x_1^2x_2^2 + b_1u_1 + b_2x_1^2u_1 \end{bmatrix} \\ \mathbf{y} = \mathbf{h}(\mathbf{x},\mathbf{u}) &= \left[c_1x_1 + c_3x_1x_2^2\right],\end{aligned}$$

hence

$$\begin{aligned}\dot{x}_1 &= a_0 + a_1x_1x_2 + a_2x_1^2 + b_1u_1 + b_2x_2u_1 \\ \dot{x}_2 &= a_4x_1^2x_2^2 + b_1u_1 + b_3x_1^2u_1 \\ y &= c_1x_1 + c_3x_1x_2^2,\end{aligned}$$

and application of differentiation as defined in Eq. 2.24 gives:

$$\begin{aligned}\dot{\tilde{x}}_1 &= \begin{bmatrix} a_1\bar{x}_2 + 2a_2\bar{x}_1 & a_1\bar{x}_1 \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix} + \left[b_1 + b_2\bar{x}_2\right]\tilde{u}_1 \\ \dot{\tilde{x}}_2 &= \begin{bmatrix} 2a_4\bar{x}_1\bar{x}_2^2 + 2b_3\bar{x}_1\bar{u}_1 & 2a_4\bar{x}_1^2\bar{x}_2 \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix} + \left[b_1 + b_3\bar{x}_1^2\right]\tilde{u}_1 \\ \tilde{y} &= \begin{bmatrix} c_1 + c_3\bar{x}_2^2 & 2c_3\bar{x}_1\bar{x}_2 \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix}.\end{aligned} \tag{2.25}$$

The resulting state-space equation is

$$\begin{aligned}\begin{bmatrix} \dot{\tilde{x}}_1 \\ \dot{\tilde{x}}_2 \end{bmatrix} &= \begin{bmatrix} a_1\bar{x}_2 + 2a_2\bar{x}_1 & a_1\bar{x}_1 \\ 2a_4\bar{x}_1\bar{x}_2^2 + 2b_3\bar{x}_1\bar{u}_1 & 2a_4\bar{x}_1^2\bar{x}_2 \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix} + \begin{bmatrix} b_1 + b_2\bar{x}_2 \\ b_1 + b_3\bar{x}_1^2 \end{bmatrix} \tilde{u}_1 \\ \tilde{y} &= \begin{bmatrix} c_1 + c_3\bar{x}_2^2 & 2c_3\bar{x}_1\bar{x}_2 \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix}\end{aligned} \tag{2.26}$$

Chapter 3

Satellite Equations of Motion

3.1 Dynamics of satellite with reaction wheels

Recall from Euler's moment equation [5, (Eq. 4.8.1) (p.107)] , [3, (p.50)]

$$\dot{\mathbf{h}}_{\mathbf{tot}} = \mathbf{N_e} - \omega \times \mathbf{h}_{tot} \tag{3.1}$$

where $\mathbf{h_{tot}} = \sum_{\mathbf{j}}^{\mathbf{N}} (\mathbf{I_j}\omega_{\mathbf{j}})$ is the summation over all (N) parts within the satellite and the kinematics is described by the attitude quaternion [5, (p.104)] , [1, (p.511)]

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{\Omega q} \equiv \frac{1}{2}\begin{bmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ \omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{bmatrix} \mathbf{q} \tag{3.2}$$

A rotation is described by the rotation matrix [5, Eq. A.4.15] where the four elements of the quaternion are the parameters used in the rotation matrix.

The standard form of satellite dynamics, with satellite inertia matrix $\mathbf{I}_s$, angular velocity of the satellite ω and $\mathbf{h}_w$ is the total angular momentum of wheels, all seen in the satellite's coordinate system, $\mathbf{N}_e$ is total external torque. With this notation, the dynamic equation reads

$$\frac{d}{dt}\left(\mathbf{I}_s\omega\right) + \dot{\mathbf{h}}_{\mathbf{w}} = \mathbf{N}_e - \omega \times \mathbf{I}_s\omega - \omega \times \mathbf{h}_w \tag{3.3}$$

or

$$\dot{\omega} = -\mathbf{I}_s^{-1}\left(\omega \times \mathbf{I}_s\omega\right) - \mathbf{I}_s^{-1}\omega \times \mathbf{h}_w - \mathbf{I}_s^{-1}\dot{\mathbf{h}}_w + \mathbf{I}_s^{-1}\mathbf{N}_e \tag{3.4}$$

Writing the cross product as a matrix operation using $\mathbf{S}()$ makes the notation simpler in the sequel

$$\mathbf{S}(\omega) = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{3.5}$$

gives

$$\dot{\omega} = -\mathbf{I}_s^{-1}\mathbf{S}(\omega)\mathbf{I}_s\omega - \mathbf{I}_s^{-1}\mathbf{S}(\omega)\mathbf{h}_w - \mathbf{I}_s^{-1}\dot{\mathbf{h}} + \mathbf{I}_s^{-1}\mathbf{N}_e \tag{3.6}$$

The combined nonlinear dynamic model of the satellite is completed by noting that control torque in the body coordinate system is $\mathbf{N}_c$ and gives the rate of change of the total angular momentum from wheels,

$$\dot{\mathbf{h}} = -\mathbf{N}_c, \tag{3.7}$$

and hence the dynamics of the satellite actuated by reaction wheels

$$\dot{\omega} = -\mathbf{I}_s^{-1}\mathbf{S}(\omega)\mathbf{I}_s\omega - \mathbf{I}_s^{-1}\mathbf{S}(\omega)\mathbf{h} + \mathbf{I}_s^{-1}\mathbf{N}_c + \mathbf{I}_s^{-1}\mathbf{N}_e. \tag{3.8}$$

3.2 Kinematics

The kinematics of the satellite, is described using the attitude quaternion $\mathbf{q} = [q_1, q_2, q_3, q_4]^T$ to represent a rotation. The orientation of the satellite is obtained rotating from the inertial frame to the satellite frame by the directional cosine (rotation) matrix $\mathbf{A}_{SI}$. The rotation matrix is parameterized by the quaternion, $\mathbf{A}_{SI}(\mathbf{q}_{SI})$ or just $\mathbf{A}_{SI}(\mathbf{q})$ for brevity. A vector measured in the inertial frame is $\mathbf{a}_{I,}$ the vector measured in the satellites body frame has coordinates $\mathbf{a}_S$, then $\mathbf{a}_S = \mathbf{A}_{SI}(\mathbf{q})\mathbf{a}_I$. The kinematics of the satellite is then

$$\frac{d}{dt}\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ \omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{bmatrix}\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{3.9}$$

In a compact notation,

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{\Omega}(\omega)\mathbf{q} \tag{3.10}$$

This compact notation is subsequently referred to in block diagrams. Another form of the detailed kinematics is also useful in the further analysis. Equation 3.9 is easily rewritten to separate terms with q_4 from other elements of the quaternion. Define a vector part of the first three components of the quaternion and denote this by $\mathbf{g}$, which is referred to as the Gibbs vector

$$\mathbf{g} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix}, \tag{3.11}$$

then 3.10 reads

$$\begin{aligned} \dot{\mathbf{g}} &= -\frac{1}{2}\omega \times \mathbf{g} + \frac{1}{2}q_4\omega \\ \dot{q}_4 &= -\frac{1}{2}\omega^T\mathbf{g} \end{aligned} \tag{3.12}$$

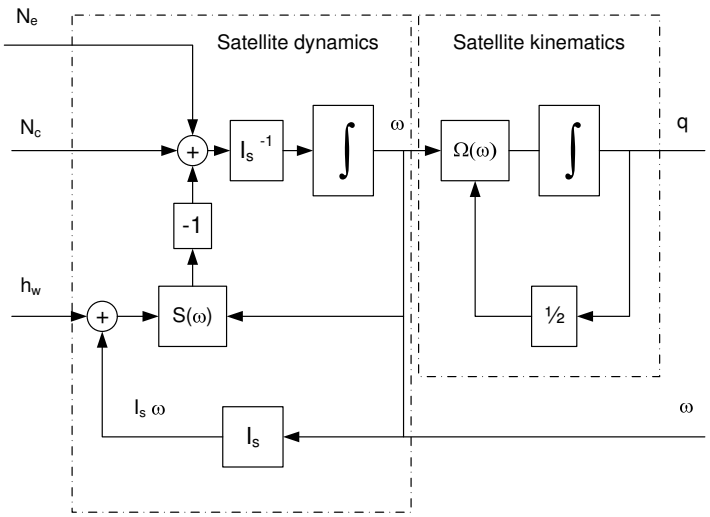


Figure 3.1: Satellite model with nonlinear dynamics and kinematics, Equations 3.4 and 3.10, $\mathbf{h}_w$ is the angular momentum of wheels, $\mathbf{N}_e$ an external torque and $\mathbf{N}_c$ the control torque from reaction wheels, all given in the SBC system. Output is $\mathbf{q}$ and ω

or written out in component form,

$$\frac{d}{dt}\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = -\frac{1}{2}\begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \\ \omega_1 & \omega_2 & \omega_3 \end{bmatrix}\begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{3.13}$$

$$+\frac{1}{2}\begin{bmatrix} q_4 & 0 & 0 \\ 0 & q_4 & 0 \\ 0 & 0 & q_4 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}. \tag{3.14}$$

We can then write the kinematics equation using $\mathbf{q} = \left[\mathbf{g}^T, q_4\right]$ where $\mathbf{g}$ is the first three components of the quaternion,

$$\frac{d}{dt}\begin{bmatrix} \mathbf{g} \\ q_4 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} -\mathbf{S}(\omega) \\ -\omega^T \end{bmatrix}\mathbf{g} + \frac{1}{2}q_4\begin{bmatrix} \mathbf{I}_{3\times 3} \\ 0 \end{bmatrix}\omega \tag{3.15}$$

The unit length of the quaternion is preserved in this operation. The model of the satellite comprise the dynamics from Equation 3.4 and the kinematics from Equation 3.10. This model is illustrated in Figure 3.1

3.3 Reaction Wheels as Actuators

A reaction wheel is an electromechanical device composed of an electric motor that drives a disc or wheel designed to have a large inertia for the mass it has. The motor can be integrated in the wheel structure. With permanent magnets incorporated in the circumference of the rotor and coils wound in the stator part, a brush-less DC motor would result.

3.3.1 Wheel Aligned with Satellite Coordinate System

assume there is a single reaction wheel in the satellite. First, assume the wheel's axes is aligned with the satellite body coordinate system. Denote the wheel inertia by $\mathbf{J}$, the angular velocity of the wheel by ω_w and the angular momentum of the wheel by $\mathbf{h}_w$. The torque from the stator to the rotor of the wheel as $\mathbf{N}_w$, then

$$\dot{\mathbf{h}} = \frac{d}{dt}\left(\mathbf{J}\omega_w\right) = \mathbf{N}_w \tag{3.16}$$

For the satellite and the wheel as an entirety, we observe from Equation 3.4 that the control torque on the satellite is

$$\mathbf{N}_c = -\dot{\mathbf{h}} = -\frac{d}{dt}\left(\mathbf{J}\omega_w\right) \tag{3.17}$$

When a reaction wheel is used as actuator, $\dot{\mathbf{h}}$ of the wheel provides the (negative) control torque on the satellite and the angular momentum of the wheel $\mathbf{h}_w$ (which was measured in satellite coordinates) gives a cross-coupling term in the dynamic equations of motion of the satellite, see Equation 3.4.

3.3.2 Wheel Aligned in Other Direction

When the wheel is not aligned with an axis of the satellite body coordinate system (SBC), the shaft axis of the wheel, is rotated with respect to the SBC. The angular momentum vector of the wheel, seen from the satellite (SBC) is then needed, using a rotation from wheel coordinates to SBC. The components of this angular momentum in the SBC system is a projection of the wheel's angular momentum along its axis of rotation. The projection is a column vector, the directional cosines of which are

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}_{sbc} = \begin{bmatrix} \mathbf{e}_w \cdot \mathbf{e}_{1s} \\ \mathbf{e}_w \cdot \mathbf{e}_{2s} \\ \mathbf{e}_w \cdot \mathbf{e}_{3s} \end{bmatrix} \left[|\mathbf{h}|\right]_w = \begin{bmatrix} a_{1w} \\ a_{2w} \\ a_{3w} \end{bmatrix} \left[h\right]_w \tag{3.18}$$

If the wheel assembly consists of several wheels, say four, we have the directional cosines for each wheel in the columns of the direction matrix for

the wheels,

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}_{sbc} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{12} & a_{13} & a_{14} \\ a_{31} & a_{12} & a_{13} & a_{14} \end{bmatrix} \begin{bmatrix} h_{w1} \\ h_{w2} \\ h_{w3} \\ h_{w4} \end{bmatrix} \tag{3.19}$$

Abbreviated to vector form, this is

$$\mathbf{h} = \mathbf{A}_w \mathbf{h}_w \tag{3.20}$$

The matrix $\mathbf{A}_w$ has three rows and a number of columns equal to the number of reaction wheels in the satellite.

3.4 Actuator Dynamics - the Wheel Control Loop

The above model assumed an ideal wheel servo, i.e. the wheel can accelerate as fast as demanded. In practice, however, a torque control loop can not be ideal but has a limited, although high bandwidth. The block diagram in Fig. 3.2 shows the schematics of a wheel control loop.

Referring to the idealized case (A) in Figure 3.2

$$\mathbf{N}_c(t) = -\mathbf{N}_w(t) \tag{3.21}$$

In case (B) of Figure 3.2, the speed control loop around $\dot{h}$ gives rise to a low-pass dynamic relation between $\mathbf{N}_w$ and $\mathbf{N}_c$.

$$\begin{aligned} \tau_w \ddot{h} &= -\dot{h} + N_w \\ N_c &= -\dot{h} \\ h(t) &= \int_0^t \dot{h}(\tau) d\tau + h(0) \end{aligned} \tag{3.22}$$

In the Laplace domain, torque on the satellite reads, from Eq. 3.22

$$N_c(s) = -\frac{1}{1 + s\tau_w} N_w(s) \tag{3.23}$$

3.4.1 Combined Model

The combined model of satellite and wheels is shown in Figure 3.3

3.5 Nonlinear model

The combined dynamic and kinematics equations Eqs. 3.4 and 3.15give the general nonlinear model for the satellite's angular motion,

$$\frac{d}{dt} \begin{bmatrix} \omega \\ \mathbf{g} \\ q_4 \\ \mathbf{h} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_s^{-1} \left(-\mathbf{S}(\omega)\mathbf{I}_s\omega - \mathbf{S}(\omega)\mathbf{h} + \mathbf{N}_c + \mathbf{N}_{dist} \right) \\ -\frac{1}{2}\mathbf{S}(\omega)\mathbf{g} + \frac{1}{2} q_4 \mathbf{I}_{3\times 3}\omega \\ -\frac{1}{2}\omega^T \mathbf{g} \\ -\mathbf{N}_c \end{bmatrix} \tag{3.24}$$

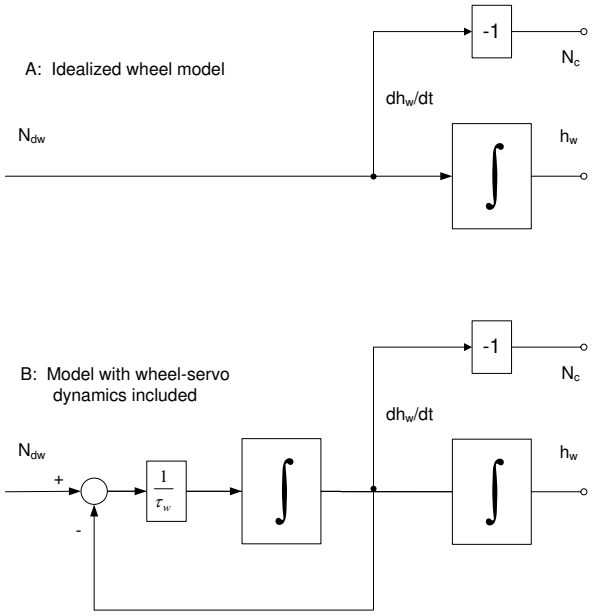


Figure 3.2: Schematic wheel control loop. The upper part (A) shows the idealized model where $\dot{h}$ is identical to the demand torque. The bottom part (B) shows a more realistic configuration where a speed control servo loop is closed around the wheel. This gives the generation of $\dot{h}$ a low-pass behaviour. The bandwidth of the $\dot{h}$ loop is equal to ω_w.

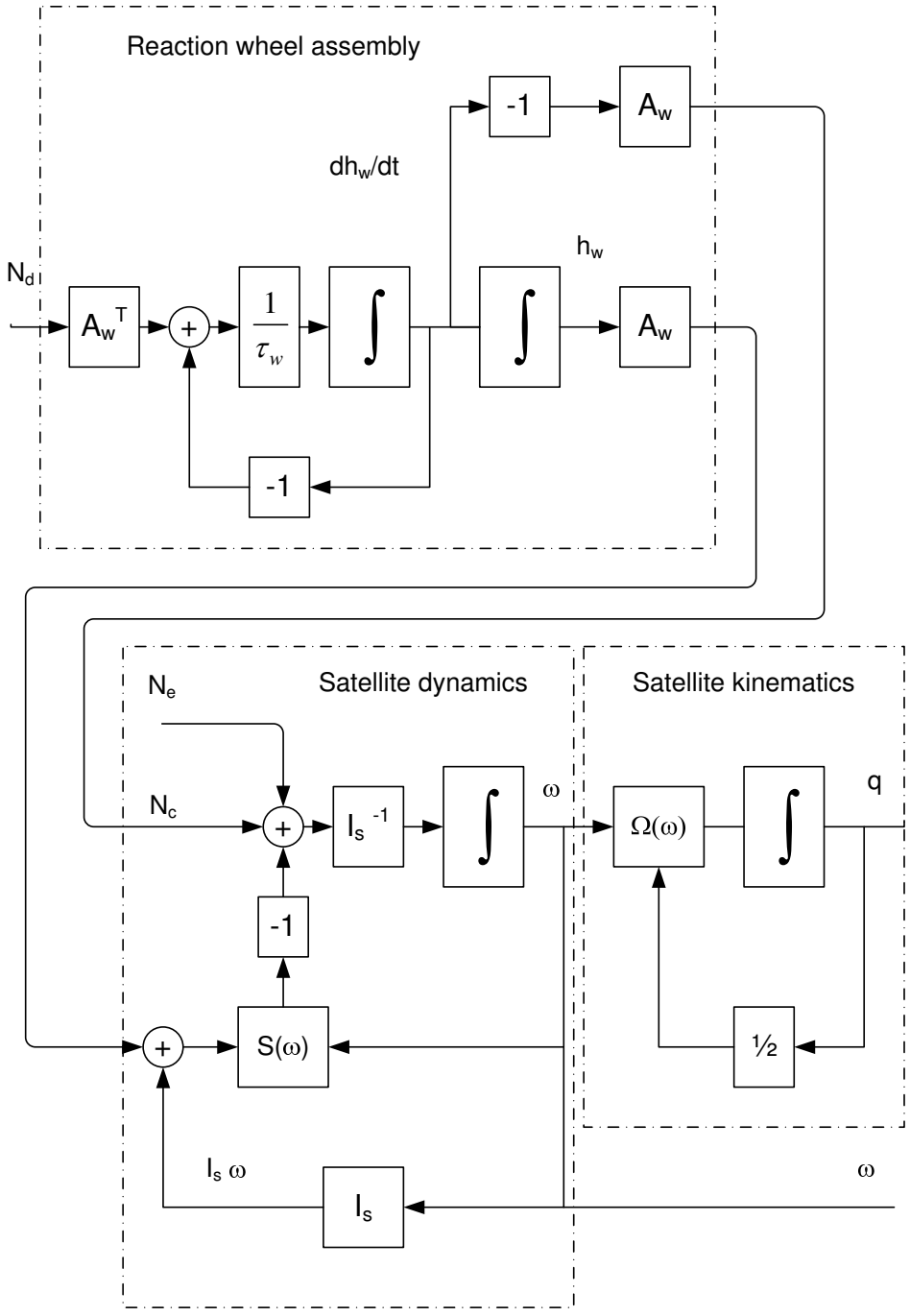


Figure 3.3: Satellite model with reaction wheel assembly as actuator, satellite dynamics and kinematics.

Limitations of the model

It is noted that the effects of flexibility of the satellite's structure or appendages is not included in this model.

Remark 1 *Change of reference coordinate system*
As a remark, it is noted that the dynamic model has a structure that is invariant with the choice of coordinate system. As long as all quantities are referred to the same coordinate system, the structure of the equation is unchanged. The values of elements in the inertia tensor do change, needles to say. This is seen as follows.
The effects caused by a rotation of both measured values and reference coordinates for the inertia tensor to a satellite geometry defined coordinate system is achieved by rotating the various quantities in the dynamic equation,

$$\frac{d}{dt}\left(\mathbf{I}_s\boldsymbol{\omega_p}\right) = -\boldsymbol{\omega}_p \times \mathbf{I}_s\boldsymbol{\omega}_p - \boldsymbol{\omega_p} \times \mathbf{h}_p - \dot{\mathbf{h}}_p + \mathbf{N}_{dist,p} \tag{3.25}$$

then, rotating to the body system using $\boldsymbol{\omega}_p = \mathbf{A}_{pb}\boldsymbol{\omega}_b$, $\mathbf{h}_p = \mathbf{A}_{pb}\mathbf{h}_b$, $\mathbf{N}_p = \mathbf{A}_{pb}\mathbf{N}_b$, $\mathbf{I}_s = \mathbf{A}_{pb}\mathbf{I}_{sb}\mathbf{A}_{pb}^T$ *gives*

$$\begin{aligned}\frac{d}{dt}\left(\mathbf{A}_{pb}\mathbf{I}_{sb}\mathbf{A}_{pb}^T\mathbf{A}_{pb}\boldsymbol{\omega}_b\right) &= -\mathbf{A}_{pb}\boldsymbol{\omega}_b \times \mathbf{A}_{pb}\mathbf{I}_{sb}\mathbf{A}_{pb}^T\mathbf{A}_{pb}\boldsymbol{\omega}_b \\ &-\mathbf{A_{pb}}\boldsymbol{\omega}_b \times \mathbf{A}_{pb}\mathbf{h}_{pb} - \mathbf{A}_{pb}\dot{\mathbf{h}}_{pb} + \mathbf{A}_{pb}\mathbf{N}_{e,b} + \mathbf{A}_{pb}\mathbf{N}_{dist,b}\end{aligned} \tag{3.26}$$

since $\mathbf{A}\boldsymbol{\omega} \times \mathbf{A}\mathbf{h} = \mathbf{A}(\boldsymbol{\omega} \times \mathbf{h})$, *this simplifies to*

$$\frac{d}{dt}\left(\mathbf{I}_{sb}\boldsymbol{\omega_b}\right) = -\boldsymbol{\omega}_b \times \mathbf{I}_{sb}\boldsymbol{\omega}_b - \boldsymbol{\omega}_b \times \mathbf{h}_b - \dot{\mathbf{h}}_b + \mathbf{N}_{e,b} + \mathbf{N}_{dist,b} \tag{3.27}$$

which is again the dynamics in its original structure but now expressed in body axes instead of principal axes.

Chapter 4

Nonlinear Simulation Model

The complete nonlinear model of satellite dynamics and kinematics, and wheel actuation was implemented in Simulink® in the program sat3_sim.mdl (by the author).

The torque on the satellite is $N_c = -A_w N_{dw}$ where N_{dw} is the wheel torque demand and A_w the distribution matrix for wheel torque that is defined in the Matlab program RWA_sim.m. The distribution matrix has its number of columns equal to the number of wheels. The number of rows is always 3, since the N_c torque is the torque the three axes of the satellite coordinate system.

The spacecraft parameters used are, as an example:

- Principal moments of inertia $I_{xx} = 5,\ I_{yy} = 6,\ I_{zz} = 7$ kgm^2.
- The maximum angular momentum of the reaction wheel 0.12 Nms.
- The maximum wheel speed 280 rad/sec.
- moments of inertia of the reaction wheel 0.00043 kgm^2.
- The nominal wheel speed 100 rad/sec.
- The closed loop bandwidth the reaction wheel 10.0 rad/s.
- Maximum torque produced by the reaction wheel 0.0075 N.
- The reaction wheel assembly is defined by the number of wheels and their alignment.

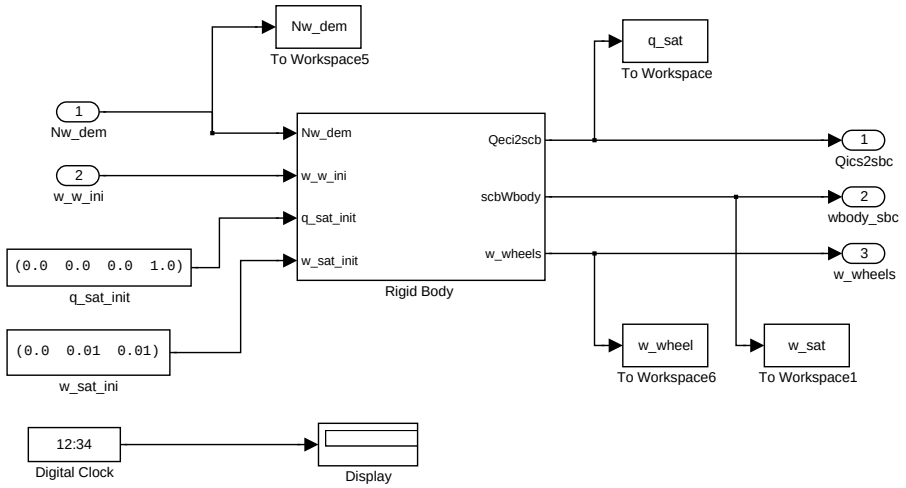


Figure 4.1: Simulink mainwindow for sat3_sim.mdl. Input are control torque demand, initial values for attitude quaternion of satellite, angular rate of satellite, initial val be present in the workspace before you start a simulation.

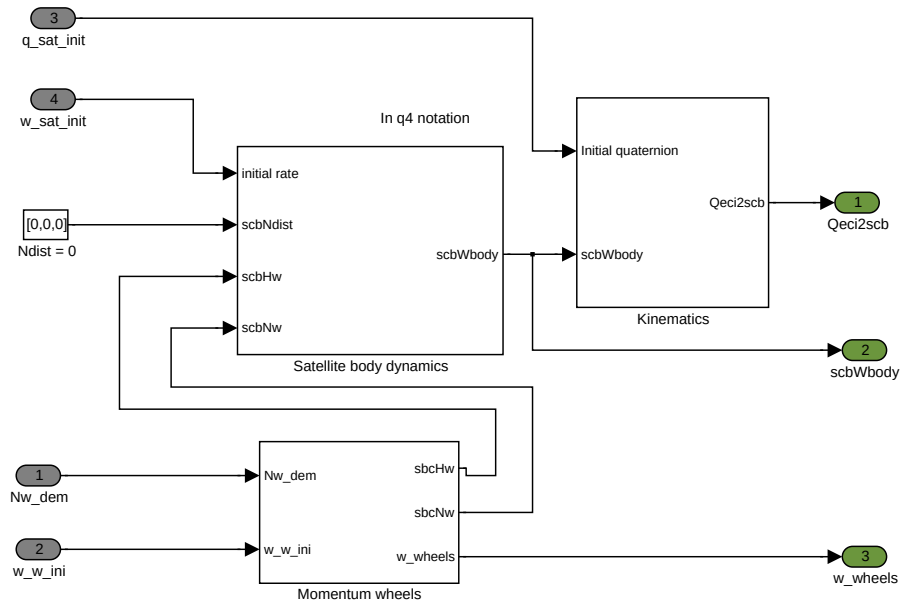


Figure 4.2: Level 2 of sat3sim with three components: satellite dynamics, satellite kinematics and wheel model.

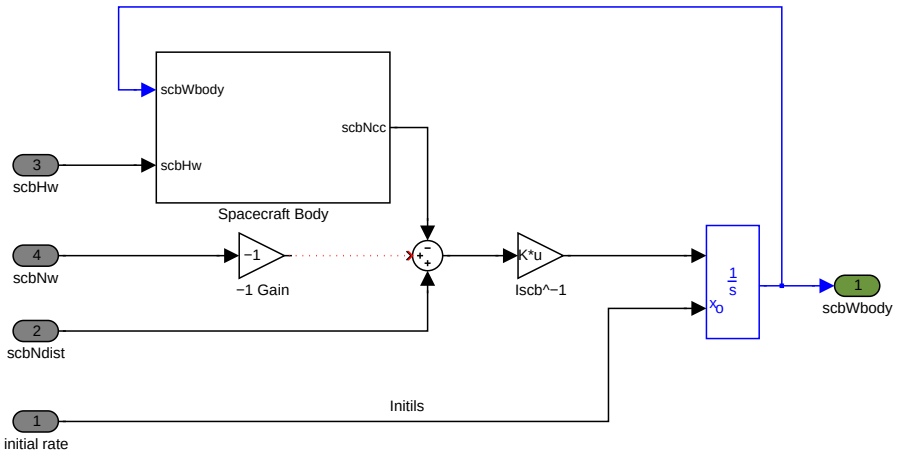


Figure 4.3: Satellite dynamics is a direct implementation of Eulers moment equation.

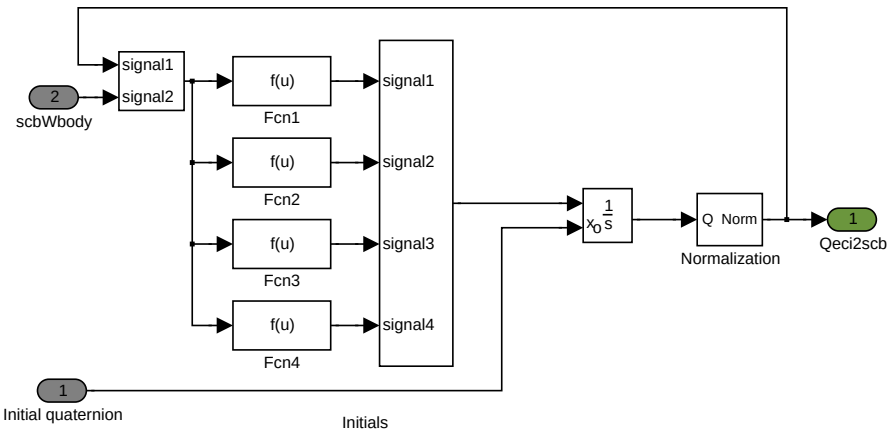


Figure 4.4: Kinematics implemented using quaternion parameterization.

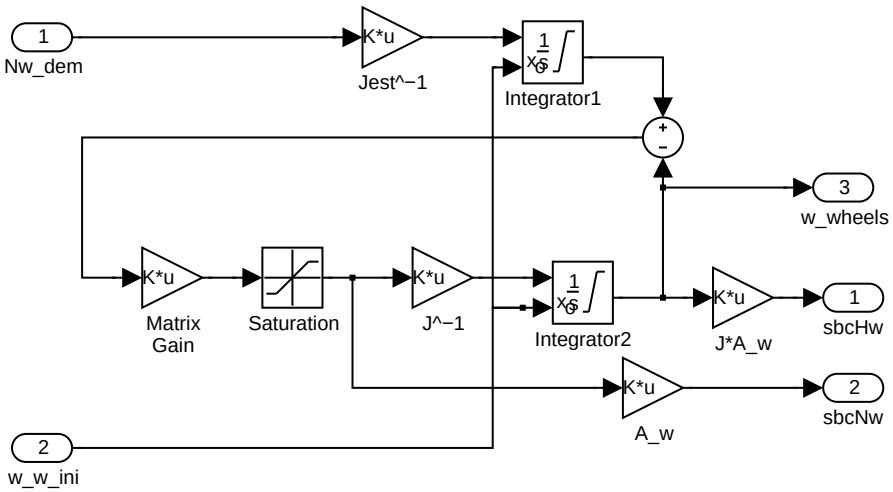


Figure 4.5: Simulation of reaction wheel assembly. The number of wheels are determined by the dimension of the number of columns in the A_war matrix, defined in the Matlab function RWA_sim.

- An example for n_w = 4 (four wheels) is the thetraeder directions of the wheels and the distribution matrix:

$$\mathbf{A}_w = \begin{bmatrix} \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} & -\sqrt{\frac{1}{3}} \\ \sqrt{\frac{2}{3}} & -\sqrt{\frac{2}{3}} & 0 & 0 \\ 0 & 0 & -\sqrt{\frac{2}{3}} & \sqrt{\frac{2}{3}} \end{bmatrix}. \tag{4.1}$$

- An example with three orthogonal wheels aligned with the axes of the satellite gives the distribution matrix

$$\mathbf{A}_w = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.2}$$

Chapter 5

Linear Model for Analysis

This non-linear equation of motion is now linearised in an arbitrary point of operation $(\bar{\boldsymbol{\omega}}, \bar{\mathbf{g}}, \bar{q}_4, \bar{h})$ in order to arrive at a set of linear state space equations. This is needed to enable a stringent stability and performance analysis for the linear control systems. The deviation from steady state (point of linearization) is denoted by a tilde above the variables, $\boldsymbol{\omega} = \bar{\boldsymbol{\omega}} + \tilde{\boldsymbol{\omega}}$, but the quaternion representation of attitude poses a specific problem. With $d\mathbf{g}$ denoting the orientation at time $t + dt$ relative to the attitude at time t, then, since

$$\begin{bmatrix} \tilde{g} \\ \tilde{q}_4 \end{bmatrix} = \begin{bmatrix} dg_1 \\ dg_2 \\ dg_3 \\ dq_4 \end{bmatrix} = \begin{bmatrix} e_1 \sin\left(\frac{1}{2}\omega dt\right) \\ e_2 \sin\left(\frac{1}{2}\omega dt\right) \\ e_3 \sin\left(\frac{1}{2}\omega dt\right) \\ \cos\left(\frac{1}{2}\omega dt\right) \end{bmatrix} \simeq \begin{bmatrix} \frac{1}{2}\omega_1 dt \\ \frac{1}{2}\omega_2 dt \\ \frac{1}{2}\omega_3 dt \\ 1 \end{bmatrix} \tag{5.1}$$

then $\frac{d}{dt}q_4 = 0$ and $\mathbf{S}(\omega)d\mathbf{g} = \mathbf{0}$ hence

$$\frac{d}{dt}\tilde{\mathbf{g}} = -\frac{1}{2}\mathbf{S}(\boldsymbol{\omega})\tilde{\mathbf{g}} + \frac{1}{2}\tilde{q}_4\mathbf{I}_{3\times 3}\,\boldsymbol{\omega} = \frac{1}{2}\mathbf{I}_{3\times 3}\boldsymbol{\omega} \tag{5.2}$$

$$\mathbf{h} = \bar{\mathbf{h}} + \tilde{\mathbf{h}}; \quad \frac{d}{dt}\mathbf{h} = \frac{d}{dt}\tilde{\mathbf{h}} \tag{5.3}$$

The desired form of the linear equation of motion has a state vector

$$\mathbf{x} = (\tilde{\omega}_1, \tilde{\omega}_2, \tilde{\omega}_3, \tilde{g}_1, \tilde{g}_2, \tilde{g}_3, h_1, h_2, h_3)^T$$

and has control input $\mathbf{u} = \mathbf{N}_c$. The external and disturbance torques have zero mean. The state space equation is then

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}_u(t)\mathbf{N}_c(t) + \mathbf{B}_d(t)\mathbf{N}_{dist}(t) \tag{5.4}$$

where

$$A_{ij} = \frac{\partial f_i}{\partial x_j}; \quad B_{ij} = \frac{\partial f_i}{\partial u_j} \tag{5.5}$$

and

$$f = \begin{bmatrix} -\mathbf{I}_s^{-1}\mathbf{S}(\boldsymbol{\omega})\mathbf{I}_s\boldsymbol{\omega} - \mathbf{I}_s^{-1}\mathbf{S}(\boldsymbol{\omega})\mathbf{h} + \mathbf{I}_s^{-1}\left(\mathbf{N}_c + \mathbf{N}_{dist}\right) \\ -\frac{1}{2}\mathbf{S}(\boldsymbol{\omega})\mathbf{g} + \frac{1}{2}q_4\mathbf{I}_{3\times 3}\boldsymbol{\omega} \\ -\frac{1}{2}\boldsymbol{\omega}^T\mathbf{g} \\ -\mathbf{N}_c \end{bmatrix} \tag{5.6}$$

Using symbolic manipulation to calculate the Jacobians Eq. 5.5, we obtain

$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_s^{-1}\mathbf{A}_{\omega,\omega} & \mathbf{0} & \mathbf{I}_s^{-1}\mathbf{A}_{\omega,h} \\ \frac{1}{2}\mathbf{I}_{3\times 3} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}; \tag{5.7}$$

$$\mathbf{B}_u = \begin{bmatrix} \mathbf{I}_s^{-1} \\ \mathbf{0} \\ -\mathbf{I}_{3\times 3} \end{bmatrix};\ \mathbf{B}_d = \begin{bmatrix} \mathbf{I}_s^{-1} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix};$$

with

$$\mathbf{A}_{\omega,\omega} = [\mathbf{A}_{\omega,1}, \mathbf{A}_{\omega,2}, \mathbf{A}_{\omega,3}] \tag{5.8}$$

and the three columns of $\mathbf{A}_{\omega,\omega}$ are

$$\mathbf{A}_{\omega,1} = \begin{bmatrix} \omega_2 I_{31} - \omega_3 I_{21} \\ -2I_{31}\omega_1 - I_{32}\omega_2 - \omega_3 I_{33} + \omega_3 I_{11} + h_3 \\ I_{21}\omega_1 + \omega_2 I_{22} + I_{23}\omega_3 - \omega_2 I_{11} - h_2 \end{bmatrix} \tag{5.9}$$

$$\mathbf{A}_{\omega,2} = \begin{bmatrix} I_{31}\omega_1 + 2I_{32}\omega_2 + \omega_3 I_{33} - \omega_3 I_{22} - h_3 \\ \omega_3 I_{12} - \omega_1 I_{32} \\ -\omega_1 I_{11} - 2I_{12}\omega_2 - I_{13}\omega_3 + \omega_1 I_{22} + h_1 \end{bmatrix} \tag{5.10}$$

$$\mathbf{A}_{\omega,3} = \begin{bmatrix} -I_{21}\omega_1 - \omega_2 I_{22} - 2I_{23}\omega_3 + \omega_2 I_{33} + h_2 \\ \omega_1 I_{11} + I_{12}\omega_2 + 2I_{13}\omega_3 - \omega_1 I_{33} - h_1 \\ \omega_1 I_{23} - \omega_2 I_{13} \end{bmatrix} \tag{5.11}$$

Further,

$$\mathbf{A}_{\omega,h} = \begin{bmatrix} 0 & \omega_3 & -\omega_2 \\ -\omega_3 & 0 & \omega_1 \\ \omega_2 & -\omega_1 & 0 \end{bmatrix} \tag{5.12}$$

Equation 5.4 with matrices defined in Equations 5.7, 5.8, 5.9, 5.10, 5.11 and 5.12 constitute the linear model of the satellite in an arbitrary nominal condition (point of operation). The nominal condition is expressed through the parameters $\omega = \bar{\omega}$, the average angular rate of the satellite, and $h = \bar{h}$, the average stored angular momentum expressed in satellite body coordinates.

This linear model accept an arbitrary moment of inertia tensor, which enables subsequent use for both controller design and analysis of sensitivity (robustness) properties. Uncertainties include magnitude and rotation of the inertia tensor and alignment of wheels. The basic dynamic properties

change with the resulting angular momentum of the wheels. The changes in linear dynamics could be analyzed should the satellite be demanded to rotate along one of its axes, e.g. during maneuvers. The linear model is then available should quantitative analysis be desired. Nonlinear analysis is limited to mainly stating the more fundamental question of stability.

Example 1 *Inertial Pointing Satellite*

Let the following parameters apply for a satellite in inertial pointing condition. Assume the principal axes coincide with the satellite body coordinate system.

$$\boldsymbol{\omega} = (0,0,0)[rad/s]; \quad h_i \in [0, h_{\max}]\,[Nms]; \tag{5.13}$$

$$\mathbf{I}_s = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} [kgm^2] \tag{5.14}$$

The nominal model for small signals (linear model) case is hence, when the satellite has $\omega = \mathbf{0}$,

$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_s^{-1}\mathbf{A}_{\omega,\omega} & \mathbf{0} & \mathbf{0} \\ \frac{1}{2}\mathbf{I}_{3\times 3} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}; \tag{5.15}$$

$$\mathbf{B}_u = \begin{bmatrix} \mathbf{I}_s^{-1} \\ \mathbf{0} \\ -\mathbf{I}_{3\times 3} \end{bmatrix}; \; \mathbf{B}_d = \begin{bmatrix} \mathbf{I}_s^{-1} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}; \tag{5.16}$$

and $\mathbf{A}_{\omega,\omega}$ *of Equations 5.8, 5.9, 5.10 and 5.11 simplifies to*

$$\mathbf{A}_{\omega,\omega} = \begin{bmatrix} 0 & -h_3 & h_2 \\ h_3 & 0 & -h_1 \\ -h_2 & h_1 & 0 \end{bmatrix} \tag{5.17}$$

Example 2 *Earth pointing satellite An Earth-pointing satellite in a circular orbit has orbit period* T_o. *The orbit rate is* $\omega_o = \frac{2\pi}{T_o}$. *The nominal angular rate of the satellite is then* $\boldsymbol{\omega}$*=(0,-*ω_o*,0) [rad/s]. The negative sign arrives from the definition of the orbit coordinate system with its x-axis along the satellite's velocity vector and the z-axis pointing towards Nadir (Center of the Earth). Investigation of properties of the satellite should be done by linearizing the system at* $\boldsymbol{\omega}$*=(0,-*ω_o*,0).*

5.1 Linear Satellite Model with Actuator Dynamics

When the dynamic model of satellite Eqs. 5.4 and 5.4 gets its input torque from wheels with actuator dynamics described in Eq. 3.22, the wheel state $\dot{h}$ need be added to the state vector. The linear dynamic model of the satellite is hence extended with three states $\dot{h_x}$, $\dot{h_y}$ and $\dot{h_z}$ for $\dot{h}$ in each of the three axes of the satellite body coordinate system, denoted sbc. The state vector for the aggregated model is $\mathbf{x} = \left(\omega, \mathbf{g}, \mathbf{h}_{sbc}, \dot{\mathbf{h}}_{sbc}\right)^T$. Input to the state equation is $\mathbf{N}_d$ and $\mathbf{N}_{dist}(t)$

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}_u\mathbf{N}_d(t) + \mathbf{B}_d\mathbf{N}_{dist}(t) \tag{5.18}$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_s^{-1}\mathbf{A}_{\omega,\omega} & \mathbf{0} & \mathbf{I}_s^{-1}\mathbf{A}_{\omega,h} & -\mathbf{I}_s^{-1} \\ \frac{1}{2}\mathbf{I}_{3\times3} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_{3\times3} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -k_w\mathbf{I}_{3\times3} \end{bmatrix}, \tag{5.19}$$

$$\mathbf{B}_u = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ -k_w\mathbf{I}_{3\times3} \end{bmatrix}, \mathbf{B}_u = \begin{bmatrix} \mathbf{I}_s^{-1} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

Note that torque demand and the wheel angular momentum vectors are in satellite body coordinates in this equation.

The wheel dynamics should be included when a high performance satellite control loop is designed. For more basic designs, it is certainly sufficient to choose gains in the satellite attitude control such that the poles of the closed loop satellite control are 10-15 times lower than the wheel bandwidth (w_bw in the parameter list below). The poles for the satellite with closed loop attitude control are found using either of the following Matlab commands
damp(satmodcl) or
[p,z]=pzmap(satmodcl)
on the LTI structure *satmodcl* for the closed loop system. The latter command provides both system poles and system zeros.

Chapter 6

Matlab Function to get Linearised Model

The different versions of the models described above are calculated by the Matlab function *lin_sat_wheels_mod*. The call is

[satmod_c, satmod_dist] = lin_sat_wheels_mod(Is, ws, hs, w_bw, Ts)

Input and output parameters are described in the table:

variable	unit	type	explanation
input			
Is	kgm^2	matrix (3,3)	Satellite's inertia matrix
ws	rad/s	vector (3,1)	Satellite's mean angular velocity
hs	Nms	vector (n,1)	Mean ang. wheel ang. momentum in sbc
w_bw	rad/s	scalar	Wheel bandwidth - same for all wheels
Ts	s	scalar	Sampling time for a discrete time model
output			use Ts = 0 to get continuous time model
satmod_c		Matlab LTI	model using N_{dw} as input
satmod_dist		Matlab LTI	model using N_{dist} as input

The last two parameters *w_bw* and *Ts* can be omitted from the call.

6.1 Linear model without wheel dynamics

To obtain the model of Eq. 5.4 for the case with idealized wheels, use the call

[satmod_c, satmod_dist] = lin_sat_wheels_mod(Is, ws, hs)

The result is a set of two Matlab LTI (Linear Time Invariant) structures, *satmod_c*, *satmod_dist*, that comprises the system description in state space

form. The first comprise the model using wheel torque as input. The second describes the system using disturbance torque as input. The matrices of the state space model can be retained using the command

[A, Bu, C, D] = ssdata(*satmod_c*)

[A, Bd, C, D] = ssdata(*satmod_dist*)

6.2 Linear Models with wheel dynamics

To obtain the model of Eq. 5.19 with wheel dynamics, use the call
[satmod_c, satmod_dist] = lin_sat_wheels_mod(Is, ws, hs, w_bw)

6.3 Linear model for sampled system

To obtain a sampled (discrete time) model for the case with wheel dynamics included, use the call

[satmod_c, satmod_dist] = lin_sat_wheels_mod(Is, ws, hs, w_bw, Ts)

where Ts is the sampling time in seconds.

Chapter 7

Attitude Control

Attitude control means to use the measurements and a reference to calculate a torque demand that will make the measured state equal to the reference. Following the argument in [5, (pp154-156)], that the satellite body coordinate system and a target coordinate system will be aligned when zero rotation is needed to align the satellite body system with the axes of the target system. This can be done using a control law

$$\mathbf{u} = -\mathbf{K}_p\mathbf{q}_e - \mathbf{K}_{pd}\omega \tag{7.1}$$

where $\mathbf{K}_p$and $\mathbf{K}_{pd}$ are controller gain matrices, ω the angular velocity vector of the satellite and $\mathbf{q}_e$ is the quaternion error,

$$\begin{bmatrix} q_{1e} \\ q_{2e} \\ q_{3e} \\ q_{4e} \end{bmatrix} = \begin{bmatrix} q_{4r} & q_{3r} & -q_{2r} & -q_{1r} \\ -q_{3r} & q_{4r} & q_{1r} & q_{2r} \\ q_{2r} & -q_{1r} & q_{4r} & -q_{3r} \\ q_{1r} & q_{2r} & q_{3r} & q_{4r} \end{bmatrix} \begin{bmatrix} q_{1s} \\ q_{2s} \\ q_{3s} \\ q_{4s} \end{bmatrix} \tag{7.2}$$

i.e. a quaternion that describes the rotation needed to align the satellite body coordinate axes with those of the target coordinate system. The quaternion $\mathbf{q}_s$ describes the rotation to go from inertial axes to statelite body coordinate system axes, $\mathbf{q}_r$ the rotation of inertial axes to the reference (target) coordinate system axes.

The feedback in Eq. 7.1 has $\mathbf{q}_t$ as reference for pointing, but no reference in ω. When we want to make a reorientation manoeuvre with a spacecraft, it could be convenient to introduce a reference also in ω, to make a maneuvre that can follow an angular rate, e.g. when a satellite shall track another object or a point on Earth. A control law with reference in both quaternion and in angular velocity is:

$$\mathbf{u} = -\mathbf{K}_p\mathbf{q}_e - \mathbf{K}_{pd}\omega_e \tag{7.3}$$

where

$$\omega_e = \omega_s - \omega_r \tag{7.4}$$

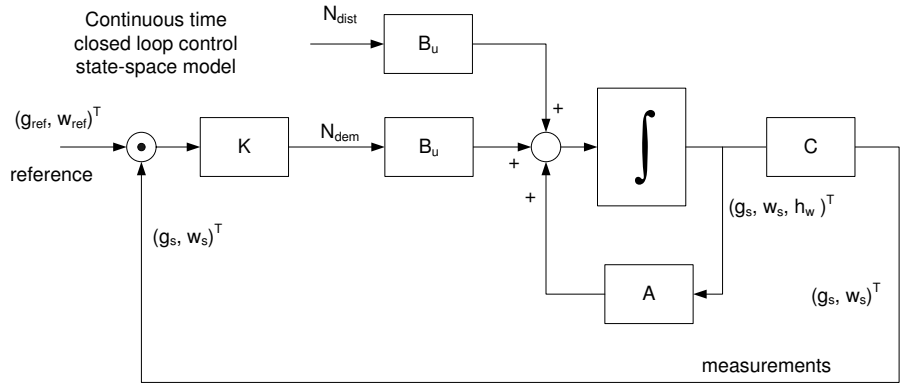


Figure 7.1: Closed loop with output feedback and reference to the loop from both ω and $\mathbf{g}$ in the linear state space representation of the satellite.

We will analyse both cases in the sequel, Eq. 7.3 with reference values in both angular velocity and attitude, Eq. 7.1 with reference quaternion but zero angular velocity as reference.

7.1 Reference in both angular rate and attitude

Attitude control means to use the measurements and a reference from both angular rate and direction (quaternion) to obtain a closed loop control as illustrated in Figure 7.1

The feedback law is taken to be

$$\mathbf{N}_d = -\mathbf{K}_p \begin{bmatrix} q_{1e}q_{4e} \\ q_{2e}q_{4e} \\ q_{3e}q_{4e} \end{bmatrix} - \mathbf{K}_{pd} \begin{bmatrix} \omega_{1e} \\ \omega_{2e} \\ \omega_{3e} \end{bmatrix} \tag{7.5}$$

where the quaternion error is defined as

$$\mathbf{q}_e = \mathbf{q}_s^{-1}\mathbf{q}_r \tag{7.6}$$

and the angular velocity error is

$$\omega_e = \omega_s - \omega_r \tag{7.7}$$

For small angles, and taking q_4 positive,

$$\mathbf{q}_e \simeq \begin{bmatrix} g_{1r} - g_{1s} \\ g_{2r} - g_{2s} \\ g_{3r} - g_{3s} \\ 1 \end{bmatrix}. \tag{7.8}$$

and the control law can be written as

$$\begin{aligned}\mathbf{N}_d &= \mathbf{K}_p \begin{bmatrix} g_{1r} - g_{1s} \\ g_{2r} - g_{2s} \\ g_{3r} - g_{3s} \end{bmatrix} + \mathbf{K}_{pd} \begin{bmatrix} \omega_{1r} - \omega_{1s} \\ \omega_{2r} - \omega_{2s} \\ \omega_{3r} - \omega_{3s} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{K}_{pd} & \mathbf{K}_p \end{bmatrix} \begin{bmatrix} \omega_{1r} - \omega_{1s} \\ \omega_{2r} - \omega_{2s} \\ \omega_{3r} - \omega_{3s} \\ g_{1r} - g_{1s} \\ g_{2r} - g_{2s} \\ g_{3r} - g_{3s} \end{bmatrix} \end{aligned} \tag{7.9}$$

The linear model has the state vector $x = [\omega_1, \omega_2, \omega_3, g_1, g_2, g_3, h_1, h_2, h_3]^T$ and the measurement is $y = [\omega_1, \omega_2, \omega_3, g_1, g_2, g]^T$. With reference in both ω and $\mathbf{g}$, $\mathbf{y}_{ref} = [[\omega_{1r}, \omega_{2r}, \omega_{3r}, g_{1r}, g_{1r}, g_{3r}]^T$ y this is the same as

$$\begin{aligned}\mathbf{N}_d &= \begin{bmatrix} \mathbf{K}_{pd} & \mathbf{K}_p \end{bmatrix} (\mathbf{y}_r - \mathbf{y}) \\ &= - \begin{bmatrix} \mathbf{K}_{pd} & \mathbf{K}_p \end{bmatrix} \mathbf{C}\mathbf{x} + \begin{bmatrix} \mathbf{K}_{pd} & \mathbf{K}_p \end{bmatrix} \mathbf{y}_r \\ &= -\mathbf{K}_y \mathbf{C}\mathbf{x} + \mathbf{K}_r \mathbf{y}_r \end{aligned} \tag{7.10}$$

where the last two lines define the gain matrices $\mathbf{K}_y = \begin{bmatrix} \mathbf{K}_{pd} & \mathbf{K}_p \end{bmatrix}$ and $\mathbf{K}_r = \begin{bmatrix} \mathbf{K}_{pd} & \mathbf{K}_p \end{bmatrix}$.

The transfer function from reference to output with the closed feedback is then

$$\mathbf{y}(s) = \mathbf{C}(s\mathbf{I}_{9,9} - (\mathbf{A} - \mathbf{B}_u \mathbf{K}_y \mathbf{C}))^{-1} \mathbf{B}_u \mathbf{K}_r \mathbf{y}_r(s) \tag{7.11}$$

The form of this expression is

$$\mathbf{y}(s) = \mathbf{C}(s\mathbf{I}_{9,9} - \mathbf{A}_{cl})^{-1} \mathbf{B}_{cl} \mathbf{y}_r(s) \tag{7.12}$$

with

$$\mathbf{A}_{cl} = (\mathbf{A} - \mathbf{B}_u \mathbf{K}_y \mathbf{C}) \tag{7.13}$$

$$\mathbf{B}_{cl} = \mathbf{B}_u \mathbf{K}_r = \mathbf{B}_u \begin{bmatrix} \mathbf{K}_{pd} & \mathbf{K}_p \end{bmatrix} \tag{7.14}$$

$$\mathbf{C}_{cl} = \mathbf{C} \quad \text{and} \quad \mathbf{D}_{cl} = \mathbf{0}_{6,6} \tag{7.15}$$

The closed loop transfer functions 7.12 are calculated using the commonplace Matlab functions (see below), using the LTI system definition

$$sys\_cl = ss(\mathbf{A}_{cl}, \mathbf{B}_{cl}, \mathbf{C}_{cl}, \mathbf{D}_{cl}). \tag{7.16}$$

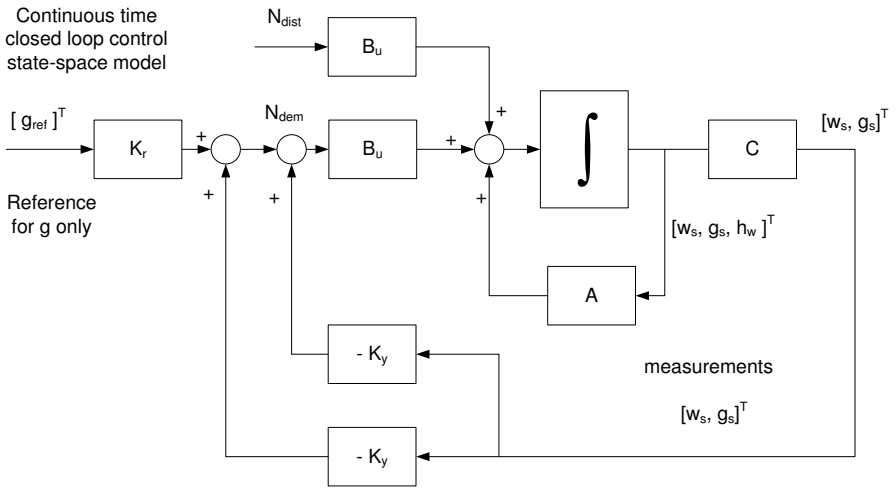


Figure 7.2: Closed loop with output feedback and reference to the loop from **g** only. Linear state space model of the satellite.

7.2 Reference in Attitude

When the reference is not desired in angular velocity, the block diagram 7.2 is applicable.

With reference from **g** only, $\mathbf{y}_r = [g_{1r}, g_{2r}, g_{3r}]^T$. This gives

$$\begin{aligned}\mathbf{N}_d &= -\begin{bmatrix}\mathbf{K}_{pd} & \mathbf{K}_p\end{bmatrix}\mathbf{y} + \mathbf{K}_p\mathbf{g}_r \\ &= -\begin{bmatrix}\mathbf{K}_{pd} & \mathbf{K}_p\end{bmatrix}\mathbf{C}\mathbf{x} + \mathbf{K}_p\mathbf{g}_r \\ &= -\mathbf{K}_y\mathbf{C}\mathbf{x} + \mathbf{K}_r\mathbf{g}_r\end{aligned} \tag{7.17}$$

where the last two lines define the gain matrices $\mathbf{K}_y$ and $\mathbf{K}_r$ The transfer function from reference to output is therefore

$$\mathbf{y}(s) = \mathbf{C}(s\mathbf{I}_{9,9} - (\mathbf{A} - \mathbf{B}_u\mathbf{K}_y\mathbf{C}))^{-1}\mathbf{B}_u\mathbf{K}_r\mathbf{g}_r(s) \tag{7.18}$$

Using the abbreviations

$$\mathbf{A}_{cl} = (\mathbf{A} - \mathbf{B}_u\mathbf{K}_y\mathbf{C}) \tag{7.19}$$

$$\mathbf{B}_{cl} = \mathbf{B}_u\mathbf{K}_r = \mathbf{B}_u\mathbf{K}_p \tag{7.20}$$

$$\mathbf{C}_{cl} = \mathbf{C} \quad \text{and} \quad \mathbf{D}_{cl} = \mathbf{0}_{6,3} \tag{7.21}$$

we can again analyse the closed loop system using the form

$$\mathbf{y}(s) = \mathbf{C}(s\mathbf{I}_{9,9} - \mathbf{A}_{cl})^{-1}\mathbf{B}_{cl}\mathbf{y}_r(s) \tag{7.22}$$

7.3 Stability Analysis and Simple Controller Design

The essential properties of a closed loop control system include closed loop eigenvalues to analyse stability, bode plot from reference input to output, step response to unit step at the reference input, and response to torque disturbances.

Design of closed loop control systems can be achieved in several elegant ways, including optimal and robust control methods. Even with a limited background in control theory, one can design well performing controllers using the stepwise procedure below.

Algorithm 1 *Stepwise procedure for simple control system design*

Commands are Matlab® control systems toolbox and the supplied functions for the course.

1. *Determine the linear model without wheel dynamics using*
   [satmod_c, satmod_dist] = lin_sat_wheels_mod(Is, ws, hw)
2. *Eigenvalues in the open loop (no control) system using*
   damp(satmod_c)
   pzmap(satmod_c)
3. *Get the* $\mathbf{A}, \mathbf{B}_u, \mathbf{C}, \mathbf{D}$ *matrices using*
   [$\mathbf{A}, \mathbf{B}_u, \mathbf{C}, \mathbf{D}$*] = ssdata(satmod_c)*
4. *Get the* $\mathbf{A}, \mathbf{B}_{dist}, \mathbf{C}, \mathbf{D}$ *matrices using*
   [$\mathbf{A}, \mathbf{B}_d, \mathbf{C}, \mathbf{D}$*] = ssdata(satmod_dist)*
5. *Define the closed loop by*
   $\mathbf{D}_{cl}$ *= zeros(size(*$\mathbf{C}$*,1),size(*$\mathbf{K}_r$*,2))*
   [satmod_c_closed] = ss(($\mathbf{A} - \mathbf{B}_u\mathbf{K}_y\mathbf{C},\ \mathbf{B}_u\mathbf{K}_r,\ \mathbf{C}_{cl},\ \mathbf{D}_{cl}$*))*
6. *Investigate eigenvalues of the closed loop by*
   damp(satmod_c_closed)
   pzmap(satmod_c_closed) and
   [P,Z]=pzmap(satmod_c_closed)
7. *Once a suitable stable controller has been found, determine frequency response properties by*
   bode(satmod_c_closed)
8. *and step responses by*
   step(satmod_c_closed)

9. *Investigate disturbance rejection by*
   [satmod_dist_closed] = ss(($\mathbf{A} - \mathbf{B}_u\mathbf{K}_y\mathbf{C},\ \mathbf{B}_d,\ \mathbf{C},\ \mathbf{D}$*))*
   bode(ssatmod_dist_closed)
   step(satmod_dist_closed)

10. *iterate in the above procedure until specifications are met.*

Example 3 *Rømer-like Satellite*
Let the Rømer-like satellite have

$$\boldsymbol{\omega} = (0,0,0)^T[rad/s]; \quad h_i = [0,0.5,0.2]^T[Nms]; \tag{7.23}$$

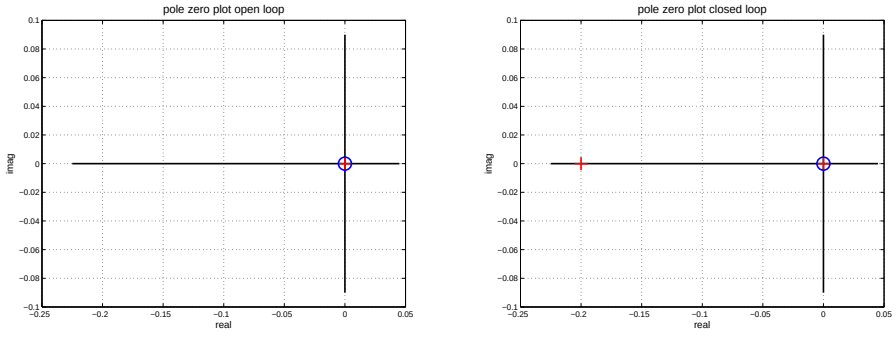
$$\mathbf{I}_s = \begin{bmatrix} 14.3 & 0 & 0 \\ 0 & 13.6 & 0 \\ 0 & 0 & 4.6 \end{bmatrix} [kgm^2] \tag{7.24}$$

Define the desired bandwidth of the closed-loop system by $\omega_{bw\_cls} = 0.2$ *[rad/s]. A simple design approach is to select gains to be*

$$\begin{aligned} k_p(j) &= \mathbf{I}_s(j,j)\omega_{bw\_cls}^2 \\ k_{pd}(j) &= 2\mathbf{I}_s(j,j)\omega_{bw\_cls} \end{aligned} \tag{7.25}$$

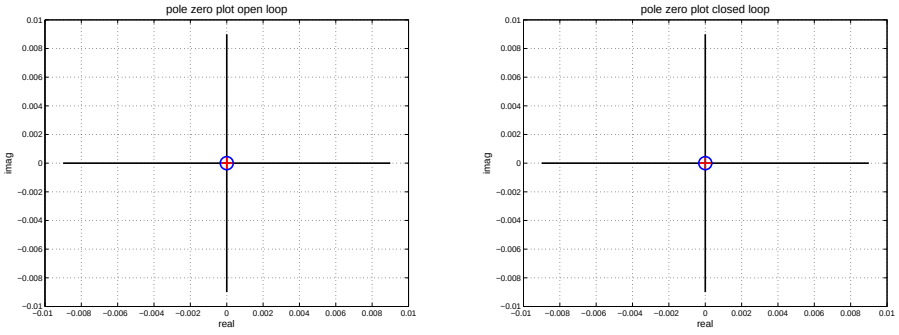
$$\begin{aligned} \mathbf{K}_p &= diag(k_p(1), k_p(2), k_p(3)) \\ \mathbf{K}_{pd} &= diag(k_{pd}(1), k_{pd}(2), k_{pd}(3)) \end{aligned} \tag{7.26}$$

We investigate the properties of this simplistic design idea by using the steps of algorithm 1. The gain factors used are the above given $\mathbf{K}_p$ *and* $\mathbf{K}_{pd}$*. Combinations of two sets of nominal values are used,* $\{\omega_s = [0;0;0]$*,* $\omega_s = [0;-0.003;0]$ $\}$*and* $\{\mathbf{h}_s = [0;0;0]$*,*$\mathbf{h}_s = [0;0;0.5]$ $\}$*The results are shown in Figures 7.3, 7.5 and 7.7.*



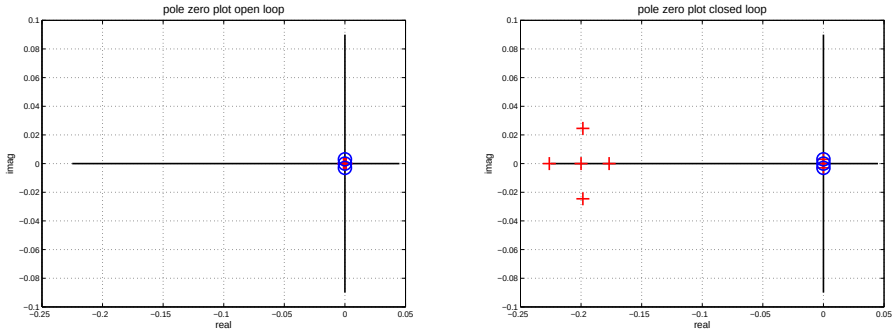
(a) open loop (b) closed loop

Figure 7.3: R loop (b). Angular velocity is $\omega_s = [0; 0; 0]$ rad/s and angular momentum is $\mathbf{h}_s = [0; 0; 0]$ Nms. The closed loop eigenvalues are all in -0.2 as intended. The small imaginary part is due to numerics. Three eigenvalues at the origin are cancelled by three zeros. The origin of this is the physical integration from $\dot{h}$ to h.



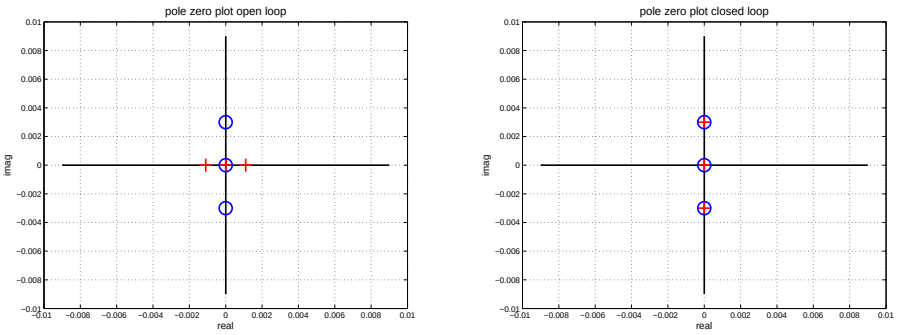
(a) open loop (b) closed loop

Figure 7.4: Same as Figure 7.3 but zoomed



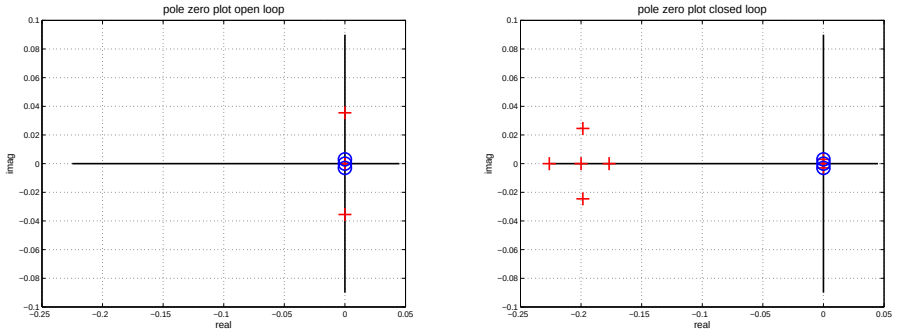
(a) open loop (b) closed loop

Figure 7.5: Roemer-like satellite pole-zero plot for open (a) and closed loop (b). Angular velocity is $\omega_s = [0; -0.003; 0]$ rad/s and angular momentum $\mathbf{h}_s = [0; 0; 0.5]$ Nms. The angular velocity around the y-axis, and the moment of inertia with $I_x < I_y < I_z$, cause the open loop system to be unstable. The closed loop is stable with closed loop poles only slightly affected by the perturbation of the open loop system.



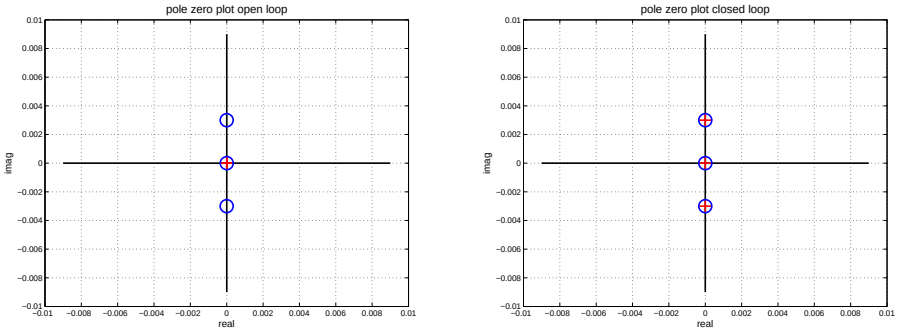
(a) open loop (b) closed loop

Figure 7.6: Same as Figure 7.5 but zoomed



(a) open loop (b) closed loop

Figure 7.7: Roemer-like satellite pole-zero plot for open (a) and closed loop (b). Angular velocity is $\omega_s = [0; -0.003; 0]$ rad/s and angular momentum $\mathbf{h}_s = [0; 0; 0.5]$ Nms. The bias in angular momentum has moved the open loop pole-zero pattern significantly. Despite the large perturbation, the closed loop poles show that the closed loop system is still nicely damped.



(a) open loop (b) closed loop

Figure 7.8: Same as Figure 7.7 but zoomed

Bibliography

[1] James A. Wertz (ed). *Spacecraft Attitude Determination and Control*. Kluwer Academic Publishers, 7th. reprint edition, 1984.

[2] Peter C. Huges. *Spacecraft Attitude Dynamics*. John Wiley and Sons, 1986.

[3] Marshall H. Kaplan. *Modern Spacecraft Dynamics and Control*. John Wiley and Sons, 1976.

[4] Jack B. Kuipers. *Quaternions and Rotation Sequences*. Princeton University Pres, 2002.

[5] Marcel J. Sidi. *Spacecraft Dynamics and Control - A Practical Engineering Approach*. Cambridge University Press, 2001.

[6] Sigurd Skogestad and Ian Postlethwaite. *Multivatiable Feedback Control: Analysis and Design*. Wiley, 2005.

[7] Bong Wie. *Space Vehicle Dynamics and Control (2^{nd} edition)*. American Institute of Aeronautics and Astronautics, 2008.

Appendix A

Programme Listings and Contents CD

The CD enclosed with the printed version of this report comprise the functions listed below. The files are available for participants in DTU course 31365: Spacecraft Dynamics and Control via file-sharing at the DTU Campusnet.

- **linSatModel.m**. Calculate A,B,C,D matrices of linear model.
- **roemerEx.m**. Script file that runs the Roemer-like satellite design example.
- **SATsimstart.m**. Script to set parameters used in the SIMULINK simulation.
- **RWAsim.m**. Script to set reaction wheel assembly parameters used in the SIMULINK simulation.
- **SATsim.m**. Script file to set satellite main body parameters in the simulation.
- **sat3sim.mdl**. SIMULINK model with nonlinear simulation of satellite with reaction wheel assembly as actuator. Gravity gradient is not included in this version.

A.1 linSatModNogg

```
function    [satmod_c, satmod_dist] = lin_sat_wheels_mod(Is, ws, hw,
tau_w, Ts)
%
%   procedure to get system matrices of satellite with
momentum wheels
%   and no gravity gradient
%
%   Terms:
%   SBC Satellite
Body Coordinate system
%
%   (C) copyright   Mogens Blanke 2001-2005
%
 version 2.01  November 2005
%
% \qquad input:
%   Is      satellite inertia
matrix in SBC     real(3,3)
%   ws      satellite's steaty state angular
velocity (SBC frame)  real(3,1)
%   hw      wheel momentum in SBC frame
    real(3,1)
%
%   tau_w   wheel loop timeconstant           scalar(1,1)
[s]
%           Include w_bw if you want to calculate the model with wheel
%
          dynamics included
%   Ts      optional sampling time to get a
discrete model
%   \qquad\qquad\qquad\qquad the discrete model is calculated
if the Ts parameter is specified
%
%   The last two parameters can be ommitted
from the call.
%
% \qquad output:
% \qquad satmod_c  statespace model
(Matlab LTI format):
% \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad
dx(t)/dt =\qquad A*x(t) + B_u*N_c(t)
% \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad
```

```
y(t) \qquad\qquad=\qquad C*x(t) + D_u*N_c(t) \qquad\qquad
%
% \qquad satmod_c
%
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad dx(t)/dt =\qquad A*x(t)
+ B_d*N_d(t)
% \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad y(t) \qquad\qquad=\qquad
C*x(t) + D_d*N_d(t) \qquad\qquad
%
% where N_c is control torque demand
to reaction wheel(s)
% and \qquad N_d is external disturbance torque on
satellite


Aw(:,1) = [0, Is(3,1), Is(2,1); ...
     -2*Is(3,1), -Is(3,2),
-Is(3,3)+Is(1,1);...
     2*Is(2,1), Is(2,2)-Is(1,1), Is(2,3)]*ws;

Aw(:,2)
= [Is(3,1), 2*Is(3,2), (Is(3,3)-Is(2,2));...
     -Is(3,2), 0, Is(1,2);...

    Is(2,2)-Is(1,1), 2*Is(1,2),-Is(1,3)]*ws;

Aw(:,3)= [-Is(2,1),Is(3,3)-Is(2,2),-2*Is(2,3);...

    Is(1,1)-Is(3,3), Is(1,2), 2*Is(1,3);...
     Is(2,3), -Is(1,3), 0]*ws
;

invrtIs = inv(Is);

Ahw  = [0, -hw(3), hw(2);...
      hw(3), 0, -hw(1);
...
     -hw(2), hw(1), 0];

Awh = [0, ws(3), -ws(2);...
     -ws(3), 0,
ws(1); ...
     ws(2), -ws(1), 0];
```

```
Aww = Aw + Ahw;

A  =  [invrtIs*Aww,
zeros(3,3), invrtIs*Awh;...
     0.5*eye(3,3), zeros(3,3), zeros(3,3);...

      zeros(3,3), zeros(3,3), zeros(3,3)];

Bu = [invrtIs; zeros(3,3); -eye(3,3)];
Bd = [invrtIs; zeros(3,3);
zeros(3,3)];

C  = [eye(6,6), zeros(6,3)]; D
 = zeros(6,3)

satmod_c = ss(A,Bu,C,D); satmod_dist = ss(A,Bd,C,D);

%
  case with wheel dynamics included (12 states in state vector)
if (nargin
== 4 | nargin == 5)
   A  =  [invrtIs*Aww, zeros(3,3), invrtIs*Awh, -invrtIs;...

    0.5*eye(3,3), zeros(3,3), zeros(3,3), zeros(3,3);...
     zeros(3,3),
zeros(3,3), zeros(3,3), eye(3,3);...
     zeros(3,3), zeros(3,3), zeros(3,3),
-eye(3,3)*1/tau_w];

Bu = [zeros(3,3); zeros(3,3); zeros(3,3); -eye(3,3)*1/tau_w];
Bd =
[invrtIs; zeros(3,3); zeros(3,3); zeros(3,3)];

C  = [eye(6,6), zeros(6,6)];
D  = zeros(6,3);

satmod_c = ss(A,Bu,C,D); satmod_dist = ss(A,Bd,C,D);

end

if
(nargin == 5)
  satmod_c = SS(A, Bu, C, D, Ts);
end
```

A.2 romerEx

```
function romer_ex
%   roemer-like example
%   MB 9. nov 2009
%

Is = [14.6 0 0; 0 13.6 0; 0 0 4.6];
hs0 = [0; 0; 0];
hs1 = [0; 0; 0.5];
ws0 = [0; 0; 0];
ws1 = [0; -0.0030; 0];

%
% parameters for plot
re_min = -0.25; re_max = 0.05; im_min=-0.1; im_max = 0.1;
re_axis = 0.9*[re_min, re_max];
im_axis = 0.9*[im_min, im_max];
re_min_zoom = -0.01; re_max_zoom = 0.01;
im_min_zoom=-0.01; im_max_zoom = 0.01;
re_axis_zoom = 0.9*[re_min_zoom, re_max_zoom];
im_axis_zoom = 0.9*[im_min_zoom, im_max_zoom];

limplotvec = [re_min re_max im_min im_max];
limplotveczoom = [re_min_zoom re_max_zoom im_min_zoom im_max_zoom];

%    run open and closed loop for ang. velocity wsy and zero angular
%    momentum
[satmod_Nw2y_open, satmod_Nd2y_open] = lin_sat_wheels_mod(Is, ws0, hs0);
[P,Z] = pzmap(satmod_Nw2y_open)
figure(1)
pole_zero_plot(P,Z,re_axis, im_axis, limplotvec, ...
    'pole zero plot open loop', 'romex_ws0_hs0_open.eps')
pole_zero_plot(P,Z,re_axis_zoom, im_axis_zoom,limplotveczoom, ...
    'pole zero plot open loop', 'romex_ws0_hs0_open_zoom.eps')


[A,B,C,D] = ssdata(satmod_Nw2y_open);
om_bw = 0.2;
Kpd = 2*Is*om_bw;
Kp = Is*om_bw^2;
Ky = [Kpd, 2*Kp];
Kr = -2*[Kp];
D_cl = zeros(size(C,1),size(Kr,2));
```

```
satmod_ref2y_closed = ss(A-B*Ky*C, B*Kr, C, D_cl );
[P,Z] = pzmap(satmod_ref2y_closed)
figure(2)
pole_zero_plot(P,Z,re_axis, im_axis, limplotvec, ...
    'pole zero plot closed loop', 'romex_ws0_hs0_closed.eps')
pole_zero_plot(P,Z,re_axis_zoom, im_axis_zoom,limplotveczoom, ...
    'pole zero plot closed loop', 'romex_ws0_hs0_closed_zoom.eps')

%    run open and closed loop for ang. velocity wsy and zero angular
%    momentum
[satmod_Nw2y_open, satmod_Nd2y_open] = lin_sat_wheels_mod(Is, ws1, hs0);
[P,Z] = pzmap(satmod_Nw2y_open)
figure(3)
pole_zero_plot(P,Z,re_axis, im_axis, limplotvec, ...
    'pole zero plot open loop', 'romex_ws1_hs0_open.eps')
pole_zero_plot(P,Z,re_axis_zoom, im_axis_zoom,limplotveczoom, ...
    'pole zero plot open loop', 'romex_ws1_hs0_open_zoom.eps')

[A,B,C,D] = ssdata(satmod_Nw2y_open);
om_bw = 0.2;
Kpd = 2*Is*om_bw;
Kp = Is*om_bw^2;
Ky = [Kpd, 2*Kp];
Kr = -2*[Kp];
D_cl = zeros(size(C,1),size(Kr,2));
satmod_ref2y_closed = ss(A-B*Ky*C, B*Kr, C, D_cl );
[P,Z] = pzmap(satmod_ref2y_closed)
figure(4)
pole_zero_plot(P,Z,re_axis, im_axis, limplotvec, ...
    'pole zero plot closed loop', 'romex_ws1_hs0_closed.eps')
pole_zero_plot(P,Z,re_axis_zoom, im_axis_zoom,limplotveczoom, ...
    'pole zero plot closed loop', 'romex_ws1_hs0_closed_zoom.eps')

%   run the example for open and closed loop with wsy and hs1 (angular
%   momentum bias)
[satmod_Nw2y_open, satmod_Nd2y_open] = lin_sat_wheels_mod(Is, ws1, hs1);
[P,Z] = pzmap(satmod_Nw2y_open)
figure(5)
pole_zero_plot(P,Z,re_axis, im_axis, limplotvec, ...
    'pole zero plot open loop', 'romex_ws1_hs1_open.eps')
pole_zero_plot(P,Z,re_axis_zoom, im_axis_zoom,limplotveczoom, ...
    'pole zero plot open loop', 'romex_ws1_hs1_open_zoom.eps')

om_bw = 0.2;
```

```
Kpd = 2*Is*om_bw;
Kp = Is*om_bw^2;
Ky = [Kpd, 2*Kp];
Kr = -2*[Kp];
D_cl = zeros(size(C,1),size(Kr,2));
satmod_ref2y_closed = ss(A-B*Ky*C, B*Kr, C, D_cl );
[P,Z] = pzmap(satmod_ref2y_closed)
figure(6)
pole_zero_plot(P,Z,re_axis, im_axis, limplotvec, ...
    'pole zero plot closed loop', 'romex_ws1_hs1_closed.eps')
pole_zero_plot(P,Z,re_axis_zoom, im_axis_zoom,limplotveczoom, ...
    'pole zero plot closed loop', 'romex_ws1_hs1_closed_zoom.eps')

end

function pole_zero_plot(P, Z, re_axis, im_axis, limplotvec, plottitle, saveplotas)

zeroataxis = [0, 0];
plot( re_axis,zeroataxis,'k-',zeroataxis,im_axis,'k-','LineWidth',2), hold on
plot(real(P),imag(P),'r+',real(Z),imag(Z),'bo','LineWidth',2,'MarkerSize',16), grid
axis(limplotvec); hold off
title(plottitle,'FontSize',14),
xlabel('real','FontSize',12), ylabel('imag','FontSize',12)
eval(['print -depsc2 -tiff ',saveplotas])
end
```

A.3 SATsimstart

```
% script file to test Simulink satellite simulation
%
 (C) copyright  Mogens Blanke 2001-2003
%   version 03a  December 2003
%
global
fpc_gain;
global fpc_flag;
global fpc_param;
global fpc_state;
global time;

%
initialize all simulation parameters
SAT_sim; RWA_sim;

%   FPC_sim;
%
 AD_sim;
%   start simulation % in front of the line you don't want !

sat3_sim
```

A.4 RWAsim

```
%   RWAsim
%   This script initializes your wheel model.
%
 It enables different number of wheels and orientations by
%  selecting
the parameters Awar, w-w_ini and N_dw.
%  Number of columns in Awar MUST
equal number of wheels
%  Wheel initial velocity is specified in w_w_ini
%
 N_dw specifies the torque demand of each wheel
%  n_w specifies the number
of wheels
%
%  Note: # columns in Awar == dim(w_w_ini) == dim(N_dw) ==
n_w
```

```
%
%   (C) Copyright 2001-2003  Mogens Blanke, DTU
%   version 03a  December
2003
%
%   Parameters for each wheel

wheel_inertia = 0.0010;% kg m^2
tau_w
 = 0.25;
wheel_loop_bandwidth = 1/tau_w; % rad/s
wheel_infspeed = 500;%
rad/s
wheel_inftorque = 0.050; %max torque on wheel
wheel_kqn = wheel_loop_bandwidth
* wheel_inertia;

%   Example 1: 4 wheels in tetraeder orientation
n_w =
4;    % number of wheels
a = sqrt(1/3); b =sqrt(2/3);
Awar = [a b 0; a -b
0; -a 0 -b; -a 0 b]';
w_w_ini = [100, 100, 100, 100];

% Example 2: 3 wheels
in orthogonal orientation
n_w = 3; a = 1;
Awar = [a 0 0; 0 a 0; 0 0 a ]';
w_w_ini
= [100, 100, 100];

% Example 2: 1 wheel in direction 45 deg of satellite's
x-y plane
n_w = 1;
a = sqrt(1/2);
Awar = [a a 0]';
w_w_ini = [100];

%
 Remember to put % in front of lines for the cases you don't use
```

A.5 SATsim

```
%   SATsim
%   script to initialize satellite main body
parameters

Iscb = diag([3,14,15]);
Tsamp = 1.0;
```