Real time hardware-in-loop simulation of ESMO satellite attitude control system

Rune Finnset[*], Sudhakara K. Rao[†] and Jøran Antonsen[‡]

Department of Computer Science, Electrical Engineering and Space Technology
Narvik University College
N-8515 NARVIK
NORWAY

Abstract

This paper studies attitude control of the ESMO satellite using six reaction thrusters. Bang-bang control with dead-zone and Pulse-Width Modulation (PWM) for the modulation of the on-time of the thrusters are treated. Closed loop hardware-in-loop simulations, using the microcontroller unit (MCU) Microchip PIC18F452 for implementation of attitude control and MatLab in a standard PC for simulating satellite dynamics, are carried out. Results for real time simulation are compared with autonomous simulations. The controller gives a satisfactory performance in the real time environment.

1 Introduction

1.1 Background

The Student Space Exploration and Technology Initiative (SSETI) is a project initiated in the year 2000 by the European Space Agency (ESA). The goal of this project is to increase young Europeans´ interest in technology and space related activities. Students from several European countries are collaborating in the design and development of micro satellites. The work presented in this article is a study of mission two, the European Student Moon Orbiter (ESMO) Attitude Control system (ACS) and is based on [1]. For more information about the SSETI project, see http://www.sseti.net.

The ESMO mission is in its early stages and no final structure or configuration is chosen; hence some assumptions are made in this paper. The ESMO satellite is assumed to have dimensions of $60 \times 60 \times 80$ cm and a mass of approximately 120 kg. For attitude control ESMO uses six reaction control thrusters and possibly four/three reaction wheels. However, in this paper only the reaction control thrusters for attitude control are used.

1.2 Previous ACS work

Several other students have lately performed studies of the attitude control system for the ESMO satellite. In [2] a study of reaction wheels arranged in a tetrahedron can be found, which also emphesises on a Failure Detection and Isolation (FDI) scheme. A momentum dumping scheme for the ESMO satellite can be found in [3] where a study of disturbance torques for the ESMO mission is also performed.

2 MODELING

2.1 Reference frames

To analyze the motion of a satellite, it is necessary to define reference frames to which the motion is related. Utilized reference frames are given as

Earth-Centered Inertial (ECI) Reference Frame This frame is denoted $\mathcal{F}_i$, and has its origin located in the center of the earth. Its z-axis is directed along the rotation axis of the earth towards the celestial north pole, the x-axis is directed towards the vernal equinox, and finally the direction of the y-axis completes a right handed orthogonal frame.

[*]M.Sc, email: rfinnset@home.no
[†]Visiting Professor, email: rao@hin.no
[‡]M.Sc, email: ja@hin.no

Orbit Reference Frame The orbit frame, denoted $\mathcal{F}_o$, has its origin located in the center of mass of the satellite. The *z*-axis is pointing towards the center of the earth, and the *x*-axis is directed forward in the traveling direction of the satellite, tangentially to the orbit for a circular orbit. Assuming near circular orbit, the orbit frame rotates relative to the ECI frame with an angular velocity of approximately

$$\omega_o \approx \sqrt{\frac{\mu_g}{r_c^3}}$$

where μ_g is the earth's gravitational coefficient, and r_c is the distance from the frame origin to the center of the earth. Rotation about the *x*, *y* and *z*-axis is named roll, pitch and yaw, respectively.

Body Reference Frame Its origin is located in the satellite center of mass, and the axes are locked to the satellite and coincide with the principal axes of inertia. The frame is denoted $\mathcal{F}_b$.

2.2 Kinematics

Rotation between the previously described reference frames is done by rotation matrices, members of the special orthogonal group of order three, [4] i.e.

$$SO(3) = \{\mathbf{R} | \mathbf{R} \in \mathbb{R}^{3\times 3},\ \mathbf{R}^T\mathbf{R} = \mathbf{I},\ \det \mathbf{R} = 1\}$$

where $\mathbf{I}$ is the 3×3 identity matrix. A rotation matrix for a rotation θ about an arbitrary unit vector $\mathbf{k}$ can be angle-axis parameterized as

$$\mathbf{R}_{k,\theta} = \mathbf{I} + \mathbf{S}(\mathbf{k})\sin\theta + \mathbf{S}^2(\mathbf{k})(1-\cos\theta) \tag{1}$$

where $\mathbf{S}(\cdot)$ is the cross product operator given by

$$\mathbf{S}(\mathbf{k}) = \mathbf{k}\times$$

and coordinate transformation of a vector $\mathbf{r}$ from frame *a* to frame *b* is written as $\mathbf{r}^b = \mathbf{R}_a^b\mathbf{r}^a$. In general, the rotation matrix describing rotations from the orbit frame to the body frame can be described by

$$\mathbf{R}_o^b = [\mathbf{c}_1\ \mathbf{c}_2\ \mathbf{c}_3] \tag{2}$$

where the elements $\mathbf{c}_i$ are the directional cosines. The time derivative of a matrix $\mathbf{R}_a^b$ can be expressed as [4]

$$\dot{\mathbf{R}}_b^a = \mathbf{S}(\boldsymbol{\omega}_{ab}^a)\mathbf{R}_b^a = \mathbf{R}_b^a\mathbf{S}\left(\boldsymbol{\omega}_{ab}^b\right) \tag{3}$$

where $\boldsymbol{\omega}_{ab}^b$ is the angular velocity of frame *b* relative to frame *a* represented in frame *b*. The rotation matrix in (1) may be expressed by the Euler parameter representation as

$$\mathbf{R}_{\eta,\varepsilon} = \mathbf{I} + 2\eta\mathbf{S}(\boldsymbol{\varepsilon}) + 2\mathbf{S}^2(\boldsymbol{\varepsilon})$$

where

$$\eta = \cos(\theta/2) \in \mathbb{R} \tag{4}$$

$$\boldsymbol{\varepsilon} = \mathbf{k}\sin(\theta/2) \in \mathbb{R}^3 \tag{5}$$

are the Euler parameters, which satisfy the constraint

$$\eta^2 + \boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon} = 1$$

The kinematic differential equations can be found from (3) together with (4) and (5) as

$$\dot{\eta} = -\frac{1}{2}\boldsymbol{\varepsilon}^T\boldsymbol{\omega}_{ob}^b$$

$$\dot{\boldsymbol{\varepsilon}} = \frac{1}{2}\left[\eta\mathbf{I} + \mathbf{S}(\boldsymbol{\varepsilon})\right]\boldsymbol{\omega}_{ob}^b$$

The deviation between the current attitude $[\eta\ \boldsymbol{\varepsilon}]^T$ and the desired attitude $[\eta_d\ \boldsymbol{\varepsilon}_d]^T$ is given by the Euler product [4] as

$$\begin{bmatrix} \tilde{\eta} \\ \tilde{\boldsymbol{\varepsilon}} \end{bmatrix} = \begin{bmatrix} \eta_d \\ \boldsymbol{\varepsilon}_d \end{bmatrix} \otimes \begin{bmatrix} \eta \\ -\boldsymbol{\varepsilon} \end{bmatrix} = \begin{bmatrix} \eta_d\eta + \boldsymbol{\varepsilon}_d^T\boldsymbol{\varepsilon} \\ \eta_d\boldsymbol{\varepsilon} - \eta\boldsymbol{\varepsilon}_d - \mathbf{S}(\boldsymbol{\varepsilon}_d)\boldsymbol{\varepsilon} \end{bmatrix}$$

2.3 Dynamics

Assuming rigid body movement, the dynamic model of a satellite can be found from Euler's moment equation as [5]

$$\mathbf{J}\dot{\boldsymbol{\omega}}_{ib}^b = -\boldsymbol{\omega}_{ib}^b \times (\mathbf{J}\boldsymbol{\omega}_{ib}^b) + \boldsymbol{\tau}_d^b + \boldsymbol{\tau}_a^b$$

where $\mathbf{J}$ is the satellite inertia matrix, $\boldsymbol{\omega}_{ib}^b$ is the angular velocity of the satellite body frame relative to the inertial frame, $\boldsymbol{\tau}_d^b$ is the total disturbance torque, and $\boldsymbol{\tau}_a^b$ is the actuator torque, all expressed in the body frame. The angular velocity of the satellite body frame relative to the orbit frame, expressed in the body frame $\boldsymbol{\omega}_{ob}^b$, is given by

$$\boldsymbol{\omega}_{ob}^b = \boldsymbol{\omega}_{ib}^b + \omega_o\mathbf{c}_2 \tag{6}$$

where $\mathbf{c}_2$ is the directional cosine vector from (2).

2.4 Disturbance torques

The satellite will be exposed to different internal and external disturbance torques. Internal torques might come from electromagnetic torques and fuel sloshing. The gravity gradient, atmospheric drag, solar radiation, and solar wind are external forces producing disturbance torques. The disturbance torques are neglected in the following analysis, except for the gravity gradient torque, expressed as

$$\boldsymbol{\tau}_g^b = 3\omega_o^2\mathbf{c}_3 \times (\mathbf{J}\mathbf{c}_3)$$

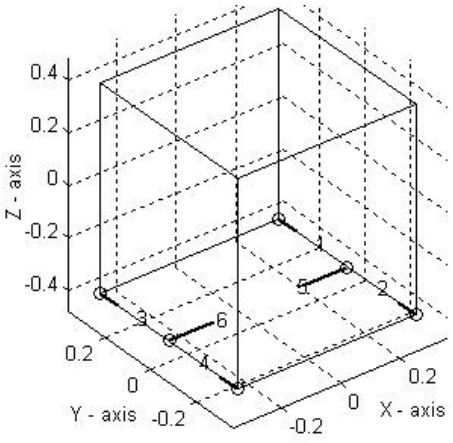

Figure 1: Thruster locations.

2.5 Actuator dynamics

In this paper ESMO is assumed to have six reaction thrusters, and the control torque from the thrusters may be expressed as

$$\boldsymbol{\tau}_a^b = \boldsymbol{\tau}_t^b = \mathbf{B}_a \mathbf{u}$$

$$\mathbf{u} = \begin{bmatrix} F_1 & F_2 & F_3 & F_4 & F_5 & F_6 \end{bmatrix}^T$$

where $\mathbf{u}$ is the vector of actuators, F_i is the magnitude of thrust from the i'th thruster. Thruster locations for ESMO is given in Figure 1. In this case $\mathbf{B}_a$ can be expressed as

$$\mathbf{B}_a = \begin{bmatrix} -r_z & r_z & -r_z & r_z & 0 & 0 \\ 0 & 0 & 0 & 0 & r_z & -r_z \\ -r_x & r_x & r_x & -r_x & 0 & 0 \end{bmatrix}$$

where $r_j,\ j = x,y,z$, are the components of the common thruster distance from the satellite center of mass, due to symmetric placement of the thrusters. To find the desired input actuator vector $\mathbf{u}$, the Moore-Penrose pseudoinverse, as found in [6], is applicable. The actuator vector $\mathbf{u}$ can be computed as

$$\mathbf{u} = \mathbf{B}_a^{\dagger} \boldsymbol{\tau}_a^b$$

where $\mathbf{B}_a^{\dagger}$ is the pseudoinverse given as

$$\mathbf{B}_a^{\dagger} = \mathbf{B}_a^T \left(\mathbf{B}_a \mathbf{B}_a^T \right)^{-1}$$

which in this case of the satellites actuator combination can be shown to satisfy

$$\mathbf{B}_a \mathbf{B}_a^{\dagger} = \mathbf{I}$$

3 Controller design

3.1 Linear PD controller

By linearizing the dynamic model, a linearized state space model can be written as

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$

where the state vector is

$$\mathbf{x} = \begin{bmatrix} \varepsilon_1 & \dot{\varepsilon_1} & \varepsilon_2 & \dot{\varepsilon_2} & \varepsilon_3 & \dot{\varepsilon_3} \end{bmatrix}^T$$

The system matrix $\mathbf{A}$ and actuator matrix $\mathbf{B}$ are found by simple linearization method. A simple PD controller may be deduced from the linearized model as

$$\mathbf{u} = -\mathbf{K}_p \tilde{\boldsymbol{\varepsilon}} - \mathbf{K}_d \dot{\boldsymbol{\varepsilon}}$$

where $\mathbf{K}_p$ and $\mathbf{K}_d$ are the state feedback calculated from desired pole placement for the system. The resulting actuator torque becomes

$$\boldsymbol{\tau}_a^b = \mathbf{B}_a(-\mathbf{K}_p \tilde{\boldsymbol{\varepsilon}} - \mathbf{K}_d \dot{\boldsymbol{\varepsilon}})$$

Further details of the PD controller can be found in [1].

3.2 Bang-bang controller with dead-zone

This is a simple algorithm for activating the thrusters. Thrusters are not activated until a certain torque limit (or error angle) is reached. When the required control torque for one of the axes exceeds the lower or upper limit, the relevant thrusters are activated. The dead-zone parameter is given as a percentage. A dead-zone of 30 % means that when the required control torque, about one axis exceeds 30 % of the maximum torque that all the thrusters may provide about it, then these thrusters are activated. When using the error angle scheme, the dead-zone is a certain angle. If the current and desired attitude differ by a small angle (a few degrees), the thrusters are not activated, thereby saving fuel. There is a weighting between fuel consumption and accuracy and stability of the attitude control. Higher dead-zone conserves fuel, but the attitude error will be larger. Figure 2 shows the logic used in this control scheme [7]. Since Euler parameters are used instead of Euler angles, the control logic used is slightly different. A Bang-bang feedback controller is implemented through a PD feedback method. The output of the controller is the error function [8]

$$\mathbf{e} = K_P \boldsymbol{\varepsilon} + K_D \boldsymbol{\omega}_{bo}^b$$

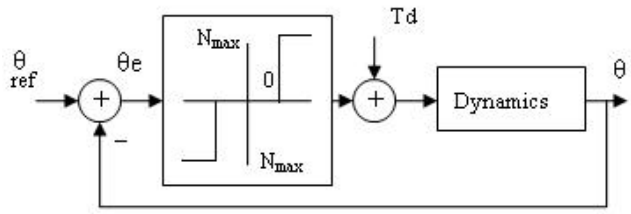

Figure 2: Bang-bang with dead-zone control

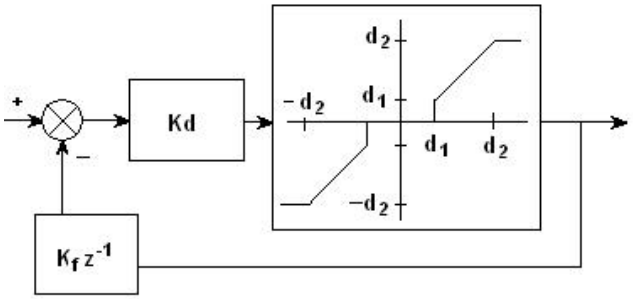

Figure 3: Pulse-Width modulation.

where **e** is the control error and K_P and K_D are the usual controller gains. The two-level switching logic is then

$$\mathbf{T}_t^B = \begin{cases} -\mathbf{N}_{max}, & \mathbf{e} \geq \mathbf{e}_{band} \text{ and } \mathbf{e} \text{ increasing} \\ \mathbf{N}_{max}, & \mathbf{e} \geq \mathbf{e}_{band} \text{ and } \mathbf{e} \text{ decreasing} \\ \mathbf{0}, & -\mathbf{e}_{band} < \mathbf{e} < \mathbf{e}_{band} \end{cases}$$

where $\mathbf{e}_{band}$ is the dead-zone, and $\mathbf{T}_t^B$ is the applied torque. The error function consists of three components, thus each of the three axes may be treated independently. The slope of the switching function, magnitude of the error quaternion and stability of nonlinear limit cycle are controlled by the controller gains. Thus the tuning of these parameters is required [8].

3.3 Pulse-Width Modulation (PWM)

Pulse modulation techniques are used to modulate the activation time of the thrusters. The purpose of the modulation is to make the average total torque exerted by the thrusters approximate the calculated control torque to the highest possible extent. In pulse-width modulation, the on-time of the thruster is increased or decreased depending on the error. Duty cycle, defined as the ratio of on-time to sampling time, is thus varied according to the error. Even though thrusters exhibit nonlinear on-off behavior, an almost linear duty cycle may be achieved. Most commonly a single or two thrusters are used to achieve rotation about one axis. The logic of the PW modulator is shown in Figure 3. The parameter d_1 represents the minimum duty cycle of the thrusters, and will be directly proportional to the attitude dead-zone. The other parameter (d_2) is the maximum pulse width, and is equal to the MCU sampling period. Designers may use the feedback gain (K_f) to obtain desirable system properties. Activation time length, t_{on}, and thruster torque, T_{th}, may be given as [9]

$$t_{on} = \begin{cases} t_s, & T_{th} > T_{max} \\ \frac{T_C}{T_{max}} t_s, & T_{min} \leq T_{th} \leq T_{max} \\ 0, & T_{th} < T_{min} \end{cases}$$

$$T_{th} = \begin{cases} T_{max}, & T_C > T_{max} \\ T_C, & T_{min} \leq T_C \leq T_{max} \\ 0, & T_C < T_{min} \end{cases}$$

where t_s is the system sampling period, T_C is the control torque, and T_{max} and T_{min} are the maximum and minimum torques which may be obtained. As seen in Figure 3, there is a dead-zone in the duty cycle of the thrusters. If the calculated duty cycle is smaller than this, then it is set to zero, and the thruster is thus not activated for the current sampling period (0 % duty cycle). Highest average torque is obtained if the thruster is activated for the full duration of the sampling period (100 % duty cycle).

4 Hardware-in-loop simulation

4.1 Introduction

Microcontroller units are used in a variety of applications, for example as controllers in a system. System variables are measured and fed to the MCU, on either digital or analog format. The measurements are used in an algorithm to generate necessary control signals. These are then transmitted to the actuators of the systems, either by using serial communication or by generating analog voltages through D/A converters. In this paper a MCU is used to implement an attitude control algorithm for ESMO. A computer simulates the attitude dynamics of the satellite. The computer integrates the dynamics equations and sends ”measurements” of these to the MCU using serial communication. Actuator signals from the MCU are then sent back to the computer. This forms a closed loop. Closed loop simulations can be made to work in real time to simulate the dynamic behavior of the satellite system with the ACS to control its attitude. The real time hardware-in-loop simulation have been carried out for ESMO using the reaction control system.

4.2 Requirements

Communication between the computer and MCU takes place in the beginning of every sample period (100 ms) of the system. Thus both the computer and the MCU must be able to complete necessary computations in less than this time. Since the given sample time is quite large, both the MCU and computer fulfill this requirement. They rely on serial communication for transmission of bytes, representing state variables and actuator signals. An error handling algorithm for corrupted communication, either missing bytes or transmission errors, should be included in a more complete system. Such an algorithm is not implemented here.

The software implementation is done in Microchip MPLAB IDE (integrated development environment), which is a windows program for development, compilation and testing of programs. The MPLAB C18 compiler is used to compile the code into hex code, which may be downloaded to the MCU. Both high level (c-code) and low level (assembly) programming techniques are available. In this paper high level programming is used exclusively. The interface between the computer and MCU is handled by an In-Circuit Debugger (ICD), attached to the computer using an USB cable. The ICD is used to program the MCU, signal it to start its program, reset it and other important functions.

4.3 Communication

The computer and the MCU use the standard RS232 protocol to send and receive data during simulation. These devices both have an Universal Synchronous Asynchronous Receiver Transmitter (USART). In this paper the asynchronous mode is used exclusively. In asynchronous transmission one byte (8 bits) is sent at a time. The line lies at logic high level when no data is transmitted. An information byte is recognized by the receiver when the line goes low. This is the function of the start bit. Information bits then follow. Dependent on the application, these bits may be followed by a parity bit, which is used for error checking. At last one or two stop bits take the line back to logic high level.

An important aspect concerning the simulation, is the bit rate used to transmit data. The MCU may operate at many different bit rates. The rate the MCU works with is actually dependent on the clock

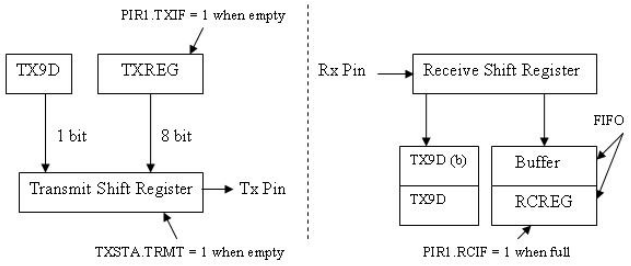

Figure 4: Transmit and receive circuits.

frequency it uses. It is therefore possible to tune this rate to one of the rates commonly used by the PC if the MCU is clocked by an external source of oscillation. However, the two bit rates are not exactly equal. Errors occur if the bit rate of the MCU is $\pm$ a certain percentage of the computer's bit rate. The actual percentage depends on settings in the MCU. Individual bits are sampled a number of times, and a majority circuit is used to determine the actual logic level of the bit. Generally the bit transition of the MCU cannot drift more than a little less than a half of one bit period of the computer. This is approximately equal to $\pm$ 5 % of the computers bit rate. Since the transmission and reception are again synchronized when a new byte is transferred, no errors will occur if the percentage error is between these limits [10].

4.4 Microchip PIC18F452 microcontroller

4.4.1 Serial buffers

Both the transmit and receive buffer of the MCU have a capacity of two bytes. As seen in Figure 4, the received data is clocked serially into the receive shift register. Once a complete byte has been obtained, the byte is shifted (in parallel) into the buffer. The byte is simultaneously shifted into the RCREG register if a flag indicates that it is "empty", i.e. data has been collected.

For transmission between the MCU and the computer, the highest possible throughput is desired. For the real time simulations 19200 bps is used, which implies 2,4 bytes per ms for continuous transmission. To obtain this rate, bytes in the RCREG register must be read once the receive flag is set. This is done efficiently by letting this flag trigger a high-priority interrupt service routine on the MCU. The routine reads all bytes into a data buffer. Since a byte is read in only a few microseconds, and the byte transmit time is almost half of a millisecond, useful work may be done

in between to get the most time efficient simulator. The transmit algorithm works in the same fashion. In the MCU data is transmitted using a loop which writes the bytes representing the actuator vector.

4.4.2 Floating-point storage

The floating-point format used to store values in the MCU is the IEEE-754 single precision format. The storage format of the bits representing a value differs slightly from the format used on a computer. Conversion between these formats is therefore required to exchange data properly. Special functions exist in the MCU, which do the conversions; but the choice has been made to implement the conversion in the computer instead. This saves time, since the computer is much faster than the MCU.

4.5 Real time simulation

Real time systems requires that the calculations must be performed within a specified time interval - in the case of the satellite system this will be the system sampling period. If the system fails to meet this timing requirement then the system will not work, and the control system will fail. Most operations always use a fixed time interval, so it may easily be determined if the system may operate in real time. Typically the system should be designed with a time margin. If the work may be done within 90 % of the available time, then a time safety margin exists.

Synchronization which essentially means that the MCU and the computer share the same clock, is important in the closed loop simulation. For this paper the MCU is used to generate a real time clock. Possible timing accuracy for the unit is dependent on the clock frequency. For the MCU used in this work, a timing accuracy of $\sim 10^{-6}s$ is possible. This is considered accurate enough for our purpose.

Timing information is obtained from a timer. A timer is a register which, when started, stores a counter variable that is incremented each clock cycle. When the counter reaches its maximum value $((2^{16}-1)$ for 16 bit counters), it overflows and triggers an interrupt. The interrupt may be used to perform one computation loop. Bytes are sent back and forth, and control and actuator values are calculated. Then the system returns to a waiting mode, until the next timing overflow occurs. The computer will be synchronized with the MCU since the computer must wait until the MCU transmits the needed data.

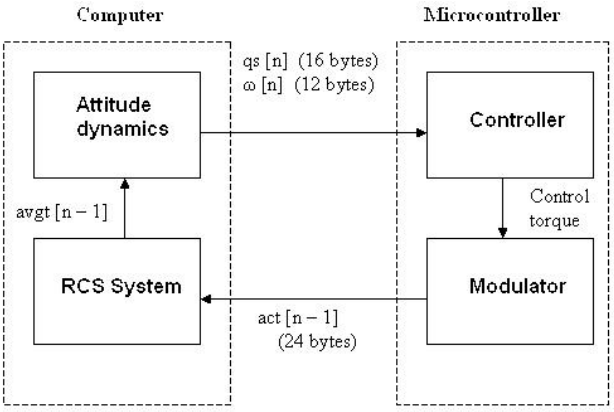

Figure 5: closed loop system.

Some sort of error detection algorithm should exist in the system. The data should be checked for their validity and acknowledgements have to be sent between the units, indicating that all data have been received properly. An error flag may be sent if an error has occurred. Further, if the error persists for n=10 consecutive cycles, some action like restart of the system has to be taken. This aspect is not studied in this paper.

4.6 Closed loop simulation - Microcontroller and Computer

Computations with respect to control and modulation of thrusters are performed in the MCU, while the computer only computes the attitude response and presents a visualization of this. Every sampling period the computer and MCU have to communicate, in order to exchange necessary data. Setup of the closed loop system is depicted in Figure 5. The computer transmits bytes representing the current Euler parameter vector and angular velocities of the satellite with respect to the orbit frame. Since each floating-point value is represented by four bytes, a total of 28 bytes is required to be sent to computer. If sent continuously, this will require about 12 ms. The actuator vector representing the duty cycles of the thrusters are sent back to the computer. Each duty cycle is given by four bytes, demanding a total transmission of 24 bytes for all six thrusters (10 ms). Thus ideally 22 ms are spent on transmission and reception. In the practical system actually some more time may be used because the computer sends the data using the operating system. There may also exist some drift in the amount of time used to send the bytes.

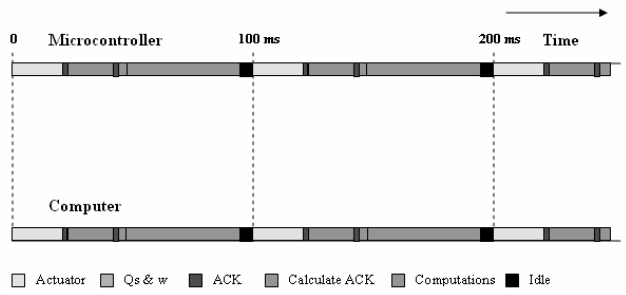

Figure 6: Timing diagram.

4.7 Timing diagram

As mentioned earlier, in practical systems there will always be delays/drifts, and some parts of the systems may depend on other parts. Therefore some devices may lag behind one or two computation cycles with respect to others. This aspect should be included in the simulation. A timing diagram as shown in Figure 6, will give good knowledge of the dependencies in the system. Time in the MCU and computer has to be synchronized, meaning that initial time is the same in the computer and the MCU. Upon receiving the data representing the system state vector, the MCU uses this information to perform one calculation loop. The Euler parameter error vector and the control torque equation are both calculated. Next the thruster allocation algorithm calculates the actuator vector. This vector will be sent at the start of the next sampling period, thus a delay of one sample period exists in the system. Simulations of the autonomous system also use this delay. In this way, they are comparable with the real time simulations.

4.8 Real time communication

A total of about 50 bytes must be sent if floating-point values are used. Four-byte single-precision floating-point values give the best approximation to the autonomous simulation. MatLab uses eight-byte double-precision values in calculations. When values are to be transmitted to the MCU, there will be some round-off due to the conversion between double and single precision values. An error of about $\sim 10^{-6}$ is the result. When using Bang-bang control, the actuators are either on or off during a sample period. Thus the actuator vector may be represented using six bits within a single byte. Each bit represents one actuator. Logic high means that the thruster should be on. Logic low means that the thruster should be off. Possibly the remaining bits could be used for some acknowledgement purpose. If a modulation technique is used, the actuator vector will have values which vary between zero and one. Only one byte may be used to represent each value of the actuator vector, if Pulse Amplitude Modulation (PAM) is used to code each actuator value into a byte pattern representing a value ranging from 0 to 255. Division of the value by 255 yields the true value of the actuator. The same techniques may be applied to the current and desired Euler parameter vectors, since they vary in a fixed interval $(-1 < \varepsilon, \eta < 1)$.

5 Simulation results and discussion

5.1 Numerical Values

In the following simulations, numerical values for ESMO have been used. The moments of inertia for the ESMO satellite are expressed as $\mathbf{J} = \text{diag}\{\ 4.0 \quad 4.5 \quad 3.5\ \}\ \text{kgm}^2$. The satellite is placed in a circular lunar orbit at an altitude of 500 km. The thrusters are capable of providing 0.13 N. Accuracy demands are set to 0.1° for the Euler angles and $0.01^\circ/s$ for the angular velocities. The attitude dynamics equations may be approximated as three second order systems, one for each axis. Each system has two poles associated with it. The poles may be placed independently for each axis. In this paper, the poles are chosen to be equal for all axes. The values are $p_1 = -0.5$ and $p_2 = -1$. The PD controller uses the Matlab command $place(A, B, p)$, which is a pole placement method, to derive the state feedback matrices $\mathbf{K}_p$ and $\mathbf{K}_d$. The Bang-bang controller with dead-zone uses $K_P = 2$, and $K_D = 3$, and a dead-zone of 0.02 is set for the error function. Simulations in this paper are carried out using a fixed step Runge-Kutta algorithm of fourth order. The fixed step algorithm is used because the simulations use a fixed sample time of 100 ms.

5.2 Autonomous

This section presents simulations for the autonomous system, which means that the computer represents the complete system. Simulations are performed using Euler parameter representation. Plots for the Euler angles are obtained by conversion.

5.2.1 Bang-bang with dead-zone

Figure 7 shows the response when a dead-zone of 0.02 degree is set for the error function for an initial errors

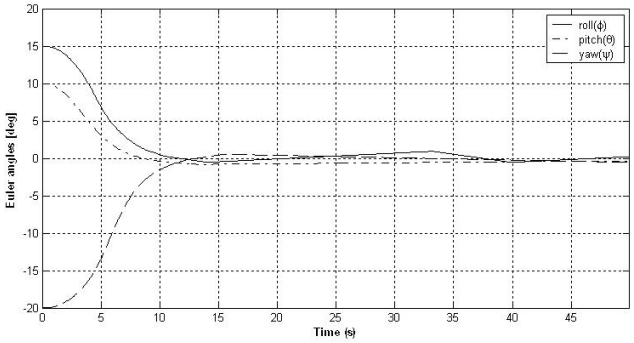

Figure 7: Attitude response using Bang-bang with dead-zone.

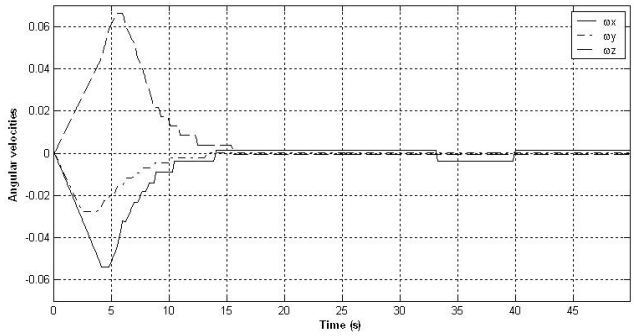

Figure 8: Angular velocities using Bang-bang with dead-zone.

of +15, +10 and -20 degree in roll, pitch, and yaw respectively. As seen in Figure 8, when the error angle becomes small, the angular velocities are constant for most of the time. Changes in the angular velocities are associated with thruster pulses. The thruster activations are shown in Figure 9. Angular velocity changes occur when pulses are applied.

5.2.2 Pulse-Width Modulation (PWM)

Modulation techniques allow the duty cycle to be adjusted in accordance with needed torque levels. Average torque, as seen over the sample period, will be approximately equal to the given control torque. The practical limitation that thrusters have a minimum on-

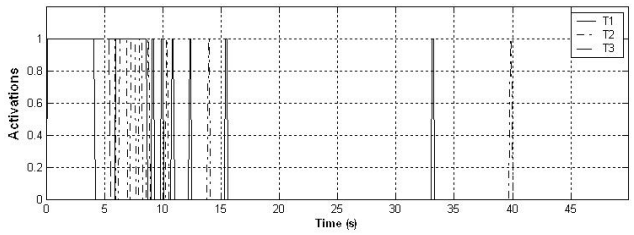

Figure 9: Thruster activations using Bang-bang with dead-zone.

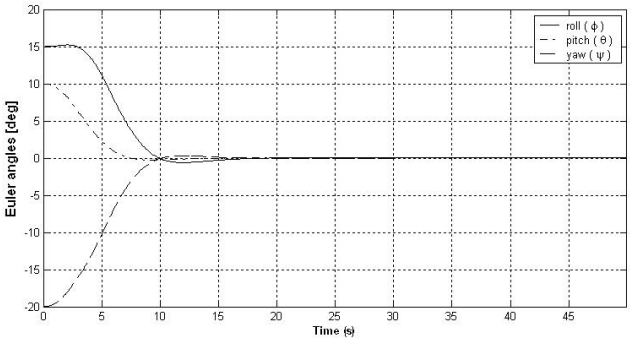

Figure 10: Attitude response using PWM.

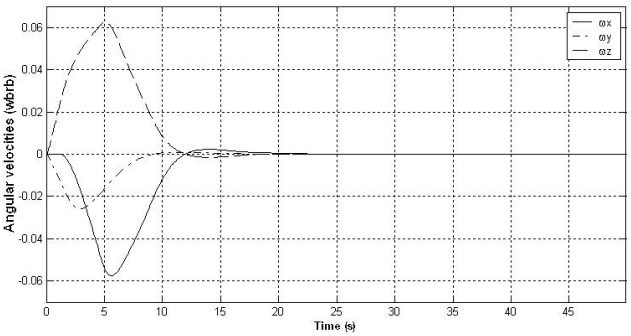

Figure 11: Angular velocities using PWM.

time, is included in the simulations. In this work, this duration is 2 ms. Figure 10 shows the result of simulation using the same parameters as in the previous case. Associated angular velocities are shown in Figure 11. Angular rates are no longer linearly varying, indicating that that thrusters are pulse-width modulated. After 20 s, both angular rates and Euler angles are within specified accuracy limits. The Euler angles are seen to overshoot no more than one degree. Critical damping may be achieved by slight change of controller gains.

5.3 Hardware-in-loop simulation

This section presents results of real time hardware-in-loop simulations carried out, where the MCU performs the control and modulation functions of the system in real time. All parameters are the same as in the autonomous simulations. Small numerical inaccuracies exist because of the rounding introduced when converting between IEEE-754 double and single precision formats.

5.3.1 Bang-bang with dead-zone

Figure 12 shows the error between the autonomous and real time closed loop simulation. The errors for the angular rates are shown in Figure 13.

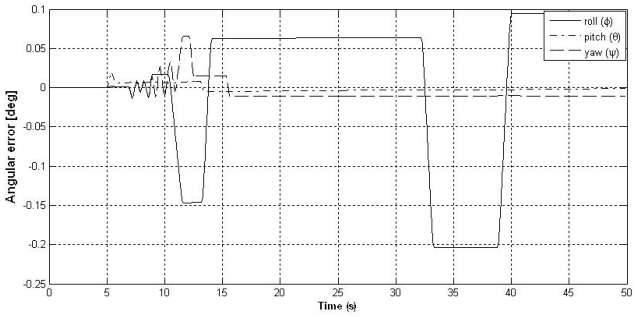

Figure 12: Angular error using Bang-bang with dead-zone.

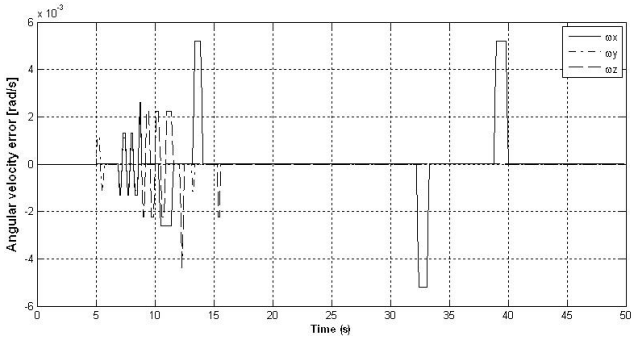

Figure 13: Angular velocities error using Bang-bang with dead-zone.

5.3.2 Pulse-Width Modulation (PWM)

Figure 14 shows the attitude response obtained from real time simulation of the PW modulator. Actuator values are transmitted as floating-point values back to MatLab. Only the smaller number of decimals should be a source of error. The angular error obtained is quite small.

Comparing the results with those given for autonomous simulations, we see that there is a close match, thus proving satisfactory performance of the ACS in real time environment for both schemes.

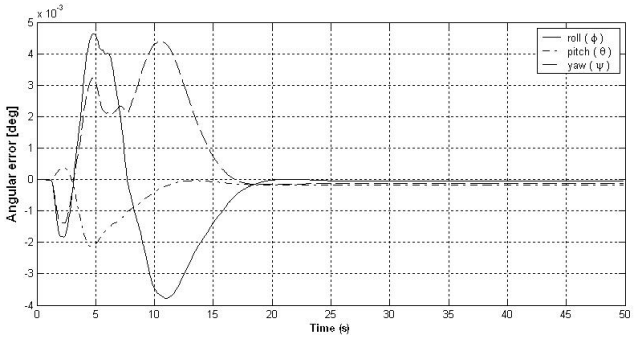

Figure 14: Angular error using PWM.

6 Conclusion

In this paper a mathematical model of ESMO has been developed, where six reaction control thrusters are used for attitude control. Bang-bang with dead-zone and PW modulation of the reaction thrusters have been studied. Simulations show that these techniques produce good responses. This paper has also shown that it is feasible to implement the attitude control system in a MCU for real time closed loop operation. Various issues connected with real time simulations of the ACS have been presented. The controller performance confirmed to be satisfactory, both in autonomous and hardware-in-loop simulations. A close match between the results are seen.

A computation cycle of 100 ms is used in the simulations. Though simulations showed satisfactory performance of the ACS in the real time environment compared to the autonomous, it is suggested to use a shorter computation cycle to improve performance. It is also suggested that improved modulation techniques like Pulse-Width Pulse Frequency (PWPFM) are adopted. Another issue which may be a subject for future study, is the need for a error handling logic to deal with transmission errors. Further studies with nonlinear controllers based on the Lyapunov method may be carried out using this test bed in the future.

References

[1] Finnset, R., *Real time hardware-in-loop simulation of ESMO ACS*, Master's thesis, Narvik University College, Narvik, Norway, 2005.

[2] Lund, H., *Reaction wheels in a tetrahedron*, Master's thesis, Narvik University College, Narvik, Norway, 2005.

[3] Paulsen, C., *Satellite Attitude Control, Disturbances and momentum management*, Master's thesis, Narvik University College, Narvik, Norway, 2005.

[4] Egeland, O. and Gravdahl, J. T., *Modeling and Simulation for Automatic Control*, Marine Cybernetics, Trondheim, Norway, 2002.

[5] Sidi, M. J., *Spacecraft Dynamics and Control*, Cambridge University Press, New York, 1997.

[6] Strang, G., *Linear Algebra and its Applications*, Harcourt Brace Jovanovich College Publishers, Orlando, Florida, USA, 1988.

[7] Wertz, J. R., *Spacecraft Attitude Determination and Control*, Kluwer Academic Publishers, London, 1978.

[8] Bordany, RE., S. W. and Wu, S., "In-orbit Estimation of the Inertia Matrix and Thruster Parameters of UoSAT-12," *14th Annual AIAA/USU Conf on Small Sats, Utah State University*, Logan, Utah, USA), 2000.

[9] Hou, L. and Michel, A. N., "Stability analysis of pulse-width-modulated feedback systems," *Acta Astronautica*, Vol. 37, No. 9, 2001, pp. 1335–1349.

[10] Peatman, J. B., *Embedded Design with the PIC18F452 Microcontroller*, Pearson Education, 2003.