THE MODELING AND SYSTEM IDENTIFICATION OF THE
DYNAMICS IDENTIFICATION AND CONTROL EXPERIMENT

by

Eric M. Choi

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science

University of Toronto
Institute for Aerospace Studies

(c) Copyright by Eric M. Choi 1997
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author’s permission.

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L’auteur conserve la propriété du droit d’auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.
The Modeling and System Identification of the Dynamics Identification and Control Experiment

by

Eric M. Choi

A thesis submitted in conformity with the requirements for the degree of Master of Applied Science

University of Toronto
Institute for Aerospace Studies

(c) Copyright by Eric M. Choi 1997

The Dynamics Identification and Control Experiment (DICE) is designed to conduct control and system identification (SI) experiments inside the middeck of the Space Shuttle. The goal of this thesis was to find out whether the current design is sufficient for producing good SI results. First, the sensitivity of the SI algorithm to generic variations in parameters was quantified. Then a prototype DICE simulator, called ProtoSim, was developed. This simulator consisted of models of the actuators and sensors connected to a model of the structural dynamics. It was validated through a set of deterministic tests. A series of SI experiments were then performed using ProtoSim. The simulations showed that the current DICE design produces relatively poor SI results. Possible reasons for this are given, and recommendations are made as to how the experiment might be modified for the real DICE mission.
"If I have seen further, it is by standing upon the shoulders of Giants."

The above quote, attributed to Newton, has been cited so often as to become almost cliché. Yet, the author cannot think of a more appropriate statement to describe his feelings towards all those that have contributed to his work. This thesis would simply not be in existence were it not for the invaluable assistance of innumerable members of the Spacecraft Dynamics Group and employees of Dynacon Enterprises Ltd. It is with the deepest regret that space only permits him to express his public gratitude to individuals whose contributions were most critical. The author's heartfelt thanks to:

- Prof. Peter Hughes, principal investigator for the Dynamics Identification and Control Experiment, for supervising this thesis and allowing the author the privilege of being a (small) part of the DICE team.
- Drs. Kieran Carroll and Phung Nguyen, of Dynacon and System Simulations respectively, for the initial definition of this thesis and providing start-up supervision.
- Robert Bauer and Robert Zee, Ph.D Candidates in the Spacecraft Dynamics Group. To Rob Bauer, for providing both the system identification software and invaluable guidance during the experiments conducted by the author, and to Rob Zee for his work in simulator development and controllers. To both gentlemen, the author's thanks also for all their day-to-day help and encouragement.
- Stephen Piggott of Dynacon, for providing the Gyrosim dynamics block and showing how to implement it, and for his tutorials on motor modeling.
- James Crawford, Ph.D Candidate in the Spacecraft Dynamics Group, for providing crucial specifications on CMGs (Crawford's Mechanical Gadgets) and for allowing the author the use of his implementation of the "reset-integrator" friction model.
- Don McTavish of Dynacon, for showing the author how to set up the "reset-integrator" friction model and his advice on motor modeling.
Andrew Allen of Neptee Corporation, for providing essential details on the Space Vision System.

- Johanne Heald, fellow M.A.Sc Candidate in the Spacecraft Dynamics Group, for the mutual *Xmath* help sessions (the blind leading the blind), profound philosophical thoughts, and good humor.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>i</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>ii</td>
</tr>
<tr>
<td>LIST OF ACRONYMS</td>
<td>vii</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>viii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>ix</td>
</tr>
<tr>
<td>LIST OF SYMBOLS</td>
<td>x</td>
</tr>
</tbody>
</table>

## 1.0 INTRODUCTION

1.1 Background                                                        1
1.2 The Dynamics Identification and Control Experiment                2
1.3 Thesis Overview and Scope                                          4

## 2.0 SYSTEM IDENTIFICATION THEORY

2.1 Overview of System Identification                                 5
2.2 Observer/Kalman Filter Identification                              6
2.3 The Eigensystem Realization Algorithm                              10

## 3.0 PROTOSIM: A PROTOTYPE DICE SIMULATOR

3.1 Overview of ProtoSim                                               13
3.2 The Gyrosim Dynamics Block                                         14
3.3 Transformation from Absolute to Relative Coordinates              20
3.4 Actuator Models                                                   25
   3.4.1 Reaction Wheels                                               25
   3.4.2 Control Moment Gyros                                          31
3.5 Sensor Models                                                     38
   3.5.1 Encoders                                                     38
   3.5.2 Inertial Rate Sensor                                          39
   3.5.3 Rib Tip Sensors                                               41
   3.5.4 Potentiometers                                               43
   3.5.5 Space Vision System                                          45
3.6 The Integrated ProtoSim                                           46
3.7 Simulator Verification                                             48
   3.7.1 Actuator Tests                                               48
   3.7.2 Energy and Momentum Balance                                  51
   3.7.3 Deterministic Simulations                                    53
3.8 ProtoSim User's Guide                                             55
4.1 The Sensitivity of OKID/ERA to Parameter Variations 57
4.2 System Identification of ProtoSim 60
4.2.1 Identification of the Dynamics Block 61
4.2.2 Identification of the Dynamics Block in Relative Coordinates 63
4.2.3 Sensor Sizing 66
4.2.4 Identification of ProtoSim 68

5.0 CONCLUSION 71

6.0 REFERENCES 73

A. APPENDIX A: PROTOSIM SCRIPTS AND FUNCTIONS 75
A.1 ProtoSim Execution Script: protorun.ms 75
A.2 Parameter Setup Function: set_parameters.msf 82
A.3 Random Input Generation Function: random_input.msf 88
A.4 NASTRAN Matrices Setup Function: simmattipcon.msf 89

B. APPENDIX B: PROTOSIM VERIFICATION RUNS 97
B.1 Energy Balance Test Results 98
B.1.1 Force Input 98
B.1.2 Torque Input 100
B.2 Momentum Balance Test Results 102
B.2.1 Force Input 102
B.2.2 Torque Input 103
B.3 Deterministic Test Results 104
B.3.1 Predicted Outputs for X- and Y-Axis Reaction Wheel Pulses 104
B.3.2 Simulation Results for X-Axis Reaction Wheel Pulse 105
B.3.3 Simulation Results for Y-Axis Reaction Wheel Pulse 109
B.3.4 Predicted Outputs for Z-Axis Reaction Wheel Pulse 113
B.3.5 Simulation Results for Z-Axis Reaction Wheel Pulse 114
B.3.6 Simulation Results for CMG #1 Pulse 117
B.3.7 Simulation Results for CMG #2 Pulse 118

C. APPENDIX C: SYSTEM IDENTIFICATION SENSITIVITY RUNS 119
C.1 OKID/ERA Parameter Sensitivity Runs 120
   Sampling Frequency and SI Quality 121
   Skew Time and SI Quality 122
   Sensor Resolution and SI Quality 123
   Actuator Resolution and SI Quality 124
   Actuator Noise and SI Quality 125
   Sensor Noise and SI Quality 126
   Input/Output Noise and SI Quality 127
C.2 ProtoSim System Identification Runs
   C.2.1 Dynamics Block
   C.2.2 Dynamics Block in Relative Coordinates
       One Minute Experiment
       Two Minute Experiment
       Three Minute Experiment
   C.2.3 Sensor Sizing
       Hub Rotation from SVS
       Hub Rotation from Integrating Inertial Rate Sensor Signal
   C.2.4 Full ProtoSim Simulator
**LIST OF ACRONYMS**

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMG</td>
<td>Control Moment Gyro</td>
</tr>
<tr>
<td>DICE</td>
<td>Dynamics Identification and Control Experiment</td>
</tr>
<tr>
<td>ERA</td>
<td>Eigensystem Realization Algorithm</td>
</tr>
<tr>
<td>EUTMA</td>
<td>Engineers Use Too Many Acronyms</td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier Transform</td>
</tr>
<tr>
<td>FSS</td>
<td>Flexible Space Structures</td>
</tr>
<tr>
<td>IDFT</td>
<td>Inverse Discrete Fourier Transform</td>
</tr>
<tr>
<td>LED</td>
<td>Light Emitting Diode</td>
</tr>
<tr>
<td>LEPD</td>
<td>Lateral Effect Photo-Diode</td>
</tr>
<tr>
<td>MACE</td>
<td>Middeck Active Control Experiment</td>
</tr>
<tr>
<td>MODE</td>
<td>Middeck 0-Gravity Dynamics Experiment</td>
</tr>
<tr>
<td>OKID</td>
<td>Observer/Kalman Filter Identification</td>
</tr>
<tr>
<td>PD</td>
<td>Proportional-Derivative</td>
</tr>
<tr>
<td>RMS</td>
<td>Root-Mean-Squared</td>
</tr>
<tr>
<td>SI</td>
<td>System Identification</td>
</tr>
<tr>
<td>STS</td>
<td>Space Shuttle Transportation System</td>
</tr>
<tr>
<td>SVS</td>
<td>Space Vision System</td>
</tr>
<tr>
<td>UTIAS</td>
<td>University of Toronto Institute for Aerospace Studies</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>Table 3.1</td>
<td>The inputs and outputs of the REZGyroSim DICE dynamics block.</td>
</tr>
<tr>
<td>Table 3.2</td>
<td>The inputs and outputs of the Transformation block.</td>
</tr>
<tr>
<td>Table 3.3</td>
<td>The inputs and outputs of the Rotation Transformation block.</td>
</tr>
<tr>
<td>Table 3.4</td>
<td>The inputs and outputs of the reaction_wheel block.</td>
</tr>
<tr>
<td>Table 3.5</td>
<td>The inputs and outputs of the rw_dynamics block.</td>
</tr>
<tr>
<td>Table 3.6</td>
<td>The inputs and outputs of the CMG gimbal_motor block.</td>
</tr>
<tr>
<td>Table 3.7</td>
<td>The inputs and outputs of the CMG spin_motor block.</td>
</tr>
<tr>
<td>Table 3.8</td>
<td>The inputs and outputs of the cmg_dynamics block.</td>
</tr>
<tr>
<td>Table 3.9</td>
<td>The inputs and outputs of the cmg_unit block.</td>
</tr>
<tr>
<td>Table 3.10</td>
<td>The inputs and outputs of the encoder block.</td>
</tr>
<tr>
<td>Table 3.11</td>
<td>The inputs and outputs of the rate_sensor block.</td>
</tr>
<tr>
<td>Table 3.12</td>
<td>The inputs and outputs of the lepdribtip sensor block.</td>
</tr>
<tr>
<td>Table 3.13</td>
<td>The inputs and outputs of the potentiometer block.</td>
</tr>
<tr>
<td>Table 3.14</td>
<td>The inputs and outputs of the svs block.</td>
</tr>
<tr>
<td>Table 3.15</td>
<td>ProtoSim's inputs and outputs.</td>
</tr>
<tr>
<td>Table 3.16</td>
<td>Actuator model test results.</td>
</tr>
<tr>
<td>Table 3.17</td>
<td>Energy balance test results.</td>
</tr>
<tr>
<td>Table 3.18</td>
<td>Linear and angular momentum balance test results.</td>
</tr>
<tr>
<td>Table 3.19</td>
<td>CMG gimbal current step test results.</td>
</tr>
<tr>
<td>Table 3.20</td>
<td>ProtoSim files required for execution.</td>
</tr>
<tr>
<td>Table 4.1</td>
<td>The effect of experiment time on SI model quality.</td>
</tr>
<tr>
<td>Figure 1.1</td>
<td>The Daisy facility at UTIAS.</td>
</tr>
<tr>
<td>Figure 1.2</td>
<td>DICE configuration in the Space Shuttle middeck.</td>
</tr>
<tr>
<td>Figure 2.1</td>
<td>The basic SI process.</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>The REZGyroSim block of the DICE dynamics.</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>Illustration of absolute reference frames employed by Gyrosim</td>
</tr>
<tr>
<td>Figure 3.3</td>
<td>Vectors expressed in the absolute and relative rib-tip frames.</td>
</tr>
<tr>
<td>Figure 3.4</td>
<td>The absolute-to-relative Transformation block for rib deflections.</td>
</tr>
<tr>
<td>Figure 3.5</td>
<td>The Rotation Transformation block for rib #1.</td>
</tr>
<tr>
<td>Figure 3.6</td>
<td>The reaction_wheel block.</td>
</tr>
<tr>
<td>Figure 3.7</td>
<td>The reaction wheel dynamics block rw_dynamics.</td>
</tr>
<tr>
<td>Figure 3.8</td>
<td>The CMG gimbal_motor block.</td>
</tr>
<tr>
<td>Figure 3.9</td>
<td>The CMG spin_motor block.</td>
</tr>
<tr>
<td>Figure 3.10</td>
<td>The cmg_dynamics blocks for ribs 1, 3, and 5.</td>
</tr>
<tr>
<td>Figure 3.11</td>
<td>The integrated cmg_unit block for ribs 1, 3, and 5.</td>
</tr>
<tr>
<td>Figure 3.12</td>
<td>ProtoSim’s cmg_encoder block.</td>
</tr>
<tr>
<td>Figure 3.13</td>
<td>The rate_sensor block.</td>
</tr>
<tr>
<td>Figure 3.14</td>
<td>The lepd_rib tip sensor block.</td>
</tr>
<tr>
<td>Figure 3.15</td>
<td>The inputs and outputs of the potentiometer block.</td>
</tr>
<tr>
<td>Figure 3.16</td>
<td>The svs block that emulates the Space Vision System.</td>
</tr>
<tr>
<td>Figure 3.17</td>
<td>The top-level ProtoSim superblock.</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>A two mass-spring-dashpot mechanical system.</td>
</tr>
<tr>
<td>Figure 4.2</td>
<td>SI experiment results for rib #1 (dynamics block only).</td>
</tr>
<tr>
<td>Figure 4.3</td>
<td>SI experiment results for rib #1 (dynamics block, relative coordinates)</td>
</tr>
<tr>
<td>Figure 4.4</td>
<td>SI experiment results for rib #1 (using sensor data).</td>
</tr>
<tr>
<td>Figure 4.5</td>
<td>SI experiment results for rib #1 (with complete ProtoSim).</td>
</tr>
</tbody>
</table>
Lowercase Symbols

\( a \)  unit vector defining spin axis
\( b \)  unit vector defining gimbal axis
\( c \)  dashpot constant
\( e \)  eigenvector
\( f \)  frequency
\( f \)  force
\( f \)  vector of generalized forces
\( g \)  torque
\( h \)  angular momentum
\( i \)  current
\( k \)  stiffness parameter
\( m \)  mass
\( n \)  gear ratio
\( p \)  number of non-zero Markov parameters
\( p \)  linear momentum
\( q \)  vector of generalized coordinates
\( q \)  image distance
\( r \)  displacement
\( t \)  time
\( x \)  state vector
\( u \)  input vector
\( v \)  component of velocity
\( v \)  velocity
\( w \)  work
\( y \)  output vector

Uppercase Symbols

\( A \)  state matrix
\( A^T \)  observer state matrix
\( B \)  influence matrix
\( B^T \)  observer influence matrix
\( B_o \)  input distribution matrix
\( C \)  output influence matrix
\( C^T \)  rotation matrix
\( C_o \)  output distribution matrix
\( D \)  transmission matrix
\( E \)  elastic modulus
IIlULl
Id\VL
bL
frame
of reference
observer gain
matrix
generalized Han
ekel matrix
identity matrix
second moment-of-inertia taken about mass center
motor inertia
second moment-of-inertia taken about body-fixed point other than mass center
constant
"stiffness" matrix
inductance
system inertia matrix
null-matrix
observability matrix
controllability matrix
resistance
kinetic energy
voltage in frequency domain
observer Markov parameter

Lowercase Greek Symbols

ς  rotational displacement
δ  beam deflection
c  small perturbation about nominal conditions
σ  non-zero singular value
λ  eigenvalue
ν  spin rate
η  vector of modal coordinates
τ  torque
ω  angular velocity

Uppercase Greek Symbols

Ω  matrix of modal frequencies
Σ  diagonal matrix of non-zero singular values
θ  angle
ζ  damping ratio

Other Notation

1  unit matrix
1.0 INTRODUCTION

1.1 Background

Since the launch of Alouette 1 in 1962, Canada has become a world leader in the development of satellite control systems. This expertise was largely gained from the fact that Alouette, Hermes, and many subsequent Canadian satellites can be classified as flexible space structures (FSS). In other words, these satellites were sufficiently flexible such that this flexibility was an important factor in their attitude dynamics, and thus to the design of their controllers [1].

The unique problems posed by FSS have vexed engineers for decades. Knowing the dynamical behavior of a spacecraft is crucial to designing its control system. On Earth, gravity makes it difficult to accurately test a spacecraft's flexible modes. To save weight, appendages such as solar panels and antennae are designed to hold themselves up only the microgravity of space. On the ground, these components must be supported by external mechanisms. Since this changes the structural characteristics of the spacecraft, ground-based tests seldom provide an accurate measure of how they will actually behave in orbit.

Computer and mathematical models are not much help for precisely the same reason -- there are no ground tests available that can verify with a high degree of confidence whether these models are correct. The database of in-flight experience from actual missions is still fairly limited because many satellites were artificially stiffened to make them more "predictable", and some computer codes still simulate spacecraft as rigid bodies. This is obviously unrealistic, for a modern satellite with lots of booms and solar panels is anything but perfectly stiff.

In a spacecraft, flexibility effects are highly sensitive to both pointing accuracy and size. Therefore, structural elasticity cannot be neglected in spacecraft with high accuracy requirements like Milstar or the Hubble Space Telescope, or with physically massive structures like the International Space Station. In this regard, existing computer simulations can be far off the mark.
Current spacecraft controller designs can therefore involve an unsettling amount of guesswork. This can manifest itself in unexpected vibrational problems like those experienced by Hubble with its original set of solar panels.

Since the early 1980s, the premier ground-based FSS experimental facility has been Daisy, illustrated in Figure 1.1. Daisy is a flexible spacecraft emulator at UTIAS. It simulates to a high degree of fidelity many important FSS characteristics. Daisy has three rigid modes, 20 elastic modes, damping, and satellite-related sensors and actuators such as reaction wheels, thrusters, accelerometers, and encoders [2]. Work on Daisy has been highly successful, with the facility being instrumental to innumerable projects and papers.

However, Daisy has its limits. Since it is on Earth, gravity and even aerodynamic drag induce torques that are simply not present in orbit (friction in the gravity-loaded bearings is one example). This is a significant impediment to its utility as a test-bed for spacecraft controller techniques. To improve results, the gravity factor should be eliminated. A spaceborne test article is therefore the next logical step.

1.2 The Dynamics Identification and Control Experiment

Thus far, there have been only two FSS test articles launched aboard the Space Shuttle. The Middeck 0-Gravity Dynamics Experiment (MODE) has flown twice, most recently on STS-62 in March 1994. The most ambitious on-orbit FSS experiment to date has been the Middeck Active Control Experiment (MACE), which was executed on STS-67 in March 1995. While these missions were noteworthy successes, the results of only three experiments is far too meagre a data set for a
field as expansive and important as FSS. Further insight into this complex problem is required.

Under development by UTIAS and Dynacon Enterprises Ltd., the Dynamics Identification and Control Experiment (with the agreeable acronym DICE) is an advanced spaceborne FSS test article scheduled for launch aboard a Space Shuttle mission in 1999. As illustrated in Figure 1.2, DICE is essentially a miniature satellite about a meter in diameter. It includes many components analogous to its full-sized cousins: thrusters (which use recycled cabin air) for station-keeping and momentum management, reaction wheels, control moment gyros (CMGs), a complete command and telemetry system, and five flexible ribs. These ribs are intended to emulate the non-rigid parts of a real satellite such as booms or antennae.

DICE will differ significantly from previous FSS experiments in three important ways. First, the structure of the test article is different. Second, DICE will have a more advanced suite of actuators and sensors, including the Space Vision System (SVS). SVS was the sensing system used in November 1995 by Canadian astronaut Chris Hadfield to attach a docking module to the Mir space station. Finally, while MODE and MACE were tethered during their operation, DICE will be a free-flyer.

The objectives of the DICE mission are twofold:

- To conduct on-orbit system identification (SI).
- To test high-accuracy, modern control system designs.

During its flight, the reaction wheels and the CMGs will be used to excite the structure of the test article. The dynamical response of DICE to these excitations will be sensed. Using SI, a mathematical model of DICE will then be developed based on the known inputs and the measured response of the test article. This generated SI model will then be used to refine and update the controller design.

This unique experiment will provide engineers with hard data on how flexible spacecraft behave. The knowledge gained may one day make feasible very elaborate structures like the
glamorous roulette-wheel space station in *2001: A Space Odyssey*. It will also take much of the guesswork out of FSS-related design issues. By rolling the DICE in space, UTIAS and Dynacon will help ensure that future engineers won't have to.

### 1.3 Thesis Overview and Scope

The purpose of this thesis was to contribute to the top-level system requirement definition for DICE. This was accomplished by developing a DICE simulator that was then used to find out whether SI could be successfully performed with the current design. Section 2.0 offers a brief summary of SI theory and algorithms. The development of the DICE simulator is described in Section 3.0, and the results of the SI runs are shown in Section 4.0. Finally, the thesis is concluded in Section 5.0, in which recommendations and conclusions are drawn.
2.1 Overview of System Identification

System identification (SI) is the process of developing or improving a mathematical representation of a physical system using experimental data. In the field of aerospace, there are essentially three types of SI work: modal parameter identification, structural-model parameter identification, and control-model identification. In the third type, the identification task is to produce a model that will adequately describe the input/output mapping. This model is then used as the "plant" in the design of control strategies for structural systems, particularly FSS.

The SI process is depicted schematically in Figure 2.1. Five key steps are involved. First, the levels of structural dynamic response that are likely to occur should be estimated. This leads to the second step, which is to set the instrumentation requirements necessary to sense the behavior of the system with sufficient accuracy and scope. Of primary concern are what kinds of actuators and sensors should be used, how many are needed, and where they should be located. The third step is to do the actual SI experiment, which involves exciting the structure and recording the associated response. Then, an SI algorithm must be applied to these data to identify the characteristics of the structure. These characteristics are typically described in a model consisting of a set of system matrices. Finally, this model is applied to the design of the controller.

![Figure 2.1: The basic SI process.](image_url)
Many different SI algorithms have been developed for control system design. These include Observer/Kalman Filter Identification and the Eigensystem Realization Algorithm (OKID/ERA) [3], Q-Markov Covariance Equivalent Realization [4], and the Koopmans-Levin Method [5]. In a comparative survey, Bauer [6] endorsed OKID/ERA based on its performance and suitability for coding. This technique, implemented in Xmath software by Bauer [7], was thus chosen as the SI method used for this thesis. The software works in three major computational steps:

1. The observer Markov parameters are calculated with OKID.
2. From the observer Markov parameters, OKID recovers the system Markov parameters.
3. Given the system Markov parameters, ERA is used to realize the discrete-time state-space system matrices $A$, $B$, $C$, and $D$.

**2.2 Observer/Kalman Filter Identification**

Many common SI algorithms, including ERA, use a sequence of constant matrices known as the system Markov parameters as the basis for identifying mathematical models. These Markov parameters are the unit sample response or pulse response histories of the discrete-time system. Several techniques have been developed to extract Markov parameters from experimental input/output data. In one method, Fast Fourier Transforms (FFT) of the inputs and measured outputs are used to compute the frequency response functions. An Inverse Discrete Fourier Transform (IDFT) is then used to compute the Markov parameters.

In a comparative study, Bauer [6] concluded that Observer/Kalman Filter Identification (OKID) is superior to the FFT method. The FFT algorithm requires long input and output histories, is sensitive to aliasing and numerical ill-conditioning problems, and often yields inaccurate Markov parameters. OKID can find the Markov parameters more quickly and accurately than FFT/IDFT using significantly less data.

Rather than identifying the system Markov parameters directly, which may exhibit very slow decay in lightly damped systems, OKID first introduces an asymptotically stable observer. This observer compresses the data and reduces the influence of noise. Since any poles needed
can be assigned for the observer, the damping of the system can also be effectively increased. Observer Markov parameters are then solved from the input and output data, from which the system Markov parameters are finally computed.

Given a set of experimental input and output data, the OKID algorithm can be summarized as follows, based on the development by Juang [3]. For a rigorous derivation, the reader is urged to consult that reference. From control systems theory [8], the general state equation for a system with \( r \) inputs, \( m \) outputs, and of order \( n \) can be expressed as:

\[
x(k+1) = Ax(k) + Bu(k)
\]  
\[
(2.1)
\]

where:
\[ A = \text{state matrix} \ (n \times n) \]
\[ B = \text{influence matrix} \ (n \times r) \]
\[ x(k) = \text{state vector} \ (n \times 1) \]
\[ u(k) = \text{input vector} \ (r \times 1) \]

The corresponding output equation is:

\[
y(k) = Cx(k) + Du(k)
\]  
\[
(2.2)
\]

where:
\[ C = \text{output influence matrix} \ (m \times n) \]
\[ D = \text{transmission matrix} \ (m \times r) \]

An observer determines an estimate of the system state from the input and the measured output. Adding and subtracting the term \( Gy(k) \) to the right-hand side of (2.1) yields:

\[
x(k+1) = Ax(k) + Bu(k) + Gy(k) - Gy(k)
\]  
\[
(2.3)
\]

where:
\[ G = \text{observer gain matrix} \ (n \times m) \]
\[ y(k) = \text{output vector} \ (m \times 1) \]
Thus, the original system state equation (2.1) becomes the observer state equation:

$$\dot{x}(k+1) = \bar{A}x(k) + \bar{B}v(k)$$  \hspace{1cm} (2.4)

where: \[ \bar{A} = A + GC \]
\[ \bar{B} = [B+GD \quad -G] \]
\[ v(k) = [u(k) \quad y(k)]^T \]

$\bar{A}$ can be made as asymptotically stable as needed because the observer gain $G$ is arbitrarily chosen. If this is the case, then for large $k$ the estimated state converges to the true state [3]. Ideally, $G$ should be chosen such that the state estimate is robust to the effects of systemic noise. The famous Kalman filter gives an optimal estimate if the system is contaminated by noise [8] -- hence, its place as the "K" in OKID.

To find the pulse response history of the observed system, $u_i(0) = 1 \ (i=1,2,...,r)$ and $u_i(k) = 0 \ (k=1,2,...)$ are substituted into (2.4) and (2.2). The result can be assembled into a sequence of $m \times r$ pulse response matrices $Y_k$. These are the observer Markov parameters:

$$Y_0 = D, \quad Y_1 = CB, \quad \bar{Y}_2 = CA\bar{B}, \ldots, \quad \bar{Y}_k = CA^{k-1}\bar{B}$$  \hspace{1cm} (2.5)

Equations (2.4) and (2.2) can be represented in a matrix form:

$$y = \bar{Y}V$$  \hspace{1cm} (2.6)

where:

$$y = [y(0) \quad y(1) \quad y(2) \ldots y(p) \ldots y(n-1)]$$
$$\bar{Y} = [D \quad CB \quad CA\bar{B} \ldots \quad CA^{(p-1)}\bar{B}]$$
$$V = \begin{bmatrix} u(0) & u(1) & u(2) & \ldots & u(p) & \ldots & u(n-1) \\ v(0) & v(1) & v(p-1) & \ldots & v(n-2) \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ v(0) & \ldots & v(n-p-1) \end{bmatrix}$$
The parameter $p$ in (2.6) denotes the number of non-zero observer Markov parameters to be computed. It must be chosen such that $p \geq n/m$, where $n$ is the order of the system and $m$ is the number of outputs [3]. In theory, SI results should improve as $p$ increases [7]. However, it has been experimentally observed that this is only the case for systems corrupted with noise. For ideal noise-free simulated models, increasing $p$ may actually worsen the SI results due to numerical errors (see Section 4.2).

To compute the observer Markov parameters, solve (2.6) for $\overline{Y}$:

$$\overline{Y} = y V^t$$

(2.7)

where: $V^t = V^T [VV^T]^{-1}$, the pseudo-inverse of $V$

The system Markov parameters $Y$ are recovered from the observer Markov parameters by partitioning the left-hand side of (2.7) such that:

$$\overline{Y} = \begin{bmatrix} \overline{Y}_0 & \overline{Y}_1 & \overline{Y}_2 & \ldots & \overline{Y}_p \end{bmatrix}$$

(2.8)

where: $\overline{Y}_0 = D$

$$\overline{Y}_k = \begin{bmatrix} C(A+CG)^{(k-1)}(B+GD) & -C(A+GC)^k G \end{bmatrix}$$

$\triangleq \begin{bmatrix} \overline{Y}_k^{(1)} & \overline{Y}_k^{(2)} \end{bmatrix}$

The general relationship between the actual system Markov parameters and the observer Markov parameters is:

$$Y_0 = \overline{Y}_0 = D$$

$$Y_k = \overline{Y}_k^{(1)} - \sum_{i=1}^{k} \overline{Y}_i^{(2)} \overline{Y}_{k-i} \text{ for } k=1,\ldots,p$$

(2.9)

$$Y_k = -\sum_{i=1}^{p} \overline{Y}_i^{(2)} Y_{k-i} \text{ for } k=p+1,\ldots,\infty$$
Knowledge of the actual system Markov parameters allows one to obtain a discrete-time state-space realization of the system of interest using a method like ERA. The OKID method was an integral part of the SI testing of many real space systems, including the Canadarm and the Hubble Space Telescope [3].

2.3 The Eigensystem Realization Algorithm

Having found the system Markov parameters using a method like OKID, a state-space representation of the system can now be constructed. The idea of using Markov parameters as the basis for time-domain state-space SI was first developed by Ho and Kalman. Their technique was subsequently developed into the Eigensystem Realization Algorithm (ERA). The ERA method can be summarized as follows, based on the derivation by Juang [3]. Readers requiring a rigorous development should consult that reference.

ERA begins with the formation of an $\alpha m \times \beta r$ matrix called the generalized Hankel matrix, which is composed of the system Markov parameters from (2.9):

$$
H(k-1) = \begin{bmatrix}
Y_k & Y_{k-1} & \cdots & Y_{k-\beta-1} \\
Y_{k-1} & Y_{k-2} & \cdots & Y_{k-\beta} \\
\vdots & \vdots & \ddots & \vdots \\
Y_{k-\alpha-1} & Y_{k-\alpha} & \cdots & Y_{k-\alpha-\beta-1}
\end{bmatrix}
$$

(2.10)

In the Hankel matrix of (2.10), $\alpha = p$ and $\beta = (Npm)/r$, where $p$ was defined in Section 2.2 and $N$ is an integer satisfying:

$$
N \geq 1
$$

(2.11)

It can be shown [3] that the Hankel matrix can be decomposed into three matrices:

$$
H(k) = P_\alpha A^k Q_\beta
$$

(2.12)
where: \( P_a = \begin{bmatrix} C & CA & CA^2 & \ldots & CA^{n-1} \end{bmatrix}^T \), the observability matrix

\( Q_p = \begin{bmatrix} B & AB & A^2B & \ldots & A^{\beta-1}B \end{bmatrix} \), the controllability matrix

Using singular value decomposition, the Hankel matrix for \( k = 1 \) is factorized:

\[
H(0) = R\Sigma S^T
\]  

(2.13)

where:

\[
R_n^T R_n = I_n = S_n^T S
\]

\( I_n \) = identity matrix of order \( n \)

\( \Sigma \) is a rectangular matrix defined in the following manner:

\[
H(k-1) = \begin{bmatrix} \Sigma_n & 0 \\ 0 & 0 \end{bmatrix}
\]  

(2.14)

where:

\( \Sigma_n = \text{diag}[\sigma_1, \sigma_2, \ldots, \sigma_i, \sigma_{i+1}, \ldots, \sigma_n] \)

\( \sigma_i \) = non-zero singular values

Using (2.12) and (2.13), the system Markov parameters can be represented by the following expression [3]:

\[
Y_k = E_n^T R_n \Sigma_n^{1/2} \Sigma_n^{-1/2} R_n^T H(1) S_n \Sigma_n^{-1/2} S_n^{1/2} S_n^{1/2} S_n^T E_r
\]  

(2.15)

where:

\( E_n^T = \begin{bmatrix} I_m & O_m & \ldots & O_m \end{bmatrix} \)

\( E_r^T = \begin{bmatrix} I_m & O_r & \ldots & O_r \end{bmatrix} \)

\( O_n = \text{null matrix of order } n \)
Comparing the expression for the Markov parameters produced by OKID (2.9) and the alternate representation of (2.15) yields the realization of the discrete-time SI model:

\[ A = \Sigma_n^{-1/2} R_n^T H(1) S_n \Sigma_n^{-1/2} \]
\[ B = \Sigma_n^{1/2} S_n E_r \]
\[ C = E_m^T R_n \Sigma_n^{1/2} \]
\[ D = Y_o \quad (2.16) \]

This ERA-derived model can now serve, for example, as the "plant" in the design of an FSS control scheme, or as a basis for finding the dynamical properties of the system. The ERA technique has been successfully employed in many aerospace projects, including a modal survey of the Galileo spacecraft [9].
3.1 Overview of ProtoSim

The need for a simulation to model the DICE test article was identified early in the project [10]. Originally intended to test SI and controller methods, the uses for the simulator have grown as the DICE project moves forward. Among the tasks requiring a simulator are:

- Verifying that the DICE design can meet the research requirements.
- Testing the validity of simplifying assumptions.
- Developing momentum management techniques for the controllers.
- Estimating power consumption by post-processing of simulation data.
- Helping in the determination, selection, and sizing of components.
- Providing ground support during the on-orbit operation of DICE.
- Producing data for graphical animations.

In response to this necessity, a dynamical model of DICE was derived from first principles by Nguyen [11] with subsequently modifications by Crawford and Zee [12]. This model will be set up in SystemBuild by Zee in the future. To service near-term needs, an interim prototype DICE simulator called ProtoSim has been developed by the author. Intended primarily to support this thesis, ProtoSim will also serve be a useful basis for comparison and mutual verification when the first-principles DICE model is set up. It may also be used to support some of the other investigations listed above, if certain results are required before the full simulator is available.

ProtoSim consists of actuator and sensor models developed by the author connected to a model of the DICE dynamics. The actuators components consist of:

- Reaction wheels.
- Control moment gyros (CMG).

The sensor components are:

- Encoders for measuring the speed of the reaction and CMG wheels.
- Potentiometers for measuring the CMG gimbal angles.
- Rib tip sensors.
• Space Vision System (SVS) for detecting bus translation and attitude.
• Inertial rate sensor for measuring bus rotation rate.

At the heart of ProtoSim is a dynamical model of the DICE free-flyer. This model is a modified version of the Gyrosim block developed by S.R. Piggott [13] of Dynacon. This linear model of the DICE dynamics accepts torque inputs from the actuator models and passes the corresponding outputs to the appropriate sensor models.

3.2 The Gyrosim Dynamics Block

Gyrosim was originally developed as a data source for graphical renditions in 3-D Studio. It now serves as the dynamical plant for ProtoSim. Gyrosim's structural and stiffness properties were defined in NASTRAN and then imported into SystemBuild as a modal model. The key assumptions in the development of Gyrosim are:

• The origins of all the reference frames are collocated at the mass center.
• The rotor is axi-symmetric.
• The angular motions and rates are small.
• The CMG has only one active control axis, and the mass centers of the wheel and carrier are collocated and are in principal axis frames.

The development of the motion equations summarized here is from [13]. It is similar to that by P.C. Hughes for the \( ^\alpha + ^\beta \) gyrostat [15]. Applying this system to DICE, the scalar motion equations that model the DICE rib-tip package \( ^\beta \) with its spinning CMG wheel \( ^\alpha \) are:

\[
\begin{align*}
\mathbf{h} + \omega_t \mathbf{h} &= \mathbf{g}_e \\
\mathbf{h}_g + \mathbf{C}_w \omega_r + \mathbf{C}_w^T \mathbf{h}_g + \omega_w \mathbf{h}_g &= \mathbf{g}_c + \mathbf{g}_c + \mathbf{g}_s
\end{align*}
\]

(3.1)

where:

- \( \mathbf{h} \) = system net angular momentum
- \( \mathbf{g}_e \) = external torque on \( ^\alpha \)
- \( \mathbf{g}_c \) = control axis torque between \( ^\alpha \) and \( ^\beta \)
- \( \mathbf{g}_s \) = constraint torque between \( ^\alpha \) and \( ^\beta \)
\( \mathbf{g}_s = \) spin rate control torque between \( \mathcal{S} \) and \( \mathcal{S} \) \\
\( g = \) with respect to frame \( \mathcal{J}_g \) attached to gimbal rotating structure \\
\( w = \) with respect to frame \( \mathcal{J}_w \) attached to \( \mathcal{W} \) \\
\( r = \) with respect to frame \( \mathcal{J}_r \) attached to \( \mathcal{W} \)

Next, (3.1) is linearized. First, rates that multiply together are neglected since these are assumed to be small. The only rate that is not assumed constant is \( \omega_w \), the nominal spin rate of the wheel. Instead, it is set to \( \omega_w = v + \epsilon \), where \( v \) is constant and \( \epsilon \) is a small perturbation that is subject to linearization.

Second, it is assumed that the control angle the spin axis will be rotated about it small. Hence \( \mathbf{C}_{wr} = 1 - \gamma \), where \( \gamma \) is a column matrix of the angles. Since there are other small terms everywhere \( \mathbf{C}_{wr} \) occurs, \( \gamma \) does not appear in the final formulation. So, rewriting (3.1) with this substitution for \( \mathbf{C}_{wr} \) and expanding out the \( h \) terms to show the associated second moments of inertia \( \mathbf{J} \) yields the linearized motion equations:

\[
\mathbf{J}\dot{\omega}_r + \omega_w \mathbf{J}_g v + \mathbf{J}_g (\dot{\omega}_w + \dot{\epsilon}) + \omega_r \mathbf{J}_g v = g_c \\
\mathbf{J}_g (\dot{\omega}_w + \dot{\epsilon} + \omega_r) + \omega_r \mathbf{J}_g v + \omega_w \mathbf{J}_g v = g_c + g_i + g_s
\]

Define:

\[
\omega_r \triangleq [\dot{\alpha}_1, \dot{\alpha}_2, \dot{\alpha}_3]^T \\
\omega_w \triangleq [\dot{\gamma}, 0, 0]^T \\
v \triangleq [0, 0, v]^T \\
\epsilon \triangleq [0, 0, \epsilon]^T \\
g_c \triangleq [g_w, 0, 0]^T \\
g_i \triangleq [0, g_w, 0]^T \\
g_s \triangleq [0, 0, g_w]^T \\
\mathbf{J}_b \triangleq \text{diag}\{J_1, J_2, J_3\} \\
\mathbf{J}_w \triangleq \text{diag}\{I_w, I_{w}, I_{w}\}
\[ h_x = I_a \nu \]

Now, using the definitions of (3.3), the motion equations can be assembled in a matrix format:

\[
\begin{bmatrix}
J_1 & 0 & \bar{\alpha}_1 \\
0 & J_2 & \bar{\alpha}_2
\end{bmatrix}
+ h_x
\begin{bmatrix}
0 & 1 & \bar{\alpha}_1 \\
-1 & 0 & \bar{\alpha}_2
\end{bmatrix}
= \begin{bmatrix}
g_{c,1} \\
g_{c,2}
\end{bmatrix}
\]  \hspace{1cm} (3.4)

This is in the standard form \( m\ddot{x} + \chi x = f \), and it is what is implemented in Gyrosim. Piggott [13] notes that this agrees with the "gyrostat with a non-spinning carrier" expression by Hughes [16]:

\[
J\ddot{\alpha} + h_a \alpha \dot{\alpha} = g_c
\]  \hspace{1cm} (3.5)

The linear elastic properties of DICE were developed for Gyrosim under a NASTRAN finite element formulation. The hub was modeled as a single rigid element that is free to move in three axes of translation and rotation. Grid points were placed at the rib connection points with a concentrated mass element containing the inertia properties. Rib tips were also modeled as rigid bodies with a concentrated mass element spaced away from the rib end by an assumed mass center distance.

Given this construction, NASTRAN forms a linear model of the system dynamics in the form:

\[
m\ddot{q} + \chi \dot{q} = f
\]  \hspace{1cm} (3.6)

where: 
- \( m = n \times n \) system inertia matrix
- \( \chi = n \times n \) "stiffness" matrix
- \( f = n \times 1 \) matrix of generalized forces
\[ \mathbf{q} = n \times 1 \text{ matrix of generalized coordinates} \]

For Gyrosim, the eigensolution of (3.6) was found to extract the modal properties for use as the basis of the SystemBuild model:

\[ (-\lambda \mathbf{m} + \mathbf{c})\mathbf{e} = \mathbf{0} \quad (3.7) \]

where: \( \lambda_i \) = the \( i \)th eigenvalue  
\[ e_i = \text{the } i\text{th eigenvector} \]

NASTRAN then computes the mass-normalized eigenvectors. These satisfy:

\[ \mathbf{e}^T \mathbf{m} \mathbf{e} = 1 \quad (3.8) \]
\[ \mathbf{e}^T \mathbf{c} \mathbf{e} = \Omega^2 \]

where: \( \mathbf{e} = n \times n \) matrix of eigenvectors  
\[ \Omega^2 = \text{diag}\{\omega_i^2\}, \text{ the squared modal frequencies} \]

Given the identities of (3.8), a relationship between the generalized coordinates \( \mathbf{q} \) and a set of modal coordinates \( \mathbf{\eta} \) can be written:

\[ \mathbf{q} = \mathbf{e} \mathbf{\eta} \quad (3.9) \]

Pre-multiplying (3.6) with \( \mathbf{e}^T \) and substituting (3.9) results in the decoupled motion equation:

\[ \mathbf{\dot{\eta}} + \Omega^2 \mathbf{\eta} = \mathbf{e}^T \mathbf{f} \quad (3.10) \]

In the original version of Gyrosim, (3.10) included a gyricity term. Since in ProtoSim
Gyricity is provided by the CMG models, this term was removed from the SystemBuild formulation. Finally, the original SystemBuild simulation algorithm included a Schur decomposition to extract information on the gyrnic modes. Since these are also not needed by ProtoSim, this segment was removed. This modification was done by R.E. Zee, and thus the dynamical block in ProtoSim is labelled REZGyroSim to differentiate it from Piggott's work. Note that in Zee's implementation, a handy addition is made to (3.10) and (3.9):

\[
\bar{\eta} + \Omega^2 \eta = \psi^T B_o f \\
q = C_o \psi \eta
\]  

(3.11)

where: \(B_o, C_o\) = input and output distribution matrices respectively which, depending on whether there is a 1 or 0 on the diagonal, selects the rows of input or output to be used (the default for both is the identity matrix 1).

The REZGyroSim block is shown in Figure 3.1, and the inputs and outputs for this block are summarized in Table 3.1.

---

1In terms of correctness, Gyrosim without gyrnicity cannot really be called "Gyrosim". However, for the sake of clarity, this rather catchy name will be retained here.
Figure 3.1: The REZGyroSim block of the DICE dynamics.
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Units</th>
<th>Source/Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>1-3  hub x,y,z forces</td>
<td>N</td>
<td>Air_Thruster_Forces</td>
</tr>
<tr>
<td></td>
<td>4-6  hub x,y,z torques</td>
<td>Nm</td>
<td>reaction_wheels</td>
</tr>
<tr>
<td></td>
<td>7-9  rib #1 x,y,z torques</td>
<td>Nm</td>
<td>cmg</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td></td>
<td>19-21 rib #5 x,y,z torques</td>
<td>Nm</td>
<td>cmg</td>
</tr>
<tr>
<td>Outputs</td>
<td>1-3  hub x,y,z translations</td>
<td>m</td>
<td>svs</td>
</tr>
<tr>
<td></td>
<td>4-6  hub x,y,z rotations</td>
<td>rad</td>
<td>svs</td>
</tr>
<tr>
<td></td>
<td>7-9  rib #1 x,y,z translations</td>
<td>m</td>
<td>Rib_Trans_Relative</td>
</tr>
<tr>
<td></td>
<td>10-12 rib #1 x,y,z rotations</td>
<td>rad</td>
<td>Rib_Rot_Relative</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td></td>
<td>31-33 rib #5 x,y,z translations</td>
<td>m</td>
<td>Rib_Trans_Relative</td>
</tr>
<tr>
<td></td>
<td>34-36 rib #5 x,y,z rotations</td>
<td>rad</td>
<td>Rib_Rot_Relative</td>
</tr>
<tr>
<td></td>
<td>37-39 hub x,y,z translation rates</td>
<td>m/s</td>
<td>external</td>
</tr>
<tr>
<td></td>
<td>40-42 hub x,y,z rotation rates</td>
<td>rad/s</td>
<td>rate_sensor</td>
</tr>
<tr>
<td></td>
<td>43-45 rib #1 x,y,z translation rates</td>
<td>m/s</td>
<td>external</td>
</tr>
<tr>
<td></td>
<td>46-48 rib #1 x,y,z rotation rates</td>
<td>rad/s</td>
<td>external</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td></td>
<td>67-69 rib #5 x,y,z translation rates</td>
<td>m/s</td>
<td>external</td>
</tr>
<tr>
<td></td>
<td>70-72 rib #5 x,y,z rotation rates</td>
<td>rad/s</td>
<td>external</td>
</tr>
</tbody>
</table>

Table 3.1: The inputs and outputs of the REZGyroSim DICE dynamics block.

3.3 Transformation from Absolute to Relative Coordinates

The structural mass and stiffness properties of Gyrosim were defined with NASTRAN. Like many finite element packages, NASTRAN uses an absolute coordinate system. This means that the grid point displacements are measured with respect to fixed inertial positions. While such an arrangement simplifies the finite element formulation, it is not suitable for ProtoSim.
As illustrated in Figure 3.2, Gyrosim employs six such absolute frames. Hub translation and rotation are taken with respect to the hub frame $\mathcal{J}_{\text{hub}}$. This frame has its origin at the geometric center of the hub, with the x-axis parallel to the un-deformed axis of rib #1 and the z-axis normal to the rib plane. Tip movements are taken with respect to five tip frames $\mathcal{J}_{\text{rib}}$ one for each rib. Here, the x-axis is along the rib and the y-axis points to the next rib in a counterclockwise fashion.

Since SVS will be seeing the motion of DICE from a fixed location (relative to the Shuttle middeck), the Gyrosim outputs of hub translation and rotation can be passed straight to ProtoSim's SVS model. Unfortunately, this is not so for the ribs. In the real DICE, the rib motions will be sensed by the rib-tip sensors located in the moving hub. So before these Gyrosim outputs can be used by the sensor models, they must be transformed from absolute to relative coordinates.

Consider the vector $\mathbf{a}$ to the arbitrary point $\mathbf{P}$ in Figure 3.3, expressed in the absolute rib-tip frame $\mathcal{J}_{\text{rib}}$. It can be written as a combination of the other vectors as follows:

$$\mathbf{a} = -\mathbf{b} + \mathbf{c} + \mathbf{d} + \mathbf{e} + \mathbf{f} + \mathbf{a}'$$  \hspace{1cm} (3.12)

Solving (3.12) for $\mathbf{a}'$, the vector to $\mathbf{P}$ expressed in the relative frame $\mathcal{J}_{\text{rib}}$ yields:

$$\mathbf{a}' = \mathbf{a} + \mathbf{b} + \mathbf{c} - \mathbf{d} - \mathbf{e} - \mathbf{f}$$  \hspace{1cm} (3.13)
Explicitly expressing each vector in its respective frame, (3.13) becomes:

\[
\mathbf{J}_{\text{rib}} ^T \mathbf{a}_{\text{rib}} = \mathbf{J}_{\text{rib}} ^T \mathbf{a}_{\text{rib}} + \mathbf{J}_{\text{rib}} ^T \mathbf{b}_{\text{rib}} + \mathbf{J}_{\text{rib}} ^T \mathbf{c}_{\text{rib}} - \mathbf{J}_{\text{rib}} ^T \mathbf{d}_{\text{hub}} - \mathbf{J}_{\text{rib}} ^T \mathbf{e}_{\text{rib}} - \mathbf{J}_{\text{rib}} ^T \mathbf{f}_{\text{rib}} 
\]  

(3.14)

Multiplying both sides of (3.14) by \( \mathbf{J}_{\text{rib}} \) results in:

\[
\mathbf{a}'_{\text{rib}} = \mathbf{J}_{\text{rib}} \mathbf{a}_{\text{rib}} + \mathbf{J}_{\text{rib}} \mathbf{b}_{\text{rib}} + \mathbf{J}_{\text{rib}} \mathbf{c}_{\text{rib}} - \mathbf{J}_{\text{rib}} \mathbf{d}_{\text{hub}} - \mathbf{J}_{\text{rib}} \mathbf{e}_{\text{rib}} - \mathbf{J}_{\text{rib}} \mathbf{f}_{\text{rib}} 
\]

\[
= \mathbf{J}_{\text{rib}} \mathbf{J}_{\text{rib}} ^T \mathbf{a}_{\text{rib}} + \mathbf{J}_{\text{rib}} \mathbf{b}_{\text{rib}} + \mathbf{J}_{\text{rib}} \mathbf{c}_{\text{rib}} - \mathbf{J}_{\text{rib}} \mathbf{d}_{\text{hub}} - \mathbf{J}_{\text{rib}} \mathbf{e}_{\text{rib}} - \mathbf{J}_{\text{rib}} \mathbf{f}_{\text{rib}} 
\]

\[
= \mathbf{C}_{\text{rib,lhub}} \mathbf{c}_{\text{hub,lhub}} \mathbf{C}_{\text{hub,rib}} \mathbf{a}_{\text{rib}} + \mathbf{b}_{\text{rib}} + \mathbf{c}_{\text{rib}} - \mathbf{C}_{\text{rib,lhub}} \mathbf{C}_{\text{hub,lhub}} \mathbf{d}_{\text{hub}} - \mathbf{e}_{\text{rib}} - \mathbf{f}_{\text{rib}} 
\]

(3.15)

Factoring (3.15) results in the final form that was incorporated into ProtoSim:

\[
\mathbf{a}'_{\text{rib}} = \mathbf{C}_{\text{rib,lhub}} \mathbf{c}_{\text{hub,lhub}} \mathbf{C}_{\text{hub,rib}} \left[ \mathbf{C}_{\text{hub,rib}} \mathbf{a}_{\text{rib}} + \mathbf{b}_{\text{rib}} + \mathbf{c}_{\text{rib}} - \mathbf{d}_{\text{hub}} \right] - \mathbf{e}_{\text{rib}} - \mathbf{f}_{\text{rib}} 
\]

(3.16)

where: \( \mathbf{a}_{\text{rib}} \) = absolute rib translation, from Gyrosim (m)

\( \mathbf{b}_{\text{rib}}, \mathbf{f}_{\text{rib}} \) = rib length (0.4763 m)

\( \mathbf{c}_{\text{rib}}, \mathbf{e}_{\text{rib}} \) = hub radius (0.0762 m)

\( \mathbf{d}_{\text{hub}} \) = hub translation, from Gyrosim (m)

\( \mathbf{C}_{\text{rib,lhub}} \) = rotation matrix, \(-((i-1)2\pi)/5\) about z-axis

\( \mathbf{C}_{\text{hub,lhub}} \) = rotation matrix, using attitude angles from Gyrosim

\( \mathbf{C}_{\text{hub,rib}} \) = rotation matrix, \((i-1)2\pi)/5\) about z-axis

\( i = \) rib number \( (1,2,\ldots,5) \)

A similar expression transforms the rib rotations from absolute to relative coordinates:

\[
\mathbf{r}'_{\text{rib}} = \mathbf{C}_{\text{rib,lhub}} \mathbf{c}_{\text{hub,lhub}} \mathbf{C}_{\text{hub,rib}} \left[ \mathbf{C}_{\text{hub,rib}} \mathbf{r}_{\text{rib}} - \mathbf{h}_{\text{hub}} \right] 
\]

(3.17)

where: \( \mathbf{r}_{\text{rib}} \) = absolute rib rotation,
from Gyrosim (rad)

\[ h_{\text{hub}} = \text{hub rotation, from} \]

Gyrosim (rad)

Since the rotation matrices \( C_{\text{hub},\text{rib}} \) and \( C_{\text{rib},\text{hub}} \) are constant for a given rib, separate pairs of blocks that carry out (3.16) and (3.17) were prepared for each. This is more numerically efficient since the angles need not be recalculated at each time step. Figures 3.4 and 3.5 show the translation and rotation transformation blocks respectively for rib #1. The associated inputs and outputs are summarized in Tables 3.2 and 3.3.

Figure 3.3: Vectors expressed in the absolute and relative rib-tip frames.
Figure 3.4: The absolute-to-relative Transformation block for rib #1 deflections.

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Units</th>
<th>Source/Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>1-3 rib x,y,z absolute translation</td>
<td>m</td>
<td>Gyrosim</td>
</tr>
<tr>
<td></td>
<td>4-6 hub x,y,z absolute rotation</td>
<td>rad</td>
<td>Gyrosim</td>
</tr>
<tr>
<td></td>
<td>7-9 hub x,y,z absolute translation</td>
<td>m</td>
<td>Gyrosim</td>
</tr>
<tr>
<td>Output</td>
<td>1-3 rib x,y,z relative translation</td>
<td>m</td>
<td>lepd (rib tip sensor)</td>
</tr>
</tbody>
</table>

Table 3.2: The inputs and outputs of the Transformation block.
3.4 Actuator Models

3.4.1 Reaction Wheels

ProtoSim's reaction wheel component consists of two parts. The first is an electro-mechanical model of a current-controlled spin motor. The output of this spin motor is then connected to a block that models the dynamics of the wheel itself.

For the reaction wheel spin motor, the load was assumed to consist of an inertia $I$ and a
damper with damping constant $B$. Motor shaft rate $\omega$ and developed motor torque $\tau$ were taken to be related by:

$$\tau(t) = J\dot{\omega}(t) + B\omega(t) + \tau_f(t)$$  \hspace{1cm} (3.18)

In the frequency domain, this is:

$$\tau = (Js + B)\omega + \tau_f(s)$$  \hspace{1cm} (3.19)

Since the friction torque $\tau_f(s)$ is not generally linear, it is not well defined for this definition. Therefore, it is dropped from this development and is accounted for later in time-domain modeling. For direct current-control, the back EMF and inductance effects are also neglected. Therefore, the developed motor torque $\tau$ is taken to be proportional to the applied current:

$$\tau = K_i i$$  \hspace{1cm} (3.20)

(3.19) and (3.20) yielded the desired transfer function between the applied current $i$ and shaft rate $\omega$:

$$\frac{\omega}{i} = \frac{K_i}{Js + B}$$  \hspace{1cm} (3.21)

Note that the inertia $J$ and the damping constant $B$ in (3.21) must include both the indigenous properties of the motor itself and the "reflected" characteristics of the reaction wheel attached to the shaft. These terms can be broken down into the following components:

$$J = J_m + n^2 J_{rw}$$

$$B = B_m + n^2 B_{rw}$$  \hspace{1cm} (3.22)
where: $J_m =$ inertia of the motor (kg·m²)
$J_{rw} =$ inertia of the reaction wheel (kg·m²)
$B_m =$ damping constant of the motor (Nms)
$B_{rw} =$ damping constant of the reaction wheel (Nms)
$n =$ gear ratio

Having established an electro-mechanical representation of the spin motor, the dynamical equations for the wheel dynamics must be developed. Consider a spinning wheel mounted on a platform whose angular velocity relative to an inertial frame is $\omega$. From classical mechanics, the angular momentum of this wheel is:

$$\mathbf{h}_s = I_s \omega_s \mathbf{a}$$  \hspace{1cm} (3.23)

where: $I_s =$ moment of inertia of the wheel about its spin axis (kg·m²)
$\omega_s =$ wheel spin rate (rad/s)
$\mathbf{a} =$ unit vector defining the spin axis

The torque $\mathbf{g}$ is the time derivative of (3.23) or:

$$\mathbf{g} = I_s \dot{\omega}_s \mathbf{a} + I_s \omega_s \dot{\mathbf{a}}$$
$$= I_s \dot{\omega}_s \mathbf{a} + I_s \omega_s (\dot{\mathbf{a}} + \omega \times \mathbf{a})$$ \hspace{1cm} (3.24)

For the reaction wheels the axes are fixed, so $\dot{\mathbf{a}} = 0$ and (3.24) becomes:

$$\mathbf{g}_{rw} = I_s \dot{\omega}_s \mathbf{a} + \omega \times I_s \omega_s \mathbf{a}$$ \hspace{1cm} (3.25)

(3.25) shows that by increasing or decreasing the spin rate, the wheel will generate a torque $I_s \dot{\omega}_s \mathbf{a}$ about the spin axis and a torque $\omega \times I_s \omega_s \mathbf{a}$ perpendicular to the spin axis. The latter term arises from the momentum storage in the wheel, and is treated as part of the system dynamics. Note that
it depends not only on the wheel characteristics $I_s$ and $\omega_s$ but also on $\omega$, the angular velocity of the platform (here, the DICE hub).

Figure 3.6 shows the block diagram for ProtoSim's reaction wheel model. Note that this time-domain implementation contains two components not found in the preceding frequency-domain derivation. The first is the block labelled friction, which represents the nonlinear friction term $\tau(t)$ of (3.18). This block, constructed by D.J. McTavish and J.M. Crawford of Dynacon, implements D.A. Haessig and B. Friedland's "reset-integrator" friction model [17]. The second is the block Systemic_Noise, which serves as a source of uniform random disturbance torque to the model. Both blocks can be activated and deactivated by the user as required.

As output, the motor block produces a wheel speed $\omega$ that is fed to rw_dynamics, which implements (3.25). This block is shown in Figure 3.7, and the inputs and outputs of both this and the parent reaction_wheel block are summarized in Tables 3.4 and 3.3 respectively. Note that all three of DICE's hub reaction wheels are set up together as separate streams in the same blocks.
Figure 3.6: The reaction_wheel block.

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Units</th>
<th>Source/Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>command currents $i_x, i_y, i_z$</td>
<td>$A$</td>
<td>user (external)</td>
</tr>
<tr>
<td></td>
<td>bus rotation rate $\omega(x,y,z)$</td>
<td>rad/s</td>
<td>Gyrosim</td>
</tr>
<tr>
<td></td>
<td>bus attitude angles $\theta, \phi, \Psi$</td>
<td>rad</td>
<td>Gyrosim</td>
</tr>
<tr>
<td>Outputs</td>
<td>generated torque $g_{w}(x,y,z)$</td>
<td>Nm</td>
<td>Gyrosim</td>
</tr>
<tr>
<td></td>
<td>wheel speed $\omega(x,y,z)$</td>
<td>rad/s</td>
<td>$rw_encoder$</td>
</tr>
</tbody>
</table>

Table 3.4: The inputs and outputs of the reaction_wheel block.
Continuous SuperBlock | Inputs | Outputs
---|---|---
rw_dynamics | 12 | 3

![Diagram of rw_dynamics block]

**Figure 3.7:** The reaction wheel dynamics block rw_dynamics.

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Units</th>
<th>Source/Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>1-3</td>
<td>bus attitude angles $\theta, \phi, \Psi$</td>
<td>rad</td>
</tr>
<tr>
<td></td>
<td>4-6</td>
<td>bus rotation rate $\omega(x,y,z)$</td>
<td>rad/s</td>
</tr>
<tr>
<td></td>
<td>7-9</td>
<td>wheel speed $\omega_s(x,y,z)$</td>
<td>rad/s</td>
</tr>
<tr>
<td></td>
<td>10-12</td>
<td>wheel acceleration $\dot{\omega}_s(x,y,z)$</td>
<td>rad/s$^2$</td>
</tr>
<tr>
<td>Output</td>
<td>1-3</td>
<td>generated torque $g_{in}(x,y,z)$</td>
<td>Nm</td>
</tr>
</tbody>
</table>

**Table 3.5:** The inputs and outputs of the rw_dynamics block.
3.4.2 Control Moment Gyros

Like the reaction wheel spin motor, the CMG gimbal motor is current controlled. Since the required output here is the commanded angle $\theta$, the transfer function is simply the integral of (3.21):

$$\frac{\theta}{i} = \frac{K_i}{s(Js + B)} \quad (3.26)$$

Note that analogous to (3.22), the inertia $J$ and the damping $B$ in (3.26) represent the total value for both the motor and the CMG assembly. The block diagram for the CMG gimbal motor is similar to that of the reaction wheel spin motor, and is shown in Figure 3.8. Note the addition of the *Gimbal Limit* block, which limits the pivoting of the CMG to that prescribed by the actual design. The inputs and outputs to this block are summarized in Table 3.6.

<table>
<thead>
<tr>
<th>Continuous SuperBlock</th>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>gimbal_motor</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

Figure 3.8: The CMG *gimbal_motor* block.
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Units</th>
<th>Source/Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>command current $i_c$</td>
<td>A</td>
<td>user (external)</td>
</tr>
<tr>
<td>Outputs</td>
<td>gimbal rate $\dot{\theta}$</td>
<td>rad/s</td>
<td>cmg_dynamics</td>
</tr>
<tr>
<td></td>
<td>gimbal angle $\theta$</td>
<td>rad</td>
<td>cmg_dynamics</td>
</tr>
<tr>
<td></td>
<td>gimbal acceleration $\ddot{\theta}$</td>
<td>Nm</td>
<td>cmg_dynamics</td>
</tr>
</tbody>
</table>

Table 3.6: The inputs and outputs of the CMG gimbal motor block.

In contrast to the gimbal motors, the CMG spin motors are voltage-controlled. For such a motor, the armature voltage loop is described in the time domain by:

$$v_a = R_a i_a + L_a \dot{i_a} + v_{emf}$$  \hspace{1cm} (3.27)

Doing a Laplace transform into the frequency domain, (3.27) becomes:

$$V_a = (R_a + L_a s)i_a + V_{emf}$$  \hspace{1cm} (3.28)

Unlike the current-controlled motors, the effect of back EMF voltage cannot be neglected in voltage-controlled motors. Here, it is taken to be proportional to the shaft speed:

$$V_{emf} = K_i \omega$$  \hspace{1cm} (3.29)

Combining (3.28) and (3.29) with Equations (3.19) and (3.20) from Section 3.4.1 yields:

$$V_a = (K_m + \frac{R_a B}{K_m} + \frac{R_a J + L_a B}{K_m}s + \frac{L_a J}{K_m}s^2)\omega + (\frac{R_a + L_a s}{K_m})\tau_f$$  \hspace{1cm} (3.30)

Again, since $\tau_f$ is not generally linear, it is not well-defined in this formulation and is therefore better accounted for through time domain modeling. Note that this is a second-order
transfer function. It is highly overdamped, so the poles are real and well separated. To avoid very small step sizes during simulation and to improve numerical efficiency, the high frequency pole was deleted by setting the inductance $L_s = 0$. So, the desired linear transfer function between the applied voltage $V_a$ and the shaft rate $\omega$ is:

$$\frac{\omega}{V_a} = \frac{K_m}{K_m^2 + R_o(Js + B)}$$

(3.31)

As with the previous motor models, the inertia $J$ and the damping constant $B$ in (3.31) includes both the properties of the motor itself and the "reflected" characteristics of the CMG assembly attached to it:

$$J = J_m + n^2 J_{cmg}$$
$$B = B_m + n^2 B_{cmg}$$

(3.32)

where: $J_m =$ inertia of the motor (kg⋅m²)

$J_{cmg} =$ inertia of the CMG assembly (kg⋅m²)

$B_m =$ damping constant of the motor (Nms)

$B_{cmg} =$ damping constant of the CMG assembly (Nms)

$n =$ gear ratio

The block diagram for the voltage-controlled CMG spin motor is shown in Figure 3.8. As with the previous motor models, the time-domain implementation includes a block to implement the friction term $\tau_f$ in (3.30) with a "reset-integrator" system. The inputs and outputs to the spin_motor block are summarized in Table 3.7.
The development of the CMG dynamics model is similar to that for the reaction wheel dynamics of Section 3.4.1. Starting with (3.24), and noting that $\\mathbf{a} = \boldsymbol{\Omega} \times \mathbf{a}$, the basic torque expression becomes:

$$g = I_\omega \omega \times a + I_\omega (\boldsymbol{\Omega} \times a + \omega \times a)$$

(3.33)

A further term must be added to (3.33) to allow for the torque resulting from the rotation of the spin motor assembly relative to the CMG housing. This torque is approximately given by [14]:

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Units</th>
<th>Source/Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>1 command voltage $V_c$</td>
<td>V</td>
<td>user (external)</td>
</tr>
<tr>
<td>Outputs</td>
<td>1 wheel speed $\omega_x$</td>
<td>rad/s</td>
<td>cmg_dynamics</td>
</tr>
<tr>
<td></td>
<td>2 wheel acceleration $\dot{\omega}$</td>
<td>rad/s$^2$</td>
<td>cmg_dynamics</td>
</tr>
</tbody>
</table>

Table 3.7: The inputs and outputs of the CMG spin_motor block.
\[ g - I_\omega \dot{a} - I_\omega (\omega \times a) - I_\omega b \] (3.35)

The \textit{SystemBuild} block implementation of (3.36) for ProtoSim is shown in Figure 3.10. Note that ProtoSim actually has two CMG dynamics blocks. As illustrated in Figure 3.2, the CMGs on ribs #1, #3, and #4 are gimballed about the z-axis of the rib frame \( J_{\text{rb}} \), while the other two CMGs are actuated about the y-axis. This difference is incorporated in the block labelled \textit{Gimbal Axis} in Figure 3.10. The inputs and outputs for this block are summarized in Table 3.8.
Figure 3.10: The `cmg_dynamics` blocks for ribs 1, 3, and 5.

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Units</th>
<th>Source/Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-3</td>
<td>rib tip attitude angles θ, φ, ψ</td>
<td>rad</td>
<td>Gyrosim</td>
</tr>
<tr>
<td>4-6</td>
<td>rib tip rotation rate ω(x,y,z)</td>
<td>rad/s</td>
<td>Gyrosim</td>
</tr>
<tr>
<td>7</td>
<td>spin motor speed ω</td>
<td>rad/s</td>
<td>spin_motor</td>
</tr>
<tr>
<td>8</td>
<td>gimbal rate θ₁</td>
<td>rad/s</td>
<td>gimbal_motor</td>
</tr>
<tr>
<td>9</td>
<td>gimbal angle θ₁</td>
<td>rad</td>
<td>gimbal_motor</td>
</tr>
<tr>
<td>10</td>
<td>gimbal acceleration θ₁</td>
<td>rad/s²</td>
<td>gimbal_motor</td>
</tr>
<tr>
<td>11</td>
<td>spin motor acceleration ω₁</td>
<td>rad/s²</td>
<td>spin_motor</td>
</tr>
<tr>
<td>Outputs</td>
<td>1-3 generated torque g_{cmg}(x,y,z)</td>
<td>Nm</td>
<td>Gyrosim</td>
</tr>
</tbody>
</table>

Table 3.8: The inputs and outputs of the `cmg_dynamics` block.

Finally, these three individual components -- `gimbal_motor`, `spin_motor`, and `cmg_dynamics` -- were connected into an integrated package for each of the five CMG units. One such unit, the model for ribs #1, #3, and #4, is shown in Figure 3.11, and the associated inputs and
outputs are summarized in Table 3.9. The CMG unit model for ribs #2 and #5 are identical except for the Gimbal Axis parameter in the cmg_dynamics block.

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Units</th>
<th>Source/Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>gimbal motor command current $i_c$</td>
<td>A</td>
<td>user (external)</td>
</tr>
<tr>
<td>2-4</td>
<td>rib tip attitude angles $\theta, \phi, \Psi$</td>
<td>rad</td>
<td>Gyrosim</td>
</tr>
<tr>
<td>5-7</td>
<td>rib tip rotation rate $\omega(x,y,z)$</td>
<td>rad/s</td>
<td>Gyrosim</td>
</tr>
<tr>
<td>8</td>
<td>spin motor activation voltage $V_s$</td>
<td>rad/s</td>
<td>user (external)</td>
</tr>
<tr>
<td>Outputs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-3</td>
<td>generated torque $g_{\text{eq}}(x,y,z)$</td>
<td>Nm</td>
<td>Gyrosim</td>
</tr>
<tr>
<td>4</td>
<td>wheel speed $\omega_s$</td>
<td>rad/s</td>
<td>cmg_encoder</td>
</tr>
<tr>
<td>5</td>
<td>gimbal angle $\theta_g$</td>
<td>rad</td>
<td>potentiometer</td>
</tr>
</tbody>
</table>

Table 3.9: The inputs and outputs of the cmg_unit block.

Figure 3.11: The integrated cmg_unit block for ribs 1, 3, and 5.
ProtoSim models the DICE sensor suite as transducers. First, the raw output of Gyrosim is considered ideal. Then, the characteristics of the sensor are added to degrade this ideal output. These characteristics may include noise (which is assumed to be gaussian), drift, accuracy, bias, latency, and time delays. Other sensor-specific factors such as field-of-view, detection limits, and thresholds may also be modeled. Finally, the result is discretized and sampled based on the resolution and sampling rate of the actual sensor.

### 3.5.1 Encoders

The current baseline DICE design employs encoders to measure the speeds of both the CMG and reaction wheels. Velocity estimation is accomplished by differentiating the raw position signal of the encoder. For the DICE integrated prototype model, the encoder output goes into a pulse-counting port of the HC-11 embedded processor. The number of pulses accumulated during a specified period is counted and logged. Special post-processing is employed to detect direction changes in the wheel.

ProtoSim's encoder models emulate the resolution, frame rate, and time delay characteristics of the actual hardware. If required for a certain simulation or study, uniform random noise of a specified amplitude can be added by the user as required. Frame rate, time delay, and pulses per revolution are either taken from the specifications of the unit or assumed if no data was available. The resolution of the encoder is approximated by:

\[
\text{resolution} = \frac{2\pi}{\text{pulses per revolution}}
\]  

(3.37)

Structurally, the blocks for ProtoSim's encoder models are identical for both the reaction wheels and the CMGs. Figure 3.12 shows the blocks for the cmg_encoder, and Table 3.10 summarizes the inputs and outputs for this model.
3.5.2 Inertial Rate Sensor

DICE employs solid-state quartz rate sensors to measure the rotation rate of the hub. The unit currently baselined is the Systron-Donner QRS-II. It takes as input the rate about an axis and outputs an analog voltage signal. According to the unit's specifications, its maximum deviation from linearity is on the order of $10^{-4}$ rad/s [17]. Thus, the relationship between the input rate and the output voltage can be taken to be linear:

$$V_{irs} = K_{irs} \dot{\theta}_{hub}$$

(3.38)

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Units</th>
<th>Source/Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>1-5 raw wheel speeds $\omega_s$ for each CMG</td>
<td>rad/s</td>
<td>cmg_unit</td>
</tr>
<tr>
<td>Output</td>
<td>1-5 sensed wheel speed $\omega_s$ for each CMG</td>
<td>Nm</td>
<td>external</td>
</tr>
</tbody>
</table>

Table 3.10: The inputs and outputs of the encoder block.
where: \( K_u \) = inertial rate sensor scale factor term (Vs/rad)  
\( \dot{\theta}_{hub} \) = hub rotation rate about an axis (rad/s)

The block diagram of ProtoSim's rate_sensor model is shown in Figure 3.13, and its inputs and outputs are summarized in Table 3.11. Quantities for the systemic bias, detection limits, time delay, resolution, and sampling rate were either taken from the unit's specifications, or assumed if no data was available. The model also has a block for injecting uniform random noise into the sensor if such a capability is required by the user.

**Table 3.11:** The inputs and outputs of the rate_sensor block.

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Units</th>
<th>Source/Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>1-3</td>
<td>rad/s</td>
<td>Gyrosim</td>
</tr>
<tr>
<td></td>
<td>hub rotation rate ( \dot{\theta}_{hub} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>1-3</td>
<td>Nm</td>
<td>external</td>
</tr>
<tr>
<td></td>
<td>analog voltage signal ( V_{irs} )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3.5.3 Rib Tip Sensors

For the DICE mission, it is necessary to sense the displacement of all five rib tips to find the state of the flexible structure. A method will be employed that involves placing a light emitting diode (LED) at the tip of each rib. Its position relative to the bus module is then determined using an optical detector array based on a lateral effect photo-diode (LEPD). When coupled with the necessary electronics, this system can resolve the y- and z-axis position of the centroid of light incident on the detector array (recall from Section 3.3 that in the relative rib tip frame $\mathcal{F}_{\text{rib}}$, the x-axis points out along the rib).

Since the DICE geometry is fixed and the LEDs at the ribs will not appreciably move closer to their respective optical sensors, a single lens element is all that is required to focus the light on the LEPD array. The unidirectional field-of-view of the rib tip sensor can be calculated with the "lensmaker's equation":

$$\frac{1}{p} + \frac{1}{q} = \frac{1}{f}$$  \hspace{1cm} (3.39)

where: $p =$ object distance (m)
$q =$ image distance (m)
$f =$ focal length (m)

The magnification factor $m$ is defined as:

$$m = \frac{P}{q} \rightarrow q = mp$$  \hspace{1cm} (3.40)

Substituting the expression for $q$ from (3.40) into (3.39) and rearranging the result yields an expression for $m$ in terms of $p$ and $f$: 
\[ m = \frac{f}{p - f} \]  
\[ (3.41) \]

Both \( p \) and \( f \) are known. The former is simply the length of the rib, and the latter is given by the lens specifications. Given the size of the LEPD array \( L \), a third expression for the magnification factor \( m \) is:

\[ m = \frac{L}{2(fov)} \]  
\[ (3.42) \]

Equating (3.42) and (3.41) yields the final expression for the field-of-view is:

\[ fov = \frac{L(p - f)}{2f} \]  
\[ (3.43) \]

Field-of-view restrictions on the sensor are set up as a limiter, as shown in the block diagram of Figure 3.14. The output of the sensor is an analog voltage scaled with respect to the field-of-view. Accuracy, latency, resolution, and sampling rate are included in the model based either on specified parameters or assumed values if the required figure was not available. The inputs and outputs to this block are summarized in Table 3.12.
Figure 3.14: The lepd rib tip sensor block.

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Units</th>
<th>Source/Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>y,z deflections for each rib</td>
<td>m</td>
<td>Rib_Trans_Relative</td>
</tr>
<tr>
<td>Output</td>
<td>analog voltage signal</td>
<td>V</td>
<td>external</td>
</tr>
</tbody>
</table>

Table 3.12: The inputs and outputs of the lepd rib tip sensor block.

3.5.4 Potentiometers

Potentiometers measure the gimbal angles of the DICE CMGs. These devices employ a variable resistor whose resistance changes with angle. For a maximum CMG gimbal deflection of $\pm \theta_{\text{lim}}$, the potentiometer resistance varies from 0 at the negative limit to some maximum $R_{\text{max}}$ at the positive limit. For ProtoSim's potentiometer model, this relationship was taken to be linear, with the constant as:

$$K_p = \frac{R_{\text{max}}}{2 \theta_{\text{max}}}$$

(3.44)
The reference resistance $R_o$ is given by:

$$R_o = R_{\text{max}} - K_p \theta_{\text{max}}$$  \hspace{1cm} (3.45)

The expression for the resistance of the potentiometer as a function of the CMG gimbal angle is:

$$R(\theta) = K_p \theta + R_o$$  \hspace{1cm} (3.46)

Given the resistance $R(\theta)$ from (3.46), the output can either be an analog current signal based on a fixed reference voltage or a voltage signal based on a reference current. Since a fixed current is more robust to noise, the latter technique will be used on DICE. Therefore, the current put across the resistor is modeled as a gain in the potentiometer block of Figure 3.15. The values for the time delay, resolution, and sampling rate are taken either from specifications or assumed of data was not available. Table 3.13 summarizes the inputs and outputs of the potentiometer block.

<table>
<thead>
<tr>
<th>Continuous SuperBlock</th>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Potentiometer</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>

**Figure 3.15:** The CMG gimbal angle potentiometer block.
3.5.5 Space Vision System

The Space Vision System (SVS) is a computerized tracking system developed by the National Research Council and Neptec Corporation. SVS uses a television camera to monitor photogrammetry targets arranged on the object of interest. These targets consist of white dots on a black background whose geometries are known \textit{a priori}. As the object moves, the camera records the changing positions of the dots. Based on this information, the SVS computer determines the location and orientation of the object.

SVS will be used to sense the translation and rotation of the DICE bus module. ProtoSim's \texttt{svs} block, shown in Figure 3.16, emulates the frame rate, latency, resolution, and position/angular accuracy of the actual system. Where actual parameters were not available, assumed values were used. Table 3.14 summarizes the inputs and outputs of this sensor model.

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Units</th>
<th>Source/Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>1-5 gimbal angle $\theta$ for each CMG</td>
<td>rad</td>
<td>cmg_unit</td>
</tr>
<tr>
<td>Output</td>
<td>1-5 analog voltage signal</td>
<td>V</td>
<td>external</td>
</tr>
</tbody>
</table>

\textbf{Table 3.13:} The inputs and outputs of the \texttt{potentiometer} block.

![Figure 3.16: The \texttt{svs} block that emulates the Space Vision System.](image)
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Units</th>
<th>Source/Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>1-3 raw x,y,z bus translation</td>
<td>m</td>
<td>Gyrosim</td>
</tr>
<tr>
<td></td>
<td>4-6 raw θ,φ,Ψ bus rotation</td>
<td>m</td>
<td>Gyrosim</td>
</tr>
<tr>
<td>Outputs</td>
<td>1-3 sensed x,y,z bus translation</td>
<td>rad</td>
<td>external</td>
</tr>
<tr>
<td></td>
<td>4-6 sensed x,y,z bus rotation</td>
<td>rad</td>
<td>external</td>
</tr>
</tbody>
</table>

Table 3.14: The inputs and outputs of the SVS block.

3.6 The Integrated ProtoSim

Following the input and output relationships described by Tables 3.1 to 3.14, the dynamics, actuator, and sensor blocks were integrated to form ProtoSim. The dynamics block takes forces and torques as inputs. Thus, the actuator models were connected by treating them as “torque sources” which act either on the bus (the reaction wheels) or the rib tips (the CMGs). The dynamics block’s outputs are the translations, rotations, and rates in absolute coordinates. Some of these outputs are converted to relative coordinates before being fed to the sensor models.

ProtoSim is executed by running the script `protorun.ms`, listed in Appendix A.1. This script calls the functions `set_parameters.msf` and `simmattipcon.msf`, listed in Appendices A.2 and A.3, which respectively set up the parameters and NASTRAN structural matrices required by the simulator. Finally, the script loads and executes the top-level ProtoSim superblock, shown in Figure 3.17. The user inputs to ProtoSim and the resulting outputs are summarized in Table 3.15.
Figure 3.17: The top-level ProtoSim superblock.
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>1-3</td>
<td>reaction wheel command currents $i_x, i_y, i_z$</td>
</tr>
<tr>
<td></td>
<td>4,5</td>
<td>CMG #1 gimbal current $i_c$ and spin motor voltage $V_s$, respectively</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CMG #5 gimbal current $i_c$ and spin motor voltage $V_s$, respectively</td>
</tr>
<tr>
<td>Outputs</td>
<td>1-3</td>
<td>SVS hub $x,y,z$ translation</td>
</tr>
<tr>
<td></td>
<td>4-6</td>
<td>SVS hub $x,y,z$ rotation</td>
</tr>
<tr>
<td></td>
<td>7,8</td>
<td>rib #1 $y,z$ translation from rib tip sensor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SVS hub $x,y,z$ rotation rates from inertial rate sensor</td>
</tr>
<tr>
<td></td>
<td>15-16</td>
<td>rib #5 $y,z$ translation from rib tip sensor</td>
</tr>
<tr>
<td></td>
<td>17-19</td>
<td>hub $x,y,z$ rotation rates from inertial rate sensor</td>
</tr>
<tr>
<td></td>
<td>20-22</td>
<td>reaction wheel speeds from encoder</td>
</tr>
<tr>
<td></td>
<td>23-27</td>
<td>CMG wheel speeds from encoder</td>
</tr>
<tr>
<td></td>
<td>28-32</td>
<td>CMG gimbal angles, from potentiometer</td>
</tr>
<tr>
<td></td>
<td>33-158</td>
<td>diagnostic outputs</td>
</tr>
</tbody>
</table>

**Table 3.15**: ProtoSim's inputs and outputs.

### 3.7 Simulator Verification

#### 3.7.1 Actuator Tests

To verify that the reaction wheel and CMG blocks were set up correctly, the analytical expressions derived in Section 3.4 were used to calculate a test case. These same parameters were then fed to the ProtoSim modules and the results were compared. Given a step input, doing an inverse Laplace transform on (3.21) yields a time-domain expressions for the reaction wheel speed:
\[ \omega(t) = \frac{K_T i_s(t)}{B} \left(1 - \exp\left(-\frac{B}{J}t\right)\right) \]  

(3.47)

Differentiating (3.47) results in the reaction wheel acceleration:

\[ \dot{\omega}(t) = \frac{K_T i_s(t)}{J} \exp\left(-\frac{B}{J}t\right) \]  

(3.48)

Substituting (3.48) into (3.24) and assuming small hub rotations rates such that \( \omega = 0 \) produces an approximate scalar expression for the reaction wheel torque about one axis:

\[ \tau_{nw} = -I_s \left( \frac{K_T i_s(t)}{J} \exp\left(-\frac{B}{J}t\right) \right) \]  

(3.49)

For the CMGs, using the final value theorem [8] on (3.27) produces an expression for the steady-state wheel speed:

\[ \omega_s = \frac{K_m v_{ss}}{K_m^2 + R_a B} \]  

(3.50)

Doing an inverse Laplace transform on (3.26) with a step input results in the time-domain expression for the CMG gimbal angle:

\[ \theta(t) = \frac{K_T i_s(t)}{B} \left(t - \frac{J}{B} \exp\left(-\frac{B}{J}t\right)\right) \]  

(3.51)

Differentiating (3.51) once and twice results the CMG gimbal rate and acceleration, respectively:
\[
\begin{align*}
\dot{\theta}(t) &= \frac{K_T i_s(t)}{B} (1 - \exp\left(-\frac{B}{J} t\right)) \\
\ddot{\theta}(t) &= \frac{K_T i_s(t)}{J} \exp\left(-\frac{B}{J} t\right) 
\end{align*}
\] (3.52)

Assuming the wheel speed to be constant and the rib tip rotation rates to be small, the CMG torque expression (3.36) becomes:

\[
g_{cmg} = -I_s \omega_s (\dot{\theta} \times a) - I_e \ddot{\theta} 
\] (3.53)

where:

\[
\begin{align*}
a &= [1 \ 0 \ 0]^T \\
\dot{\theta} &= [0 \ 0 \ \dot{\theta}(t)]^T \\
\ddot{\theta} &= [0 \ 0 \ \ddot{\theta}(t)]^T
\end{align*}
\]

The test case chosen involved applying a constant step current of 1 mA to the reaction wheel and CMG models for 30.0 s. Using these inputs and the system parameters defined by the set_parameters.m function listed in Appendix A.2, the expected values were calculated using (3.47), (3.49), (3.51), and (3.53). As shown in Table 3.15, these calculations corresponded very favorably to the simulation results. The only discrepancy is the additional x-axis component of CMG torque. This value is from the \(-I_s \omega_s a\) term of (3.31), which was assumed small and neglected in (3.36).

<table>
<thead>
<tr>
<th>Actuator</th>
<th>Output</th>
<th>Expected Value</th>
<th>Simulation Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>reaction wheel</td>
<td>wheel speed (\omega_s)</td>
<td>(8.785 \times 10^{-2}) rad/s</td>
<td>(8.785 \times 10^{-2}) rad/s</td>
</tr>
<tr>
<td></td>
<td>torque (g_{cw})</td>
<td>(-2.691 \times 10^{-6}) Nm</td>
<td>(-2.691 \times 10^{-6}) Nm</td>
</tr>
<tr>
<td>CMG</td>
<td>wheel speed (\omega_s)</td>
<td>1047.133 rad/s</td>
<td>1047.131 rad/s</td>
</tr>
<tr>
<td></td>
<td>gimbal angle (\theta)</td>
<td>1.306 rad</td>
<td>1.306 rad</td>
</tr>
</tbody>
</table>
|                | torque \(g_{cmg}\)      | \[
\begin{bmatrix}
0 \\
-2.069 \times 10^{-2} \\
-4.323 \times 10^{-15}
\end{bmatrix}
\] Nm | \[
\begin{bmatrix}
-5.907 \times 10^{-17} \\
-2.061 \times 10^{-2} \\
-4.323 \times 10^{-15}
\end{bmatrix}
\] Nm |

Table 3.16: Actuator model test results.
A series of tests were conducted on ProtoSim to ensure that the simulator did not violate the laws of conservation of energy and momentum (a very serious offense!). These tests were designed to show that the simulator does not have any erroneous sources or sinks of energy or momentum. From classical mechanics, the work $w$ done on a body by an applied force $f$ or an applied torque $g$ is given by:

$$ w = \int_0^r f \, dr $$

$$ w = \int_0^\theta g \, d\theta $$

(3.54)

The corresponding kinetic energy expressions are:

$$ T = m \int_0^v \nu \, d\nu $$

$$ T = J \int_0^\omega \omega \, d\omega $$

(3.55)

An impulse test was employed to confirm that the simulator does not violate the law of conservation of momentum. From classical mechanics, impulse is defined as:

$$ \text{imp} = \int_0^r f \, dt $$

$$ \text{imp} = \int_0^\theta g \, dt $$

(3.56)
The corresponding linear and angular momentum expressions are:

\[ p = m \int_0^r dv \]
\[ p = I \int_0^\omega d\omega \]  \hspace{1cm} (3.57)

Force and torque step inputs of 5.0 s duration with magnitudes of $1.0 \times 10^{-3}$ N and $4.9 \times 10^{-3}$ Nm respectively were applied to each axis. The mass of the DICE hub was 22.679 kg and the mass of each rib tip package was 0.9072 kg, for a total mass of 27.215 kg. Using the parallel axis theorem, the axial and transverse moments of inertia were calculated to be 1.736 kg\(\cdot\)m\(^2\) and 1.594 kg\(\cdot\)m\(^2\) respectively.

To check the energy balance, the values of \(f\) and \(g\) were substituted into (3.54) and integrated over the simulated displacement and rotation respectively to compute the work applied to the system. Using the known mass and inertia properties, (3.55) was then used to calculate the resulting kinetic energy given the linear and angular velocity outputs of ProtoSim. The results are plotted in Appendix B.1 and are summarized in Table 3.17.

For the momentum balance test, the inputs \(f\) and \(g\) were integrated over the duration of the run, as in (3.56). Then, using the known mass and inertia values with the linear and angular velocity outputs of the simulation, the corresponding linear and angular momenta were calculated using (3.57). The results are plotted in Appendix B.2 and are summarized in Table 3.18. As with the energy balance test, there is a good agreement between the expected and the simulated values, conferring a high degree of confidence that ProtoSim does not violate energy and momentum conservation. The only major discrepancies were the oscillations in the kinetic energy plots of Appendix B.1. These are most likely due to the restoring effect caused by the motion of the flexible ribs.
<table>
<thead>
<tr>
<th>Input</th>
<th>Axis</th>
<th>Expected Steady-State Value</th>
<th>Simulation Steady-State Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>force</td>
<td>x</td>
<td>$4.5979 \times 10^{-7}$ J</td>
<td>$4.5869 \times 10^{-7}$ J</td>
</tr>
<tr>
<td></td>
<td>y</td>
<td>$4.5979 \times 10^{-7}$ J</td>
<td>$4.5869 \times 10^{-7}$ J</td>
</tr>
<tr>
<td></td>
<td>z</td>
<td>$4.5994 \times 10^{-7}$ J</td>
<td>$4.5859 \times 10^{-7}$ J</td>
</tr>
<tr>
<td>torque</td>
<td>x</td>
<td>$1.8869 \times 10^{-4}$ J</td>
<td>$1.8705 \times 10^{-4}$ J</td>
</tr>
<tr>
<td></td>
<td>y</td>
<td>$1.8868 \times 10^{-4}$ J</td>
<td>$1.8705 \times 10^{-4}$ J</td>
</tr>
<tr>
<td></td>
<td>z</td>
<td>$1.7344 \times 10^{-4}$ J</td>
<td>$1.7149 \times 10^{-4}$ J</td>
</tr>
</tbody>
</table>

Table 3.17: Energy balance test results.

<table>
<thead>
<tr>
<th>Input</th>
<th>Axis</th>
<th>Expected Steady-State Value</th>
<th>Simulation Steady-State Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>force</td>
<td>x</td>
<td>$5.000 \times 10^{-3}$ kg m/s</td>
<td>$4.997 \times 10^{-3}$ kg m/s</td>
</tr>
<tr>
<td></td>
<td>y</td>
<td>$5.000 \times 10^{-3}$ kg m/s</td>
<td>$4.997 \times 10^{-3}$ kg m/s</td>
</tr>
<tr>
<td></td>
<td>z</td>
<td>$5.000 \times 10^{-3}$ kg m/s</td>
<td>$4.996 \times 10^{-3}$ kg m/s</td>
</tr>
<tr>
<td>torque</td>
<td>x</td>
<td>$2.450 \times 10^{-3}$ kg m$^2$/s</td>
<td>$2.442 \times 10^{-3}$ kg m$^2$/s</td>
</tr>
<tr>
<td></td>
<td>y</td>
<td>$2.450 \times 10^{-3}$ kg m$^2$/s</td>
<td>$2.442 \times 10^{-3}$ kg m$^2$/s</td>
</tr>
<tr>
<td></td>
<td>z</td>
<td>$2.450 \times 10^{-3}$ kg m$^2$/s</td>
<td>$2.440 \times 10^{-3}$ kg m$^2$/s</td>
</tr>
</tbody>
</table>

Table 3.18: Linear and angular momentum balance test results.

3.7.3 Deterministic Simulations

The final suite of tests on ProtoSim involved a series of deterministic simulations in which the output of the simulator was compared with off-line predictions for a set of simple inputs. These inputs were:

- A current pulse applied to each of the reaction wheels.
- A current step applied to the gimbal motors of the CMGs on ribs #1 and #2.

For the reaction wheel tests, a current pulse of 0.163 A was applied to the x- and y-axis wheels and a pulse of 0.067 A was applied to the z-axis wheel. These pulses were of 5.0 s duration. The predicted values of wheel speed and torque were calculated using (3.21) and (3.25), respectively. For the rotation and rotation rate of the hub, the following kinematic expressions were employed:
\[
\dot{\omega} = \frac{g_{\text{rw}}}{I}
\]
\[
\omega = \omega_0 + \dot{\omega} t
\]
\[
\theta = \theta_0 + \omega_0 t + \frac{1}{2} \dot{\omega} t^2
\]  

(3.58)

where:
- \( g_{\text{rw}} \) = reaction wheel torque (Nm)
- \( I \) = moment of inertia of DICE (kg m\(^2\))
- \( t \) = time (s)

Predictions from (3.21), (3.25), and (3.58) are plotted in Appendix B.3. These were found to agree with the simulation results.

For the CMG tests, step currents of \( 7.3 \times 10^{-4} \) A and 5.0 s duration were applied to the units on ribs #1 and #2. Recall that CMG #1 gimbals about the z-axis while CMG #2 pivots about the y-axis, which is why two tests are required. Using (3.36) to calculate the CMG torque \( g_{\text{cmg}} \), the predicted maximum deflection \( \delta \) was roughly estimated by assuming the rib to be an aluminum cantilever beam of circular cross section and employing the static deflection equation:

\[
\delta = \frac{gL^2}{2EI_{\text{Al}}}
\]  

(3.59)

where:
- \( L \) = length of the rib (m)
- \( EI_{\text{Al}} = 0.344 \) Nm\(^2\)

It should be noted that (3.59) gives a measure of static deflection, and static and dynamic deflections are not directly comparable. The purpose here was only to get an order-of-magnitude type estimate of what the deflection should approximately be as a sort of "sanity check". Besides the magnitude of the deflection, the simulation results were also examined for other basic
measures of "reasonableness", such as whether the directions were correct and how the motions decayed over time. Finally, the deterministic outputs of ProtoSim were cross-checked with similar outputs from LinSim, a linear DICE simulator independently written by R.E. Zee as a test-bed for controller development [19]. The results from both simulators were found to agree.

<table>
<thead>
<tr>
<th>CMG</th>
<th>Predicted Maximum Deflection</th>
<th>Simulation Maximum Deflection</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$1.616 \times 10^{-5}$ m</td>
<td>$1.251 \times 10^{-5}$ m</td>
</tr>
<tr>
<td>2</td>
<td>$1.616 \times 10^{-5}$ m</td>
<td>$1.250 \times 10^{-5}$ m</td>
</tr>
</tbody>
</table>

Table 3.19: CMG gimbal current step test results.

3.8 ProtoSim User's Guide

<table>
<thead>
<tr>
<th>Filename</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProtoRun.ms</td>
<td>Main ProtoSim execution script.</td>
</tr>
<tr>
<td>ProtoSim_REZ_970330.sysbld</td>
<td>SystemBuild block diagram.</td>
</tr>
<tr>
<td>set_parameters.msf</td>
<td>Parameter setup function.</td>
</tr>
<tr>
<td>random_input.msf</td>
<td>Random input generation function.</td>
</tr>
<tr>
<td>simmattipcon.msf</td>
<td>Setup NASTRAN structural parameters.</td>
</tr>
<tr>
<td>mks_eigdata_nosprings.xmd</td>
<td>Modal parameter data from NASTRAN.</td>
</tr>
<tr>
<td>add_lists.msf</td>
<td>Functions required by Gyrosim, written by S.R. Piggott</td>
</tr>
<tr>
<td>block_diag.msf</td>
<td></td>
</tr>
<tr>
<td>cross.msf</td>
<td></td>
</tr>
<tr>
<td>extract_block_diag.msf</td>
<td></td>
</tr>
<tr>
<td>selected_evec.msf</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.20: ProtoSim files required for execution.

ProtoSim runs under the Xmath platform. All the files it requires are shown in Table 3.20, and the full listings of those written by the author can be found in Appendix A. To ensure that the simulator can find the files during execution, they must all be placed in the same directory. Before a run, make sure that Xmath is referencing the directory in which ProtoSim
resides. If it is not, click on the File option in the Xmath menu bar, select the Set Directory option, and then enter the correct path.

ProtoRun.ms is the main execution script. By following the instructions in the comments denoted by the # character, this file can be modified with any standard text editor to customize the simulation. It is a good idea to make a backup copy of ProtoRun.ms before modifying the script. Once the initial parameters have been set to the desired values, starting the simulation is very simple. In the Xmath command window, enter the following line:

```
execute file = "ProtoRun"
```

While running, the percentage of completion is shown in the Xmath display window in approximately 20% increments, along with the current values of the outputs at those intervals. When the simulation is finished, the data is saved to the files specified in the ProtoRun.ms script. The results are then plotted on the screen. Printouts of these graphs can be produced by clicking the File option on the plotting window menu bar and selecting Print.
4.1 The Sensitivity of OKID/ERA to Parameter Variations

To help define the top-level system requirements for DICE, it would be helpful to quantify the effect of generic parameter variations on the quality of the SI model. The sensitivity of the OKID/ERA algorithm to these variations will dictate the quality of the data required to produce an adequate model. This in turn imposes performance requirements on the DICE components and the definition and duration of the experiments.

One and two mass-spring-dashpot mechanical analogies were used to study this issue. Since the trends observed in both the one and two mass cases were identical, only the results of the latter system will be presented here. Figure 4.1 shows the two-mass system used in this study. The mass and spring stiffness were chosen to be $m_1 = m_2 = 1.0 \text{ kg}$ and $k_1 = k_2 = 1.0 \text{ N/m}$, respectively. It was desired to have an underdamped system, so a damping ratio $\zeta = 0.2$ was selected. The dashpot constant $c$ is related to the mass $m$, stiffness $k$, and damping $\zeta$ by [8]:

$$c = \sqrt{\zeta m k} \quad (4.1)$$

Using (4.1), the dashpot parameters were computed as $c_1 = c_2 = 0.4 \text{ kg/s}$. With the system defined, the perturbations to be studied had to be decided upon. Based upon the recommendations of P.K. Nguyen [18], the following factors were considered:

- Data sampling frequency: 2.0, 0.5, 0.2, and 0.1 times the reference sampling frequency of 5.0 Hz.
- Input-output skew time: 1, 2, 3, and 4 sampling intervals.
- Noise in sensor output: Maximum noise amplitudes of 5.0%, 1.0%, 0.1%, and 0.01% of the maximum amplitude of mass $m_2$ translation.
• Noise in actuator input: Maximum noise amplitudes of 5.0%, 1.0%, 0.1%, and 0.01% of the maximum amplitude of the applied force.

• Noise in sensor output and actuator input: Maximum output noise amplitudes of 5.0%, 1.0%, 0.1%, and 0.01% of the maximum amplitude of mass \(m_2\) translation, and maximum input noise amplitudes of 5.0%, 1.0%, 0.1%, and 0.01% of the maximum amplitude of the applied force.

• Sensor resolution: \(\pm 1\) bit and \(\pm 2\) bits from the reference resolution. The resolution in reference case is assumed to be \(12\) bits = \(2\times\) (maximum amplitude of mass \(m_2\) translation).

• Actuator resolution: \(\pm 1\) bit and \(\pm 2\) bits from the reference resolution. The resolution in reference case is assumed to be \(12\) bits = \(2\times\) (maximum amplitude of the applied force).

• Duration of experiment: 90%, 50%, 20%, and 10% of the reference simulation run time of 30 s.

For these simulations, the input excitation force \(f\) was applied to the second mass \(m_2\). Originally, short duration impulses and single-frequency sinusoids were employed. These signals produced poor SI models. The reason is that if the input signal is not frequency-rich, the matrix \(V\) in (2.6) becomes ill-conditioned [3]. If this is the case, then the system Markov parameters required by ERA cannot be computed accurately. To overcome this problem, a random input signal was used instead. Using the \texttt{random_input.msf} function listed in Appendix A.3, the input signal produced for these runs had a normal distribution and zero mean, with a maximum amplitude and frequency of 1 N and 10 Hz, respectively.

Having established the system, the perturbations to be studied, and the characteristics of the input signal, a quantitative measure of the SI quality must be defined. For this study, the root-mean-squared (RMS) ratio was used. This quantity is the 2-norm of the error divided by the 2-norm of the unperturbed simulation output:
For these analyses, the SI models were generated using the forcing function $f$ acting on mass $m_2$ as the input and the displacement of both masses $v_1$ and $v_2$ as the outputs to the OKID/ERA algorithm. The number of rows in the Hankel matrix of (2.10) was set to $p = 2$. Recall from Section 2.2 that this will result in a system order of 4, since $p$ must be chosen such that it is greater than or equal to the system order $n$ over the number of outputs $m$.

A "reference" simulation output was first generated with noise-free data with no skew, using the baseline sampling rate, resolution, and experiment duration. Each factor listed above was then applied to the data one at a time. New SI models were then developed from each of these "perturbed" data sets, which were then simulated using the original input data. Appendix C.1 displays the results of these SI runs, which compare the outputs $y_2$ from both the original simulation and SI model generated from the "perturbed" data, along with the associated RMS ratios computed from (4.2).

Based on these results, it can be concluded that the quality of SI models generated by OKID/ERA are most sensitive to perturbations in sampling frequency and resolution. This makes sense intuitively, because the SI algorithm will blindly attempt to identify the system based on whatever data it is given. If the perturbed data is very different from the original set, the algorithm is effectively trying to identify a "different" system than the original.

Changing the skew time produced large RMS ratios, but this is because the SI output is being compared to the original simulation output. As shown in Appendix C.1, OKID/ERA was successful in capturing the modes of the system. Note that the original and SI outputs are virtually identical in shape but differ only by an offset (the skew time). Since such constant offsets are easily removed in data post-processing, the presence of skew cannot be considered a significant problem for the SI process.

These results also show that OKID/ERA is fairly insensitive to random noise of small amplitude. This is because OKID uses a Kalman filter to choose an optimal value of the gain matrix $G_i$ in (2.3). These results show that this state estimation step makes OKID/ERA somewhat robust.
to the effects of systemic noise. It was also observed that the errors caused by input/output noise were approximately a superposition of the effects of the actuator and sensor noise individually, which is to be expected for a linear system like the one used here. Interestingly, the results of Appendix C.1 show that perturbations in the input data have much less effect on SI quality than the degradation of the simulation output. This may be due to the presence of damping in the system, which has the effect of "smoothing out" the motions caused by the non-ideal input.

One factor that could not be clearly quantified with these models was the effect of simulation time. For the spring-mass-damper model of Figure 4.1, it was found that changing the run time had little effect on the quality of the SI model. The exact reason for this result remains unclear. Since this model is described by a second-order differential equation, only two data points are required to uniquely define the system. Therefore, an obvious degradation in SI quality may only appear when the simulation time is significantly shorter than the ones used here. This phenomenon is unique to this system, as the results of Section 4.2 will show.

4.2 System Identification of ProtoSim

On-orbit SI is a primary objective of the DICE mission. It is therefore essential to verify that the current parameters of the free-flyer are conducive to producing good results. To this end, a series of SI experiments were conducted with ProtoSim to:

- Characterize the inputs and duration of the experiments.
- Find out if the current DICE design is adequate for SI experiments.
- Discover any remaining bugs in ProtoSim that may have slipped through the verification process.

These SI runs were done in a systematic manner, starting with the dynamics block and building up to the full simulator. The experiments were executed in the following sequence:

- Dynamics block only (\texttt{REZGyroSim}), with force/torque inputs and outputs in absolute coordinates.
- Dynamics block with outputs in relative coordinates (via the Transformation block).
- Dynamics block, outputs in relative coordinates, and sensors.
4.2.1 Identification of the Dynamics Block

Following the above procedure, SI was first tested on the structural dynamics block alone. Recall from Table 3.1 that this block has a total of 21 inputs, consisting of three air thruster forces acting on the hub, three torques acting about the hub, and three torque components acting on each rib. These inputs were generated using the `random_input.msf` function of Appendix A.3. The maximum applied force and torque amplitudes were $1 \times 10^{-3}$ N and $1 \times 10^{-3}$ Nm respectively, and the highest frequency of excitation was 50 Hz (selected because the highest frequency modeled by the dynamics block is 56 Hz [13]). The outputs of the block totalled 36, consisting of three hub translations, three hub rotations, and six translational and rotational components for each rib. Run time was one minute, and the parameter $p$ in the Hankel matrix of (2.10) was chosen to be $p = 12$.

The results of this initial SI test were surprisingly poor. The problem was eventually traced back to a problem with the first six values of $\Omega^2$ from (3.8), the squared modal frequencies of the rigid modes. These values, which were imported into SystemBuild from the original structural formulation in NASTRAN, were slightly negative due to numerical errors. Though these errors were very small (on the order of $10^{-10}$), they were significant. When the matrix setup function `simmattipcon.msf` of Appendix A.4 took the square root of $\Omega^2$, complex natural frequencies were the result. Since OKID can only accurately identify linear real systems, the presence of such complex “rigid-elastic” modes produced very poor SI results.

A modification to `simmattipcon.msf` that forced the first six system eigenvalues to be zero was the solution. This allowed OKID/ERA to produce excellent results, as witnessed by the plots in Figure 4.2 and Appendix C.2. The average RMS ratio over all the outputs was $2.62 \times 10^{-4}$. 
Figure 4.2: SI experiment results for rib #1 (dynamics block only).
4.2.2 Identification of the Dynamics Block in Relative Coordinates

The next step was to add the Transformation block so that the rib deflection outputs would be in relative coordinates. For this test, the input characteristics, simulation time, and value of $p$ were initially the same as before. The initial SI attempts ended in failure, which demanded several changes in methodology.

First, the value of $p$ was reduced from 12 to 4. Contrary to the theory presented in Section 2.2, it was found that for the case of noise-free simulations, increasing $p$ actually worsened the SI results due to the numerical errors associated with handling a much larger Hankel matrix. Later SI runs with ProtoSim would confirm this observation. Second, the number of inputs and outputs were reduced. The inputs were cut from 21 to 18 by removing the hub air thruster forces. Since these thrusters will not be used as a source of excitation for SI experiments on the real DICE anyway, realism was not compromised. Also, the outputs were reduced from 36 to 16 by dropping the rib rotational components. Since the current rib tip sensing system cannot detect torsion, this information will in reality be unavailable for SI experiments. In addition, the length of the simulation was increased from one to three minutes.

The most significant change was to correctly size and characterize the inputs. A balance must be struck such that the inputs are of sufficient magnitude to excite as many modes of the structure as possible while keeping the motion within a linear regime. This is particularly important for the Transformation block, which as (3.16) shows employs several rotation matrices to execute the frame change. These rotation matrices are subject to the small angle restriction. Note that this should not be considered a serious limitation of ProtoSim itself, since as mentioned earlier SI algorithms like OKID/ERA can only identify linear systems anyway.

Sizing the inputs proved to be a straightforward but laborious iterative process. It was discovered that the characteristics of the current DICE structure were such that the coupling between the ribs and the hub resulted in either reasonable hub rotations (less than 10°) but excessively small rib deflections (less than 1 mm), or reasonable rib deflections (approximately 70 mm) but very large hub rotations (greater than 20°). This observation means that for the real DICE mission some form of closed-loop SI may have to be employed.

For the SI experiments with ProtoSim, the solution was to constrain the motions of the hub
using a proportional-derivative (PD) controller. The proportional gain $K_p$ and the derivative gain $K_d$ were selected such that:

$$K_d = \frac{2J\zeta \omega_c}{l_s K_i}$$

Equation (4.3)

$K_p$ and $K_d$ were calculated using an axial inertia $I_a = 0.3512$ kg·m$^2$, a transverse inertia $I_t = 0.9022$ kg·m$^2$, a controller frequency $f_c = 0.4$ Hz, and a damping ratio $\zeta = 0.1$. The input torques applied to the system were $1 \times 10^{-3}$ Nm about each axis of the hub, 0.612 Nm about the x- and z-axes of each rib, and 0.644 Nm about the y-axis of each rib. This input profile produced hub rotations below $5^\circ$ and rib deflections of 40-70 mm, which were found to be adequate for generating a good SI model. The results are shown in Figure 4.3 and Appendix C.2.2.

This long run also afforded an opportunity to quantify the effect of varying experiment time on the quality of the SI model, since the spring-mass-dashpot analogies of Section 4.1 did not yield conclusive results. One and two minute subsets of the input and output data were used to generate SI models which were then simulated and compared to the original three minute simulation output. As summarized in Table 4.1, these runs indicate that the error decreases by about 30% per doubling of experiment time.

<table>
<thead>
<tr>
<th>Experiment Time (s)</th>
<th>Average RMS Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>60.0</td>
<td>0.049275</td>
</tr>
<tr>
<td>120.0</td>
<td>0.030288</td>
</tr>
<tr>
<td>180.0</td>
<td>0.027676</td>
</tr>
</tbody>
</table>

Table 4.1: The effect of experiment time on SI model quality.
Figure 4.3: SI experiment results for rib #1 (dynamics block in relative coordinates).
4.2.3 Sensor Sizing

Three sensors are available in the current DICE design to provide data for SI. Hub translations are sensed only by SVS, while the rib motions are measured by tip sensors. The rotations of the hub can be obtained either directly from SVS or indirectly by integrating the signal from the inertial rate sensor. It is necessary to verify that these sensors can provide data to produce good SI results.

The first SI attempt used hub rotation data provided by SVS. For these runs, this sensor was assumed to have a resolution of 8 bits over a maximum anticipated translation of 0.3 m and a maximum anticipated rotation of 45°. Operating with a frame rate of 30 Hz, its positional accuracy is ±1.0 mm and its angular accuracy is ±0.05°.

Using $p = 4$ and a one minute experiment time, OKID/ERA was able to realize a good system with an average RMS ratio of this run was 0.34459. The only deviation from the nominal SI procedure was that the unstable modes of the realized model had to be manually removed after the run. These unstable modes are a mathematical artifact of the SI algorithm, and occur when $p$ is set to a high value (over-parameterizing) while attempting to realize a system with noisy or corrupt data. This run indicates that such post-processing will be a necessary part of the SI experiments on the real DICE mission.

Increasing the size of the Hankel matrix to $p = 6$ produced better results, with an average RMS ratio of 0.18496. It was observed that SI results were consistently poorer for hub rotations about the z-axis and for rib translations along the plane of the rib than the other motions of DICE. These modes experience the strongest interaction with the flexible (and potentially non-linear) ribs.

It has been proposed that the hub rotations of DICE can be sensed by integrating the signal from the inertial rate sensor. This unit has a manufacturer’s specified resolution of $6.98 \times 10^{-5}$ rad/s [17], and is baselined to operate at a sampling frequency of 100 Hz. For the SI run using this data instead of SVS information, the average RMS ratio was 0.18392. This run indicates that given the current design parameters, there is virtually no difference between using SVS or the rate sensor to identify the hub rotations. Figure 4.4 shows the SI results for rib #1. The rest of the plots for this run can be found in Appendix C.2.3.
Figure 4.4: SI experiment results for rib #1 (using sensor data).
4.2.4 Identification of ProtoSim

For the final SI experiment, the full ProtoSim -- including the actuator models -- was employed. The DICE actuator suite consists of three orthogonal reaction wheels mounted in the hub, an air-thruster system housed also on the hub, and five control moment gyros (CMG) at the tip of each rib. Of these, only the reaction wheels and CMGs will be used as excitation sources for SI experiments. The air-thruster system is intended only for station-keeping and momentum management.

Adding the reaction wheels to the simulator was relatively straightforward, requiring only a modification to the PD gains to convert the output of the controller from a torque to a commanded current for the reaction wheels:

\[
K_p = \frac{J/\omega_c^2}{I_s K_t}, \quad \text{where} \quad \omega_c = 2\pi f_c
\]

\[
K_d = \frac{2J\zeta \omega_c I}{I_s K_t}
\]  \hspace{1cm} (4.4)

Finally, the simulator was completed by adding the CMG models and an SI run was conducted. The controller gains of (4.4) were calculated using \( f_c = 0.09 \) Hz, \( \zeta = 0.005 \), with inertias of 0.3512 kg\cdotm\(^2\) and 0.9022 kg\cdotm\(^2\) about the axial and transverse axes of DICE, respectively. Both of the current inputs to the reaction wheels and the CMGs were normally distributed random signals of maximum amplitude 0.02 A and 0.03 A, respectively. The latter value was constrained by the necessity of keeping the CMG gimbal angle between ±20°. For this experiment, the simulation time was 60 s.

As shown in Figure 4.5 and Appendix C.2.4 the results for this run were poor, with an average RMS ratio of 0.58252. Three problems were identified. First, the ribs were not being excited enough. In Section 4.2.2, it was found that rib deflections on the order of 40-70 mm produced excellent results. Unfortunately, as the plots in Appendix C.2.4 show, the ribs motions for this run were only 6-10 mm. The second problem was that the hub translations were very small, from \( 3 \times 10^{-4} \) m to \( 6 \times 10^{-4} \) m. This was below the detection threshold of SVS, making them useless.
for SI. Without this data, hub rotations could not be realized for this run. Finally, to get these results the size of the Hankel matrix of (2.6) had to be increased dramatically, to \( p = 14 \). If such a high observer value must be used during the actual DICE mission, this will obviously place greater demands on the experiment in requiring both increased computational power and data processing time.
Figure 4.5: SI experiment results for rib #1 (with complete ProtoSim).
The purpose of this thesis was to contribute to the top-level requirements definition of DICE. Towards this end, the effect of generic parameter variations to the quality of SI models produced by the OKID/ERA SI algorithm was quantified. Through a series of tests with spring-mass-dashpot analogies and the ProtoSim simulator, it was found that:

- The OKID/ERA algorithm is most sensitive to perturbations in sampling frequency and resolution. Skew time effects may be eliminated in post-processing, so it cannot be considered a serious problem. Thanks to the Kalman filter in OKID, the algorithm is fairly robust to small amplitude random noise.
- For successful SI, a structure must be sufficiently excited to exercise as many of the modes as possible, but not overly excited such that the motion goes outside the linear regime. This is because OKID/ERA is linear and can only produce linear models.

The most important issue addressed by this thesis was whether the current DICE design is sufficient for supporting SI experimentation. Runs with ProtoSim showed that the current design unfortunately produces relatively poor results. The primary problem was that there was a very large coupling between the hub and rib motions. Even very small excitations of the ribs produced excessively large rotations of the hub. From the perspective of SI, the hub was being excited too much (outside the linear regime) while the ribs were moving too little (not enough excitation of the modes). Through the SI runs on ProtoSim, the following conclusions may be drawn:

- If the current DICE design is used, wholly open-loop SI may not be possible. Some sort of controller, like the PD controller set up in ProtoSim, should be used to constrain the hub motions. As for SI methodology, this dictates that the primary source of excitation of the structure should come from the rib-tip CMGs. The reaction wheels would provide only secondary excitation, with their primary purpose being to constrain the hub rotations.
• The CMGs are not providing enough authority to excite the ribs sufficiently. ProtoSim suggests that a greater level of rib excitation, on the order of 40-70 mm, will produce better SI results. Either the CMGs should be re-sized, or a way must be found to condition the input such that the gimbal angle $\theta$ varies at a higher frequency (since the torque is proportional to $\theta$).

• For the magnitudes of motion required for good SI results, the DICE sensor suite is correctly sized. The rib-tip sensing system is particularly well suited. SVS was also found adequate for recording the hub motions. Given the present design parameters, there is virtually no difference between using SVS or integrating the inertial rate sensor signal to find hub rotation.

• The duration of the SI experiments conducted on ProtoSim ranged from one to three minutes. It was found that the quality of the SI model improved by approximately 30% per doubling of run time. The MACE mission employed six minute runs [20], and a similar duration has been proposed for DICE. Given the results from ProtoSim, this baseline appears correct.

Canada was the third nation, after the former Soviet Union and the United States, to have a satellite in Earth orbit. Since the launch of Alouette 1, Canadian aerospace engineers have been recognized as world leaders second to none in the field of FSS. DICE will be the flagship that will carry this proud tradition into the next century and beyond. It is the author's greatest hope that this thesis will be of assistance (or at least not be a hindrance!) to this great endeavor. Through DICE, Canadian spacecraft dynamicists will remain at the leading edge of this crucial field of aerospace technology.


73


### ProtoSim Execution Script: protorun.ms

```plaintext
# Program: ProtoSim Execution Script: protorun.ms
Author: Eric M. Choi
Department: UTIAS
Date: January 22, 1997

Purpose: "ProtoSim (and variations) Execution Script"
This script executes ProtoSim, a prototype DICE simulator that
uses Stephen Piggott's Gyrosim model as its dynamical plant.
ProtoSim includes models of the sensors and actuators written
by Eric Choi.

Modification History:

<table>
<thead>
<tr>
<th>Date</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>97/02/21</td>
<td>R. E. Zee</td>
<td>Modified to incorporate feedback control.</td>
</tr>
<tr>
<td>97/02/10</td>
<td>R. E. Zee</td>
<td>Modified to allow for changes to rigid eigenvalues.</td>
</tr>
<tr>
<td>97/01/28</td>
<td>R. E. Zee</td>
<td>Modified to run different variations of ProtoSim including ProtoSim_REZ_970127 (new implementation of structural model).</td>
</tr>
</tbody>
</table>

```
# 6: Clamp spin voltage for CMG 1 to zero.
# 7: RW dists --- x: +0.025 Nm, -0.0375 Nm
#     y: +0.025 Nm, -0.0375 Nm
#     z: +0.01 Nm, -0.015 Nm
# 8: CMG dists --- 1: +0.25 Nm, -0.375 Nm
#     2: +0.1 Nm, -0.15 Nm
#     3: -0.25 Nm, +0.375 Nm
#     4: +0.25 Nm, -0.375 Nm
#     5: +0.1 Nm, -0.15 Nm

max_step = 0.1;           # Maximum integration step size.
dt = 0.01;               # Sampling interval (s) (default dt=0.001).
duration = 60;           # Simulation duration (s).
t = [0:dt:duration]';    # Time vector (s).

display "Selecting simulator ..."
if sysbld_block == "protosim_970217"
    build_block = "ProtoSim_970217";
    save_file = "ProtoSim_SAVE";
    y_file = "ProtoSim_Y";
    u_file = "ProtoSim_U";

elseif sysbld_block == "protosim_REZ_970330"
    if inputtype == 7 | inputtype == 8
        build_block = "ProtoSim_REZ_Control"
    else
        build_block = "ProtoSim_REZ_970330"
    endif

    save_file = "ProtoSim_REZ_SAVE";
    y_file = "ProtoSim_REZ_Y";
    u_file = "ProtoSim_REZ_U";
elseif sysbld_block == "protosim_rel_970217"

    build block = "ProtoSim_Rel_970217"
save file = "ProtoSim_Rel_SAVE"
y_file = "ProtoSim_Rel_Y";
u_file = "ProtoSim_Rel_U";
endif

# Set up matrices for structural model
#-------------------------------

display "Setting up structural model ...
smmattipcon(zeta,gyricity,rigidtype,sysbld_block);

# Set up Input Parameters
#-------------------------------

display "Defining inputs ...
if inputtype == 0
    # If random inputs requested.
    fh = 10;                          # Highest frequency of excitation (Hz)
                                       # (default fh=10).
    inputs = 13;                     # Number of input channels.
    cmg.amp = 0.06;                   # Actuator command amplification factors
    wheel.amp = 0.02;
    u_raw = random_input(dt,duration,inputs,fh);
    for i=1:3;
        u(:,i) = wheel.amp*u_raw(:,i);  # Reaction wheel currents (A).
    endFor;
    for i=4:2:12;
        u(:,i) = cmg.amp*u_raw(:,i);    # CMG gimbal command currents (A).
    endFor;
    for i=5:2:13;
        u(:,i) = smotor.Vcs*ones(t);    # CMG spin motor voltage (V).
    endFor;
elseif inputtype <> 0
    npts = length(t);
    pulseduration = 5;
    len = round(pulseduration/dt + 1,(up));
    if inputtype == 1
        # RW x-pulse.
        channels = 13;
        u = zeros(npts,channels);
        torque = 1.2e-2;
        u(1:len,1) = torque/wheel.Kt(1) * ones(len,1);
        for i=5:2:13;
            u(:,i) = smotor.Vcs*ones(t);  # CMG spin motor voltage (V).
        endFor;
elseif inputtype == 2  # RW y-pulse.
    channels = 13;
    u = zeros(npts, channels);
    torque = 1.2e-2;
    u(j:len, 2) = torque/wheel.Kt(2)*ones(len, 1);
    for i=5:2:13;
        u(:, i) = smotor.Vcs*ones(t);  # CMG spin motor voltage (V).
    endFor;

elseif inputtype == 3  # RW z-pulse.
    channels = 13;
    u = zeros(npts, channels);
    torque = 4.9e-3;
    u(1:len, 3) = torque/wheel.Kt(3)*ones(len, 1);
    for i=5:2:13;
        u(:, i) = smotor.Vcs*ones(t);  # CMG spin motor voltage (V).
    endFor;

elseif inputtype == 4
    channels = 13;
    u = zeros(npts, channels);
    torque = 0.1517;  # "Constant" torque required, CMG 1, # for rib deflection approx. 5 cm.
    u(1:npts, 4) = torque*gmotor.Btotal/(gmotor.Kt*cmg.ns*cmg.I*cmg.ws)*...
                   ones(npts, 1);
    for i=5:2:13;
        u(:, i) = smotor.Vcs*ones(t);  # CMG spin motor voltage (V).
    endFor;

elseif inputtype == 5
    channels = 13;
    u = zeros(npts, channels);
    torque = 0.1517;  # "Constant" torque required, CMG 2, # for rib deflection approx. 5 cm.
    u(1:npts, 6) = torque*gmotor.Btotal/(gmotor.Kt*cmg.ns*cmg.I*cmg.ws)*...
                   ones(npts, 1);
    for i=5:2:13;
        u(:, i) = smotor.Vcs*ones(t);  # CMG spin motor voltage (V).
    endFor;

elseif inputtype == 6
    channels = 13;
    u = zeros(npts, channels);
    for i=7:2:13;
        u(:, i) = smotor.Vcs*ones(t);  # CMG spin motor voltage (V).
    endFor;

elseif inputtype == 7  # RW disturbances.
    channels = 16;
\text{u} = \text{zeros(npts, channels)};

\text{for } i = 7:2:15;
    \text{u(:,i)} = \text{smotor.Vcs*ones(t)}; \quad \# \text{CMG spin motor voltage (V)}.
\text{endFor}

\text{torque} = 0.025;
\text{torquey} = 0.025;
\text{torquez} = 0.01;

\text{halfpt} = \text{round}(\text{controltime}/(2*\text{dt})+1,\{\text{up}\});
\text{fullpt} = \text{round}(\text{controltime}/\text{dt}+1,\{\text{up}\});

\text{u}(1:\text{halfpt},4) = \text{torque}/\text{wheel.Kt}(1)*\text{ones(halfpt,1)};
\text{u}(\text{halfpt}+1:\text{fullpt},4) = -1.5*\text{torque}/\text{wheel.Kt}(1)*\text{ones(fullpt-halfpt,1)};
\text{u}(1:\text{halfpt},5) = \text{torquey}/\text{wheel.Kt}(2)*\text{ones(halfpt,1)};
\text{u}(\text{halfpt}+1:\text{fullpt},5) = -1.5*\text{torquey}/\text{wheel.Kt}(2)*\text{ones(fullpt-halfpt,1)};
\text{u}(1:\text{halfpt},6) = \text{torquez}/\text{wheel.Kt}(3)*\text{ones(halfpt,1)};
\text{u}(\text{halfpt}+1:\text{fullpt},6) = -1.5*\text{torquez}/\text{wheel.Kt}(3)*\text{ones(fullpt-halfpt,1)};

\text{elseif inputtype} \quad = \quad 8 \quad \# \text{CMG disturbances.}

\text{channels} = 16;
\text{u} = \text{zeros(npts, channels)};

\text{for } i = 7:2:15;
    \text{u(:,i)} = \text{smotor.Vcs*ones(t)}; \quad \# \text{CMG spin motor voltage (V)}.
\text{endFor}

\text{torquel} = 0.25;
\text{torque2} = 0.1;
\text{torque3} = -0.25;
\text{torque4} = 0.25;
\text{torque5} = 0.1;

\text{halfpt} = \text{round}(\text{controltime}/(2*\text{dt})+1,\{\text{up}\});
\text{fullpt} = \text{round}(\text{controltime}/\text{dt}+1,\{\text{up}\});

\text{u}(1:\text{halfpt},8) = \text{torquel*gmotor.Btotal}/(\text{gmotor.Kt*cmg.ns*cmg.I*cmg.ws}*\text{ones(halfpt,1)});
\text{u}(\text{halfpt}+1:\text{fullpt},8) = -1.5*\text{torquel*gmotor.Btotal}/(\text{gmotor.Kt*cmg.ns*cmg.I*cmg.ofones(fullpt-halfpt,1)});
\text{u}(1:\text{halfpt},10) = \text{torque2*gmotor.Btotal}/(\text{gmotor.Kt*cmg.ns*cmg.I*cmg.ws}*\text{ones(halfpt,1)});
\text{u}(\text{halfpt}+1:\text{fullpt},10) = -1.5*\text{torque2*gmotor.Btotal}/(\text{gmotor.Kt*cmg.ns*cmg.I*cmg.ws}*\text{ones(fullpt-halfpt,1)});
\text{u}(1:\text{halfpt},12) = \text{torque3*gmotor.Btotal}/(\text{gmotor.Kt*cmg.ns*cmg.I*cmg.ws}*\text{ones(halfpt,1)});
\text{u}(\text{halfpt}+1:\text{fullpt},12) = -1.5*\text{torque3*gmotor.Btotal}/(\text{gmotor.Kt*cmg.ns*cmg.I*cmg.ws}*\text{ones(fullpt-halfpt,1)});
\text{u}(1:\text{halfpt},14) = \text{torque4*gmotor.Btotal}/(\text{gmotor.Kt*cmg.ns*cmg.I*cmg.ws}*\text{ones(halfpt,1)});
\text{u}(\text{halfpt}+1:\text{fullpt},14) = -1.5*\text{torque4*gmotor.Btotal}/(\text{gmotor.Kt*cmg.ns*cmg.I*cmg.ws}*\text{ones(fullpt-halfpt,1)});
\text{u}(1:\text{halfpt},16) = \text{torque5*gmotor.Btotal}/(\text{gmotor.Kt*cmg.ns*cmg.I*cmg.ws}*\text{ones(halfpt,1)});
\text{u}(\text{halfpt}+1:\text{fullpt},16) = -1.5*\text{torque5*gmotor.Btotal}/(\text{gmotor.Kt*cmg.ns*cmg.I*cmg.ws}*\text{ones(fullpt-halfpt,1)});

\text{endif}
endif

# Controller

if zerocontrol == 1:
    ak = zeros(80,80);
    bk = zeros(80,32);
    ck = zeros(16,80);
    dk = zeros(16,32);

    controller = [ak, bk; ck, dk];
else

    display "Loading controller ..."
    load "controller.xmd"
    [ak,bk,ck,dk] = abcd(k);
    controller = [ak, bk; ck, dk];
    sensortransf = [eye(6,6), zeros(6,6);
                    zeros(10,6), inv(diagonal(lepd.K)), zeros(10,16);
                    zeros(3,16), inv(diagonal(irs.K)), zeros(3,13);
                    zeros(3,19), eye(3,3), zeros(3,10);
                    zeros(10,22), eye(10,10)];

    bk = bk * sensortransf;
    dk = dk * sensortransf;
    potparams = [pot.i(1), pot.Ro, pot.K(1)];
endif

# Begin simulation

display "Simulating ..."
str = "load "'+sysbld_block+'".sysbld";"
e:execute str

y = sim(build_block, t, u, ...
    [simclock=1, ...
      simmonitor=1, ...
      simmessage=1, ...
      simtimer=1, ...
      dtmax = max_step]);

# Save results

display "Saving results ..."
str = "save "'+save_file+'".xmd" main.*;"
e:execute str
str = "save "'+y_file+'".xmd" y;"
e:execute str
str = "save "'+u_file+'".xmd" u;"
e:execute str

#
# Graph results

```
display "Displaying graphs ...
plot(y(1,1),{rows=3,columns=2,graph_number=1,...
    title="Hub Translation (m)",xlabel="",ylabel="X")
plot(y(2,1),{graph_number=3,xlabel="",ylabel="Y")
plot(y(5,1),{graph_number=5,xlabel="Time (s)",ylabel="Z")
plot(y(4,1),{graph_number=2,title="Hub Rotation (rad)",xlabel="",ylabel=""}
plot(y(5,1),{graph_number=4,xlabel="",ylabel=""}
plot(y(6,1),{graph_number=6,xlabel="Time (s)",ylabel=""}

pause "Show rib deflections"
plot(y(7,1),{rows=5,columns=2,graph_number=1,...
    xlabel="",ylabel="Rib 1")
plot(y(8,1),{graph_number=2,title="Z Deflection (m)",xlabel="",ylabel=""}
plot(y(9,1),{graph_number=3,xlabel="",ylabel="Rib 2")
plot(y(10,1),{graph_number=4,xlabel="",ylabel=""}
plot(y(11,1),{graph_number=5,xlabel="",ylabel="Rib 3")
plot(y(12,1),{graph_number=6,xlabel="",ylabel=""}
plot(y(13,1),{graph_number=7,xlabel="",ylabel="Rib 4")
plot(y(14,1),{graph_number=8,xlabel="",ylabel=""}
plot(y(15,1),{graph_number=9,xlabel="Time (s)",ylabel="Rib 5")
plot(y(16,1),{graph_number=10,xlabel="Time (s)",ylabel=""}

pause "Show hub rates"
plot(y(17,1),{rows=2,columns=2,graph_number=1,...
    title="Hub Rotation Rate (rad/s)",...xlabel="Time (s)",ylabel="X angular speed")
plot(y(18,1),{graph_number=2,xlabel="Time (s)",ylabel="Y angular speed")
plot(y(19,1),{graph_number=3,xlabel="Time (s)",ylabel="Z angular speed")

pause "Show RW speeds"
plot(y(20,1),{rows=2,columns=2,graph_number=1,...
    title="Reaction Wheel Spin Rates (rad/s)",...xlabel="Time (s)",ylabel="X Spin Rate")
plot(y(21,1),{graph_number=2,xlabel="Time (s)",ylabel="Y spin rate")
plot(y(22,1),{graph_number=3,xlabel="Time (s)",ylabel="Z spin rate")

pause "Show CMG information"
plot(y(23,1),{rows=5,columns=2,graph_number=1,...
    title="CMG Axis Angles (rad)",...xlabel="",ylabel="Rib 1")
plot(y(24,1),{graph_number=3,xlabel="",ylabel="Rib 2")
plot(y(25,1),{graph_number=5,xlabel="",ylabel="Rib 3")
plot(y(26,1),{graph_number=7,xlabel="",ylabel="Rib 4")
plot(y(27,1),{graph_number=9,xlabel="Time (s)",ylabel="Rib 5")
plot(y(28,1),{graph_number=2,...
    title="CMG Spin Rates (rad/s)",...xlabel="",ylabel="Rib 1")
plot(y(29,1),{graph_number=4,xlabel="",ylabel="Rib 2")
plot(y(30,1),{graph_number=6,xlabel="",ylabel="Rib 3")
plot(y(31,1),{graph_number=8,xlabel="",ylabel="Rib 4")
plot(y(32,1),{graph_number=10,xlabel="Time (s)",ylabel="Rib 5")

display "Done."
```

A.2 Parameter Setup Function: set_parameters.msf

This function sets up the parameters required to execute ProtoSim:

[S] = known specification
[A] = assumed value

[D] = specification currently in the DICE Current Design Document
[-] = specification is NOT in the DICE Current Design Document

function {flag} = set_parameters();

# * Create Partitions for Variables *

if !exist(dice,{partition});
    new partition dice;       # Partition for general DICE variables.
endIf;

if !exist(irs,{partition});
    new partition irs;       # Partition for rate sensor variables.
endIf;

if !exist(svs,{partition});
    new partition svs;       # Partition for SVS variables.
endIf;

if !exist(lep, {partition});
    new partition lep;       # Partition for LEPD variables.
endIf;

if !exist(wheel, {partition});
    new partition wheel;     # Partition for reaction wheel variables.
endIf;
if !exist(ResetInt, {partition});
    new partition ResetInt;  # Partition for JMC's Reset Integrator.
endif;

if !exist(thruster, {partition});
    new partition thruster;  # Partition for air thruster parameters.
endif;

if !exist(cmg, {partition});
    new partition cmg;
endif;

if !exist(cmgencoder, {partition});
    new partition cmgencoder;  # Partition for CMG encoder parameters.
endif;

if !exist(gmotor, {partition});
    new partition gmotor;  # Partition for CMG gimbal motor.
endif;

if !exist(smotor, {partition});
    new partition smotor;  # Partition for CMG spin motor.
endif;

if !exist(pot, {partition});
    new partition pot;  # Partition for potentiometer parameters.
endif;

if !exist(rwencoder, {partition});
    new partition rwencoder;  # Partition for reaction wheel encoder.
endif;

if !exist(pd, {partition});
    new partition pd;  # Partition for PD controller parameters.
endif;

# * General DICE Parameters *

dice.radius = 0.0762;  # Radius of the DICE hub (m) [SI]
dice.rib = 0.4763;  # Length of DICE rib (m) [SI]
dice.ribs = 5;  # Number of ribs [SI]
# dice.maxtrans = 0.3;  # Maximum anticipated bus translation (m) [SI].
dice.maxtrans = 0.1;
dice.maxrot = 0.785;  # Maximum anticipated bus rotation (rad) [SI].

dice.ribangle = (2*pi)/dice.ribs;  # Angle to rotate between ribs.

# * Reset Integrator (by James Crawford) Parameters *

ResetInt.stiction = 0;  # Stiction parameter [SI].
ResetInt.gap = (2*pi)/1000;  # Size of "stuck" region (rad) [SI].

# * Inertial Rate Sensor Parameters (Systron Donner QRS11-00100-100) *

irs.K = 1.4324;  # Rate sensor constant (Vs/rad) [SI]
irs.limit = 1.7453;  # Upper detection limit (rad/s) [SI]
irs.delay = 0;  # Time delay (s) [A,-]
irs.resolution = 6.9813E-5;  # Sensor resolution (rad/s) [SI]
irs.noise = 1.3753E-3;  # Maximum systemic noise (rad/s) [SI]
irs.bias = 3.4907E-5; # Systemic bias (rad/s) [S,D]

irs.K = irs.K*ones(3,1);
irs.limit = irs.limit*ones(3,1);
irs.resolution = irs.resolution*ones(3,1);
irs.noise = irs.noise*ones(3,1);

# Sampling rate set in "rate_sensor_sampler" block:
# (currently 100 Hz) [A].

# * SVS Parameters *
svs.dposition = 1E-4;    # Position accuracy (m) [S,-]
svs.dangle = 6.727E-4;   # Angular accuracy (rad) [S,-]
svs.bits = 8;     # Resolution (bits) [A,-].
svs.transducer = 0;  # Transducer latency (s) [A,-]
svs.comm = 0;      # Communications latency (s) [A,-]

svs.pos_res = (dice.mastrans/(2^(svs.bits)))*ones(3,1);
svs.ang_res = (dice.rnaxrot/(2^(svs.bits)))*ones(3,1);
svs.dposition = svs.dposition*ones(3,1);
svs.dangle = svs.dangle+ones(3,1);

# Frame rate set in "svs_frame_rate" block (currently 30 Hz) [A].

# * LEPD Parameters *
lepd.offset = [0; 0];    # LEPD target offset [y,z] (m) [A,-]
lepd.accuracy = 5E-4;    # Position accuracy (m) [S,-]
lepd.latency = 0;        # LEPD latency (s) [A,-]
lepd.resolution = 12;    # LEPD resolution (bits) [A,-]
lepd.f = 1.27E-2;        # Focal length of LE PD lens (m) [S,D]
lepd.ic = 4E-3;          # Size of detector array (m) [S,D]
lepd.p = 0.48;           # Distance from LED to LE PD (m) [S,D]
lepd.V = 10;             # Maximum output voltage (V) [S,-]

lepd.ofx = [-lepd.offset(2), lepd.offset(1)];
lepd.ofx = [lepd.ofx, lepd.ofx, lepd.ofx, lepd.ofx, lepd.ofx];
lepd.accuracy = lepd.accuracy*ones(1,10);
lepd.fov = (lepd.ic*(lepd.p-lepd.f))/(2*lepd.f);
lepd.resolution = lepd.fov/(2*lepd.resolution)*ones(1,10);
lepd.K = (lepd.V/lepd.fov)*ones(1,10);
lepd.fov = lepd.fov*ones(1,10);

# Sampling rate set in "lepd_sampler" block (currently 100 Hz) [A].

# * Reaction Wheel -- General Parameters *
wheel.ls = 7.3E-3;       # Reaction wheel inertia (kg.m2) [S,-]
whee1.B = 0;            # Reaction wheel damping constant (Nm.s) [A,-]
wheel.noise = 3E-4;      # Maximum amplitude of noise torque (Nm) [A,-]
wheel.noise = 0;
wheel.Tlim = 1;          # Reaction wheel torque limit (Nm) [S,D]
wheel.ns = 1;            # Reaction wheel gear ratio [A,-]
wheel.ws = 523.599;      # Maximum reaction wheel speed (rad/s) [S,D]

wheel.noise = wheel.noise*ones(3,1);
wheel.Tlim = wheel.Tlim*ones(3,1);
wheel.n = wheel.ns*ones(3,1);
wheel.ws = wheel.ws*ones(3,1);
# * Reaction Wheel -- Motor Parameters (Kollmorgen RBE(H)-01212) *

wheel.isat = 10; # Current saturation level (A) [S,-]
wheel.Rl = 0.732; # Load resistance (ohms) [S,-]
wheel.Kt = 0.0735; # Motor constant (Nm/A) [S,-]
wheel.Kb = wheel.Kt; # Back emf constant
wheel.Jm = 1.8E-3; # Rotor inertia (kg.m2) [S,D]
wheel.Fi = 8.0596E-4; # Viscous damping constant (Nm.s) [S,-]
wheel.Tf = 0.0021; # Maximum static friction (Nm) [S,-]
wheel.switchF = 1; # Activate ResetInt friction (1=on, 0=off)

wheel.Kr = wheel.Tf/ResetInt.gap;
whee1.isat = wheel.isat*ones(3,1);
whee1.Kt = wheel.Kt+ones(3,1);
whee1.Tf = wheel.Tf*ones(3,1);
whee1.Jtotal = wheel.Jm + ((wheel.ns)^2)*whee1.Is; # Total inertia.
whee1.dtotal = wheel.Fi + ((wheel.ns)^2)*whee1.B; # Total damping.
whee1.switchF = wheel.switchF*ones(3,1);
whee1.B = wheel.dtotal*ones(3,1);

# * CMG -- General Parameters *

cmg.ns = 0.25; # CMG gear ratio [S,-]
cmg.I = 2.00E-4; # CMG wheel inertia (kg.m2) [S,D]
cmg.Tlim = 0.019; # Maximum generated torque (Nm) [S,D]
cmg.slewlim=1.6516; # Maximum CMG slew rate (rad/s) [A,-]
cmg.glim = 0.349; # Maximum CMG gimbal deflection (rad) [S,D]
cmg.B = 0; # Damping factor of CMG wheel (Nm.s) [A,-]
cmg.wa = 7000*(2*pi)/60; # Nominal CMG wheel speed (rad/s) [S,D]

cmg.n = cmg.ns*ones(3,1);
cmg.Tlim = cmg.Tlim*ones(3,1);
cmg.Is = cmg.I*ones(3,1);

# * CMG -- Gimbal Motor Parameters (Kollmorgen QT-0717) *

gmotor.Kt = 5.1973E-3; # Motor constant (Nm/A) [S,-]
gmotor.Jm = 1.55E-6; # Rotor inertia (kg.m2) [A,-]
gmotor.I = 5.8996E-4; # Motor assembly inertia (kg.m2) [S,-]
gmotor.Fi = 2.65E-5; # Viscous damping constant (Nm.s) [A,-]
gmotor.noise = 0; # Maximum systemic noise (Nm) [A,-]
gmotor.Tmax = 0.028; # Maximum torque of gimbal motor (Nm) [A,-]
gmotor.Isat = 3; # Gimbal motor saturation current (A) [A,-]
gmotor.B = 0; # Damping of gimbal assembly (Nm.s) [A,-]
gmotor.switchF = 1; # Activate ResetInt friction (1=on, 0=off)

gmotor.Kr = gmotor.Tf/ResetInt.gap;
gmotor.Jtotal = gmotor.Jm + ((cmg.ns)^2)*gmotor.I; # Total inertia.
gmotor.dtotal = gmotor.Fi + ((cmg.ns)^2)*gmotor.B; # Total damping.
gmotor.I = gmotor.dtotal*ones(3,1);

# * CMG -- Spin Motor Parameters (Kollmorgen QT-0717) *

smotor.ns = 1; # Spin motor gear ratio.
smotor.Vsat = 24; # Voltage saturation level (V) [S,-]
smotor.Rl = 1.91; # Load resistance (ohms) [S,-]
smotor.Kt = 5.1973E-3; # Motor constant (Nm/A) [S,-]
smotor.Jm = 1.55E-6;  # Rotor inertia (kg.m2) [A,-]
smotor.Tf = 1.808E-6;  # Maximum static friction (Nm) [A,-]
smotor.Fi = 2.85E-6;  # Viscous damping constant (Nm.s) [A,-]
smotor.Kb = smotor.Kt;  # Back emf constant
smotor.switchF = 1;  # Activate ResetInt friction (1=on, 0=Off)
smotor.Kr = smotor.Tf/ResetInt.gap;  # Friction "stiffness" for Reset integrator.

smotor.Jtotal = smotor.Jm + ((smotor.ns)^2)*cmg.I;  # Total inertia.
smotor.Btotal = smotor.Fi + ((smotor.ns)^2)*cmg.B;  # Total damping.


### # CMG -- Encoder Parameters (HP digital encoder HEDS-5505-J) *

# CMG encoder measures speed by (number of pulses)/time frame.

cmgenencoder.delay = 0;  # Time delay (s) [A,-]
cmgenencoder.ppr = 1024;  # Pulses per revolution (ppr) [S,D]
cmgenencoder.noise = 0;  # Maximum error, ie. erroneous pulses [A,-]

# Frame rate set in "cmg_encoder_frame_rate" block
# (currently 100 Hz) [A].


cmgenencoder.noise = cmgenencoder.noise*ones(5,1);
cmgenencoder.resolution = (2*pi/cmgenencoder.ppr)*ones(5,1);

# * Reaction Wheel - Encoder Parameters (HP Optical or Integral Tachometer) *

# Reaction wheel encoder measures speed by (pulse spacing)/(time-per-
# pulse).

rwencoder.delay = 0;  # Time delay (s) [A,-]
rwencoder.ppr = 4096;  # Pulses per revolution (ppr) [S,D]
rwencoder.noise = 0;  # Maximum error, ie. erroneous pulses [A,-]

# Frame rate set in "rw_encoder_frame_rate" block
# (currently 100 Hz) [A].

rwencoder.noise = rwencoder.noise*ones(3,1);
rwencoder.resolution = (2*pi/rwencoder.ppr)*ones(3,1);

# * Potentiometer Parameters (Spectra-Symbol SoftPot, contact strip pot.) *

pot.limit = cmg.lim;  # Maximum gimbal deflection (rad) [S,D].
pot.R2 = 10000;  # Upper resistance limit of potentiometer (ohms) [S,-].
pot.R1 = 0;  # Lower resistance limit of potentiometer (ohms) [S,-].
pot.i = 0.001;  # Operating current (A) [A,-]
pot.delay = 0;  # Time delay (s) [A,-]
pot.resolution = 12;  # Potentiometer resolution (bits) [A,D]
pot.noise = 0;  # Maximum gimbal "free-play" (rad) [A,-]

pot.K = (pot.R2-pot.R1)/(2*pot.limit);  # Potentiometer constant (ohms/rad).

# Sampling rate in "pot_sampler" block (currently 100 Hz) [A,-]

pot.resolution = (pot.limit/2^pot.resolution)*ones(5,1);
pot.K = pot.K*ones(5,1);
\text{pot.i} = \text{pot.i*ones(5,1)};
\text{pot.limit} = \text{pot.limit*ones(5,1)};
\text{pot.noise} = \text{pot.noise*ones(5,1)};

\# \text{**PD Controller Parameters**} \\
\text{pd.f} = 0.2; \quad \# \text{Control frequency (Hz)}.
\text{pd.zeta_a} = 0.05; \quad \# \text{Axial damping ratio}.
\text{pd.zeta_t} = 0.05; \quad \# \text{Transverse damping ratio}.
\text{pd.Ia} = 0.3512; \quad \# \text{Axial inertia of hub (kg.m²)}.
\text{pd.It} = 0.9022; \quad \# \text{Transverse inertia of hub (kg.m²)}.

\text{pd.rw} = \text{wheel.Jtotal/(wheel.Is*wheel.Kt(1))};
\text{pd.wc} = 2*\pi*pd.f;

\text{pd.Kpa} = \text{pd.rw*pd.Ia*(pd.wc^2)}; \quad \# \text{Axial P gain}.
\text{pd.Kpt} = \text{pd.rw*pd.It*(pd.wc^2)}; \quad \# \text{Transverse P gain}.
\text{pd.Kda} = \text{pd.rw*2*pd.zeta_a*pd.wc*pd.Ia}; \quad \# \text{Axial D gain}.
\text{pd.Kdt} = \text{pd.rw*2*pd.zeta_t*pd.wc*pd.It}; \quad \# \text{Transverse D gain}.


\text{flag} = 1;

e\text{ndFunction};
A.3 Random Input Generation Function: random_input.msf

%! *** Random Input Function ***

Written By : Eric M. Choi
Last Modified : 01/23/97

This function generates a matrix of random input for use with various simulations.

Inputs:  
\[ dt = \text{sampling interval (s)} \]
\[ t = \text{time vector (t)} \]
\[ \text{channels} = \text{number of inputs} \]
\[ fh = \text{highest input frequency (Hz)} \]

Output:  
\[ u_{\text{raw}} = \text{raw matrix of random input} \]

function [u_raw] = random_input(dt, duration, inputs, fh):

# order = 6;                # Order of the filter.
order = 4;
nyquist = 0.5/dt-100*eps;  # Define the Nyquist frequency (Hz).

set distribution normal;

n = duration/dt + 1;
unif = zeros(n, inputs);

if fh > nyquist;
  fcut = nyquist;
else;
  fcut = fh;
endIf;

for i=1:inputs;
  set seed=i;
  unif = random(t);    # Generate random sequence.
  # Filter the output, ensure a zero bias, and scale it to unity.
  sys = butterworth(order, [Fc=fcut, dT=dt, lowPass]);
  unif = filter(sys, unif, [zeroPhase]);
  unif = unif/max(abs(unif));
  unif = unif - mean(unif);
  u_raw(:, i) = [0; unif];
endFor;
endFunction;
A.4 NASTRAN Matrices Setup Function: simmattipcon.msf

#!
Function:    SimMatricesTipcon.msf
Author:      S. R. Piggott
Department:  UTIAS
Date:        April 8, 1996

Purpose:    To define the matrices required for the simulation of
the structural (NASTRAN) model.

Input:      zeta = modal damping ratio
            gyricity = 0, no gyricity; 1, nominal gyricity
            sysbld_block = name of system build file

Output:     (none)

Modification History:

<table>
<thead>
<tr>
<th>Date</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>97/02/18</td>
<td>E. M. Choi</td>
<td>Force the first six eigenvalues to be zero to ensure that the rigid modes are &quot;clean&quot;.</td>
</tr>
<tr>
<td>97/01/28</td>
<td>R. E. Zee</td>
<td>Modified to allow for different simulators. Added &quot;ProtoSim_REZ&quot; and &quot;ProtoSim_Rel&quot;.</td>
</tr>
<tr>
<td>97/01/07</td>
<td>R. E. Zee</td>
<td>Made a sign correction in the G_node matrix. Also corrected K_posomega and K_negomega.</td>
</tr>
<tr>
<td>96/12/05</td>
<td>E. M. Choi</td>
<td>Zeroed out the Ghat matrix because gyricity will be provided by the actuator models.</td>
</tr>
<tr>
<td>96/11/22</td>
<td>E. M. Choi</td>
<td>Break Lambda_tilde matrix into two gains K_alpha and K_omega, consisting of the diagonal and skew diagonal elements, to make Gyrosim run a little faster.</td>
</tr>
<tr>
<td>96/11/18</td>
<td>E. M. Choi</td>
<td>Modified to extract rib tip translation information.</td>
</tr>
<tr>
<td>96/08/23</td>
<td>S. R. Piggott</td>
<td>Changed the gyricity calculation to SI or MKS units.</td>
</tr>
<tr>
<td>96/06/17</td>
<td>S. R. Piggott</td>
<td>Altered to hard-zero the offdiagonal of Lambda_tilde.</td>
</tr>
</tbody>
</table>

function [flag] = simmattipcon(zeta,gyricity,sysbld_block)

#---------
# Constants
#---------
nribs = 5;  # Number of ribs.
lambda = 2*pi/nribs;  # Rib separation angle.

#-----------------------------------------------
# Determine hub to rib frame rotation matrices Cbd
Cbd = [];  # For each rib.
for i = 1:nribs
    angle = (i-1)*lambda;  # Separation angle from Rib 1.
    ca = cos(angle);
    sa = sin(angle);
    Cbd = [Cbd; ca, sa, 0;  # Rotation matrix from Fbi to Fd.
         -sa, ca, 0;
         0, 0, 1];
endfor
main.Cb1d = Cbd(1:3,:);
main.Cb2d = Cbd(4:6,:);
main.Cb3d = Cbd(7:9,:);
main.Cb4d = Cbd(10:12,:);
main.Cb5d = Cbd(13:15,:);
main.Cdb1 = main.Cb1d';
main.Cdb2 = main.Cb2d';
main.Cdb3 = main.Cb3d';
main.Cdb4 = main.Cb4d';
main.Cdb5 = main.Cb5d';

#-------------------------------------------
# Calculate required matrices for simulator
#-------------------------------------------

if sysbld_block == "protosim_970217"
    load "mks_eigdata_nosprings.xmd"
    #-------------------------------
    # Model reduction -- hard set for now.
    #-------------------------------
    nm = 31;
    eigval = main.eigval(1:nm);
    eigval(1:6) = 0;
    main.eigval(1:6) = 0;
    eigvec = main.eigvec(:,1:nm);
    nodes = main.nodes;
    #-------------------------------
    # Divide eigenvalues and eigenvectors.
    #-------------------------------
    nr = 6;
    ne = nm - nr;
    Omsq = eigval(nr+1:nm);
    Om = sqrt(Omsq);
    Ominv = inv(diag(Om));
    Er = eigvec(:,1:nr);
    Ee = eigvec(:,nr+1:nm);
    #-------------------------------
    # Form Gtip gyricity matrix: (lb-in2 changed to kg.m2).
    #-------------------------------
    I = 1.07357*(0.0254^2)/2.2046;  # 10,000 rpm.
    omw = 10000*2*pi/60;
\texttt{h = I}\texttt{^}\texttt{t} \texttt{omw;} \\
\texttt{Gr\_node = h\_cross([1;0;0]);} \\
\texttt{z = zeros(3,3);} \\
\texttt{G\_node = [z, z; z, -Gr\_node];} \\
\texttt{G\_tips = block\_diag(list(G\_node, G\_node, G\_node, G\_node, G\_node, G\_node));} \\
\texttt{tipnodes = [4001, 4002, 4003, 4004, 4005];} \\

\texttt{# Form Ghat -- gyricity transformed into real-modes' space.} \\
\texttt{# Extract the relevant part of the eigenvector matrix.} \\
\texttt{#} \\
\texttt{tmp = tipnodes';} \\
\texttt{selnodeset = tmp(1,:);} \\
\texttt{nnode = length(selnodeset);} \\
\texttt{p = [ ];} \\
\texttt{for i=1:nnode} \\
\texttt{ p = find(nodes==selnodeset(i)) + 6 - 5;} \\
\texttt{ q = i + 6 - 5;} \\
\texttt{ sel(q:q+5) = [p:p+5];} \\
\texttt{endFor;} \\
\texttt{E\_tips = eigvec(sel,:);} \\
\texttt{# Do transform.} \\
\texttt{if gyricity} \\
\texttt{ Ghat = E\_tips^\_t G\_tips E\_tips;} \\
\texttt{else} \\
\texttt{ Ghat = zeros(nm,nm);} \\
\texttt{# Zero gyricity (get from actuators).} \\
\texttt{endif} \\

\texttt{# Form scriptE prime.e} \\
\texttt{#} \\
\texttt{script\_Ep = [eye(nm, nm), zeros(nm, ne);} \\
\texttt{ zeros(ne, nm), Ominv ];} \\

\texttt{# Form S\_hat.} \\
\texttt{#} \\
\texttt{Om\_tilde = [zeros(ne, nr), diag(Om)];} \\
\texttt{S\_hat = [ Ghat, Om\_tilde^\_t;} \\
\texttt{ -Om\_tilde, zeros(ne, ne)];} \\

\texttt{# Quasi eigenproblem by Schur decomposition.} \\
\texttt{#} \\
\texttt{[Lambda\_tilde, Z] = schur(S\_hat, \{real\});} \\

\texttt{# Check results quality.} \\
\texttt{#} \\
\texttt{eps = 1.e-10;}
if (max(eig(Lambda_tilde+Lambda_tilde')) > eps) then
    display("********************************************************************")
    display("*** Poor Schur Decomposition results - not block diag")
    max(eig(Lambda_tilde+Lambda_tilde'))?
    stat = 1
    return
endIf;

if (max(Z'*Z - eye(Z)) > eps) then
    display("********************************************************************")
    display("*** Poor Schur Decomposition results - Z not orthogonal")
    max(Z'*Z - eye(Z))?  
    stat = 2
    return
endIf;

#---------------------------------------------------------------
# Insert modal damping into S_hat.
#---------------------------------------------------------------

[Nb, blocks] = extract_block_diag(Lambda_tilde);
damp_blocks = list([]);

for i=1:Nb
    l = blocks(i);
    om = abs(l(2,1));
    d = eye(2,2)*om*zeta + [0,-1;1,0]*om;
    damp_blocks = add_lists(damp_blocks, list(d));
endFor;

damp_blocks = damp_blocks(2:Nb+1);
Lambda_tilde_damp = block_diag(damp_blocks);

#---------------------------------------------------------------
# Calculate Gamma hat.
#---------------------------------------------------------------

E = selected_evec();
sel_q = [1,2,3];

for j=0:5
    sel_q = [sel_q, ([4,5,6]+j*6)];
endFor;

E_input = E(sel_q,:);
s = size(E_input);
nf = s(1);
Gamma_hat_con = Z'\script_Ep*[E_input'; zeros(ne, nf)];

#---------------------------------------------------------------
# Note E_input = [Er Ee] for the input points.
#---------------------------------------------------------------

sel_q = [];
    # Select all the modal coordinates for "E_con".
for j = 0:5;
    sel_q = [sel_q, ([1,2,3,4,5,6]+j*6)];
    # Include the tip translation and rates for "E_con".
endFor;

E_in = E(sel_q,:);
s = size(E_in);
nf = s(1);
E_con = block_diag(list(E_in(nf,:), E_in(nf,:)));

#----------------------------------
# Calculate output matrices.
#----------------------------------

C_hat = script_Ep*Z;

#----------------------------------
# Stash all the important variables in the "main" partition.
#----------------------------------

main.C_hat = C_hat;
main.Lambda_tilde = Lambda_tilde_damp;
main.Gamma_hat_con = Gamma_hat_con;
main.E_con = E_con;

K_alpha = diagonal(Lambda_tilde_damp,0)';
K_offdiag = diagonal(Lambda_tilde_damp,1)';

K_posomega = [K_offdiag, 0];
K_negomega = [0, -K_offdiag];

main.K_alpha = K_alpha;
main.K_posomega = K_posomega;
main.K_negomega = K_negomega;

main.Om = Om;

elseif sysbld_block == "protosim_REZ_970217"

#-----------------
# Load modal data
#-----------------

main.E = selected_evec();
[N,n] = size(main.E);
eigval = main.eigval(1:n);
eigval(1:6) = 0;
main.eigval(1:6) = 0;

nodes = main.nodes;

#------------------
# Squared frequency matrix
#------------------

main.Omega2 = eigval';

#-----------------
# Modal damping matrix
#-----------------
main.TwoPsi = 2*gamma*sqrt(eigval);
main.TwoPsi(1:6) = zeros(6,1);
main.TwoPsi = diagonal(main.TwoPsi);

# Input distribution matrix
#---------------------------

BO = [eye(6,6), zeros(6,15);
     zeros(3,21);
     zeros(3,6), eye(3,3), zeros(3,12);
     zeros(3,21);
     zeros(3,9), eye(3,3), zeros(3,9);
     zeros(3,21);
     zeros(3,12), eye(3,3), zeros(3,6);
     zeros(3,21);
     zeros(3,15), eye(3,3), zeros(3,3);
     zeros(3,21);
     zeros(3,18), eye(3,3)];
main.ETB0 = main.E'*BO;

# Output distribution matrix
#---------------------------

main.COE = eye(N,N)*main.E;

#----------------
# Gyricity
#---------

if gyricity
  I = 1.07357*(0.0254^2)/2.2049;
  omw = 10000*2*pi/60; # 10,000 rpm.
  h = I*omw;
  Gr_node = h*cross([1;0;0]);
  z = zeros(3,3);
  G_node = [z, -Gr_node];
  G_tips = block_diag(list(G_node, G_node, G_node, G_node, G_node));
  tipnodes = [4001, 4002, 4003, 4004, 4005];

  # Form Ghhat -- gyricity transformed into real-modes' space.  
  # Extract the relevant part of the eigenvector matrix.
  tmp = tipnodes';
  selnodeset = tmp(:,);
  nnode = length(selnodeset);
  p = [1];

  for i=1:nnode
    p = find(nodes==selnodeset(i))*6-5;
    q = i*6-5;
    sel(q:q+5) = [p:p+5];
  endfor;
  E_tips = eigvec(sel,:);  # Do transform.
E = selected_evec();
load "mks_eigdata_nosprings_rel.xmd"

[N,n] = size(main.eigvec);
eigval = main.eigval;
eigval(1:6) = 0;
main.eigval(1:6) = 0;

nodes = main.nodes;

main.omega2 = eigval';

main.TwoPsi = 2 * zeta * sqrt(eigval);
main.TwoPsi(1:6) = zeros(6,1);
main.TwoPsi = diagonal(main.TwoPsi);

B0 = [eye(6,6), zeros(6,15);
      zeros(3,21);
      zeros(3,6), eye(3,3), zeros(3,12);
      zeros(3,21);
      zeros(3,9), eye(3,3), zeros(3,9);
      zeros(3,21);
      zeros(3,12), eye(3,3), zeros(3,6);
      zeros(3,21);
      zeros(3,15), eye(3,3), zeros(3,3);
      zeros(3,21);
      zeros(3,18), eye(3,3)];
main.etaB0 = E''*B0;

main.COE = eye(N,N)*main.eigvec;

# Gyracity

#--------
if gyricity
    nnodes = length(nodes);
    nflex = N - 6;

    Is = 1.07357 + (0.0254^2)/2.2049;  # Inertia of spin wheel (kg.m^2)
    oms = 10000 * 2*pi/60;  # Spin rate (rad/s)
    hs = Is*oms;

    # Form gyricity matrix

    G = zeros(N,N);
    Gj = -hs*cross([1;0;0]);
    GG = block_diag(list(Gj,Gj,Gj,Gj,Gj));
    Goffdiag = GG*Cbd;
    G(4:6,4:6) = Cbd'*Goffdiag;

    for i = 1:nribs
        first = (i-1)*6+10;
        last = first+2;
        top = (i-1)*3+1;
        bot = top + 2;
        G(first:last,4:6) = Goffdiag(top:bot,:);
        G(4:6,first:last) = -Goffdiag(top:bot,:)';
        G(first:last,first:last) = GG(top:bot,top:bot);
    endfor

    main.TwoPsi = main.TwoPsi + main.eigvec'*G*main.eigvec;
endif
endif
flag=1;
return
def endFunction;
B. APPENDIX B: VERIFICATION RUNS

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.1</td>
<td>Energy Balance Test Results</td>
<td></td>
</tr>
<tr>
<td>B.1.1</td>
<td>Force Input</td>
<td>98</td>
</tr>
<tr>
<td>B.1.2</td>
<td>Torque Input</td>
<td>100</td>
</tr>
<tr>
<td>B.2</td>
<td>Momentum Balance Test Results</td>
<td></td>
</tr>
<tr>
<td>B.2.1</td>
<td>Force Input</td>
<td>102</td>
</tr>
<tr>
<td>B.2.2</td>
<td>Torque Input</td>
<td>103</td>
</tr>
<tr>
<td>B.3</td>
<td>Deterministic Test Results</td>
<td></td>
</tr>
<tr>
<td>B.3.1</td>
<td>Predicted Outputs for X- and Y-Axis Reaction Wheel Pulses</td>
<td>104</td>
</tr>
<tr>
<td>B.3.2</td>
<td>Simulation Results for X-Axis Reaction Wheel Pulse</td>
<td>105</td>
</tr>
<tr>
<td>B.3.3</td>
<td>Simulation Results for Y-Axis Reaction Wheel Pulse</td>
<td>109</td>
</tr>
<tr>
<td>B.3.4</td>
<td>Predicted Outputs for Z-Axis Reaction Wheel Pulse</td>
<td>113</td>
</tr>
<tr>
<td>B.3.5</td>
<td>Simulation Results for Z-Axis Reaction Wheel Pulse</td>
<td>114</td>
</tr>
<tr>
<td>B.3.6</td>
<td>Simulation Results for CMG #1 Pulse</td>
<td>117</td>
</tr>
<tr>
<td>B.3.7</td>
<td>Simulation Results for CMG #2 Pulse</td>
<td>118</td>
</tr>
</tbody>
</table>
Energy Input from Z-Axis Force Pulse

Kinetic Energy of Z-Axis Motion

Energy (J)

Time (s)
Time (s)

Energy (J)

0.00015
0.0001
5e-05

0
5
10
15
20
25
30

Energy Input from Y-Axis Torque Pulse

Kinetic Energy of Y-Axis Motion

Time (s)

Energy (J)

0.00025
0.0002
0.00015
0.0001
5e-05

0
5
10
15
20
25
30

100
Energy Input from Z-Axis Torque Pulse

Kinetic Energy of Z-Axis Motion
Angular Momentum from Y-Axis Rotation

Angular Momentum from Z-Axis Rotation
B.3.1 Predicted Outputs for X- and Y-Axis Reaction Wheel Pulses

Predicted Reaction Wheel Pulses

Predicted Reaction Wheel Torque

Predicted Reaction Wheel Rotation

Predicted Reaction Wheel Rotation
Graphs showing time (s) versus torque for X, Y, and Z axes.
B.3.4 Predicted Outputs for Z-Axis Reaction Wheel Pulse
C.1 OKID/ERA Parameter Sensitivity Runs
   Sampling Frequency and SI Quality  120
   Skew Time and SI Quality  121
   Sensor Resolution and SI Quality  122
   Actuator Resolution and SI Quality  123
   Actuator Noise and SI Quality  124
   Sensor Noise and SI Quality  125
   Input/Output Noise and SI Quality  126
   Simulation Time and SI Quality  127

C.2 ProtoSi~n System Identification Runs
C.2.1 Dynamics Block  128
C.2.2 Dynamics Block in Relative Coordinates  135
   One Minute Experiment  135
   Two Minute Experiment  142
   Three Minute Experiment  149
C.2.3 Sensor Sizing  156
   Hub Rotation from SVS  156
   Hub Rotation from Integrating Inertial Rate Sensor Signal  162
C.2.4 Full ProtoSim Simulator  169
Time (s)  [Resolution = 10 bits]  RMS Ratio: 1.1342e-11

Time (s)  [Resolution = 11 bits]  RMS Ratio: 2.0229e-11

Time (s)  [Resolution = 13 bits]  RMS Ratio: 2.1106e-11

Time (s)  [Resolution = 14 bits]  RMS Ratio: 2.7246e-11
Time (s) [Noise = 5%] RMS Ratio: 0.937644

Time (s) [Noise = 1%] RMS Ratio: 0.218016

Time (s) [Noise = 0.1%] RMS Ratio: 0.00242233

Time (s) [Noise = 0.01%] RMS Ratio: 0.000247602
X Translation (m)

Y Translation (m)

Z Translation (m)
Time (s) RMS Ratio: 0.0353288

Time (s) RMS Ratio: 0.035282
X Rotation (rad)

Y Rotation (rad)

Z Rotation (rad)

RMS Ratio: 0.0590635

RMS Ratio: 0.0255602

RMS Ratio: 0.0240182
Time (s)

RMS Ratio: 0.717253

Y Translation (m)

Time (s)

RMS Ratio: 0.493759

Z Translation (m)