Recursive Inertia Estimation with Semidefinite Programming

Zachary R. Manchester[*]
Harvard University, Cambridge, Massachusetts, 02138

Mason A. Peck[†]
Cornell University, Ithaca, New York, 14850

A recursive algorithm suitable for on-orbit estimation of a spacecraft's inertia tensor is presented. The estimation problem is formulated as a semidefinite program with explicit enforcement of positive-definiteness and other constraints on the elements of the inertia tensor, which enhances convergence and guarantees that a physically valid result is returned. A discrete mechanics formulation of the spacecraft dynamics allows data routinely generated by spacecraft attitude determination systems to be directly used as input to the estimator. Numerical examples demonstrate the performance of the algorithm on simulated telemetry data.

Nomenclature

F	=	symmetric positive-semidefinite matrix
$\mathcal{F}$	=	quaternion generalized force
h	=	time step
I	=	identity matrix
J	=	inertia matrix in body-fixed axes
$\boldsymbol{j}$	=	vector of inertia components in body-fixed axes
k	=	time index
$\mathcal{L}$	=	Lagrangian
N_r	=	number of reaction wheels
q	=	unit quaternion
Q	=	orthogonal matrix
R	=	upper-triangular matrix
s	=	slack variable
S	=	skew-symmetric cross-product matrix
$\mathcal{S}$	=	action
t	=	time
δ	=	variational derivative
$\boldsymbol{\phi}$	=	three-parameter incremental rotation
ϵ	=	small scalar
$\boldsymbol{\eta}$	=	arbitrary perturbation
$\boldsymbol{\rho}$	=	rotor angular momentum
$\boldsymbol{\tau}$	=	external torque
$\boldsymbol{\omega}$	=	angular velocity in body-fixed axes

I. Introduction

Knowledge of a spacecraft's inertia tensor is vital to the performance of guidance, navigation, and control algorithms. While estimates of the inertia can be calculated before launch by tabulating the masses and locations of spacecraft components, the accuracy of such estimates is limited. Additionally, a spacecraft's inertia may change in unpredictable ways throughout a mission due to fuel use, deployment failures, damage, or a variety of other reasons. As a result, the ability to estimate a spacecraft's inertia on orbit can offer improved pointing performance and robustness to modeling uncertainty and component failures.

Several algorithms for estimating a spacecraft's inertia parameters from telemetry data have been proposed. Bergmann, Walker, and Levy developed a filter for estimating inertia that ignores the nonlinear terms in the rigid body equations

[*]Postdoctoral Fellow, School of Engineering and Applied Sciences, 60 Oxford Street, Member, zmanchester@seas.harvard.edu
[†]Associate Professor, Sibley School of Mechanical and Aerospace Engineering, 208 Upson Hall, Member, mp336@cornell.edu

of motion [1]. Ahmed, Coppola, and Bernstein proposed an adaptive attitude-tracking controller that recovers the elements of the inertia matrix under certain excitation conditions [2]. Psiaki presented a batch least-squares algorithm for estimating inertia as well as actuator alignment and scaling parameters [3]. Tanygin and Williams developed a least-squares estimator based on kinetic energy and integration of the work done by actuator inputs [4]. Norman, Peck, and O'Shaughnessy proposed a recursive least-squares algorithm for estimating inertia and actuator alignment parameters based on angular momentum [5]. The algorithm developed here is most closely related to that of Keim, Behcet, and Shields, who formulated batch least-squares inertia estimation as a semidefinite program to enforce positive-definiteness and constrain the elements of inertia with upper and lower bounds [6].

All of the existing inertia estimation methods cited suffer from at least one of the following drawbacks: First, most do not take into account the physical constraints on the components of the inertia tensor, in particular, positive definiteness and the "triangle inequality" [7]. Without accounting for these constraints, estimation algorithms can return unphysical results and perform poorly when data is sparse or very noisy. Second, many algorithms require derivatives or integrals of measured spacecraft state variables as input, necessitating prefiltering that introduces additional complexity and numerical error. This study seeks to address both of these issues with a recursive estimator suitable for real-time implementation.

Two key ideas underlie the estimation algorithm developed in this paper. The first is semidefinite programming, which is introduced in section II, along with a review of the properties of inertia tensors and a brief overview of quaternion algebra primarily intended to introduce notation. The second is discrete mechanics, which is used in section III to develop a discrete-time equation of motion for a gyrostat spacecraft. Section IV then formulates batch inertia estimation as a semidefinite program. Section V uses the batch estimator as the basis for a recursive inertia estimation algorithm. Finally, numerical simulations involving both spinning and three-axis stabilized spacecraft are presented in section VI to demonstrate the performance of the recursive algorithm.

II. Background

This section first highlights some properties of inertia tensors that will be useful in subsequent sections. Brief overviews of semidefinite programming and the algebra of quaternions are then presented. Thorough treatments of the latter two topics are available in the books of Boyd and Vandenberghe [8], and Altmann [9].

A. Inertia Tensors

Inertia tensors have several distinguishing mathematical properties which can be revealed by analyzing the kinetic energy of a rigid body. For simplicity and clarity, bodies composed of discrete particles are treated here. However, the results also hold in the continuous case.

For a body composed of N particles, the kinetic energy is,

$$T = \frac{1}{2}\sum_{i=1}^{N} m_i \left(\boldsymbol{v}_i \cdot \boldsymbol{v}_i\right) \tag{1}$$

where m_i and $\boldsymbol{v}_i$ denote the mass and velocity, respectively, of particle i. If the motion of the body consists only of rotation about its center of mass, then the velocity of particle i is given by,

$$\boldsymbol{v}_i = \boldsymbol{\omega} \times \boldsymbol{r}_i \tag{2}$$

where $\boldsymbol{\omega}$ is the angular velocity of the body and $\boldsymbol{r}_i$ is the position of particle i relative to the center of mass. Substituting equation (2) into equation (1) results in

$$T = \frac{1}{2}\sum_{i=1}^{N} m_i \left(\boldsymbol{\omega} \times \boldsymbol{r}_i\right) \cdot \left(\boldsymbol{\omega} \times \boldsymbol{r}_i\right) \tag{3}$$

Making use of the skew-symmetric cross-product matrix,

$$S(\boldsymbol{x}) = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} \tag{4}$$

equation (3) can be expressed in body-fixed coordinates as

$$T = \frac{1}{2}\sum_{i=1}^{N} m_i (S(r_i)\,\omega)^{\mathsf{T}} (S(r_i)\,\omega) \tag{5}$$

Collecting terms in equation (5), the kinetic energy can be written in the standard way,

$$T = \frac{1}{2}\omega^{\mathsf{T}} J \omega \tag{6}$$

where J is the matrix of components of the inertia tensor expressed in body-fixed coordinates:

$$J = \sum_{i=1}^{N} m_i S(r_i)^{\mathsf{T}} S(r_i) \tag{7}$$

Taking a closer look at equations (6) and (7) reveals several properties of the matrix J. First, since the kinetic-energy quadratic form in equation (6) must be positive for any non-zero value of $\boldsymbol{\omega}$, J must be positive definite, denoted $J > 0$. Second, by expanding the matrix product in equation (7),

$$S(r)^{\mathsf{T}} S(r) = \begin{bmatrix} r_y^2 + r_z^2 & -r_x r_y & -r_x r_z \\ -r_x r_y & r_x^2 + r_z^2 & -r_y r_z \\ -r_x r_z & -r_y r_z & r_x^2 + r_y^2 \end{bmatrix} \tag{8}$$

it becomes clear that J is symmetric and that its diagonal elements are constrained by the triangle inequality [7],

$$\begin{aligned} J_{11} + J_{22} &\geq J_{33} \\ J_{11} + J_{33} &\geq J_{22} \\ J_{22} + J_{33} &\geq J_{11} \end{aligned} \tag{9}$$

where equality can hold only in the degenerate case of a body which is infinitesimally thin about one of the coordinate axes.

B. Semidefinite Programming

A semidefinite program (SDP) is an optimization problem of the form

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \boldsymbol{c}^{\mathsf{T}} \boldsymbol{x} \\ \text{subject to} \quad & F(\boldsymbol{x}) \geq 0 \end{aligned} \tag{10}$$

where $\boldsymbol{x}$ and $\boldsymbol{c}$ are vectors in $\mathbb{R}^n$ and $F(\boldsymbol{x})$ is an $m \times m$ symmetric matrix defined as follows,

$$F(\boldsymbol{x}) = F_0 + \sum_{i=1}^{n} x_i F_i \tag{11}$$

where F_0 and F_i are symmetric. The inequality constraint $F(\boldsymbol{x}) \geq 0$ means that $F(\boldsymbol{x})$ must be positive semidefinite. That is, $\boldsymbol{z}^{\mathsf{T}} F(x) \boldsymbol{z} \geq 0$ for all vectors $\boldsymbol{z}$ in $\mathbb{R}^m$. Constraints of this type are known as linear matrix inequalities (LMIs) [10]. SDPs are convex optimization problems. As a result, fast numerical algorithms that are guaranteed to converge (if a feasible solution exists) are available for solving them [8].

Many standard optimization problems, including linear and quadratic programs, can be put into SDP form [11]. In particular, the linear least-squares problem,

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|H\boldsymbol{x} - \boldsymbol{y}\|_2^2 \tag{12}$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{y} \in \mathbb{R}^m$, and $H \in \mathbb{R}^{m \times n}$, can be recast into an equivalent SDP by introducing a scalar slack variable s into the vector of optimization variables:

$$\boldsymbol{x}' = \begin{bmatrix} \boldsymbol{x} \\ s \end{bmatrix} \tag{13}$$

Using the fact that the Schur complement of a positive semidefinite matrix must also be positive semidefinite, the least-squares problem can be embedded into the following LMI,

$$F_{LS}(\boldsymbol{x}') = \begin{bmatrix} s & (H\boldsymbol{x} - \boldsymbol{y})^{\mathsf{T}} \\ (H\boldsymbol{x} - \boldsymbol{y}) & I \end{bmatrix} \geq 0 \tag{14}$$

which is equivalent to the scalar inequality

$$s - (H\boldsymbol{x} - \boldsymbol{y})^{\mathsf{T}}(H\boldsymbol{x} - \boldsymbol{y}) \geq 0 \tag{15}$$

$F_{LS}(\boldsymbol{x}')$ has the following expansion in the form of equation (11),

$$F_i = \begin{cases} \begin{bmatrix} 0 & -\boldsymbol{y}^{\mathsf{T}} \\ -\boldsymbol{y} & I \end{bmatrix} & \text{for } i = 0 \\ \begin{bmatrix} 0 & H_i^{\mathsf{T}} \\ H_i & I \end{bmatrix} & \text{for } i = 1 \ldots n-1 \\ \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} & \text{for } i = n \end{cases} \tag{16}$$

where H_i is the i^{th} column of H and the zeros in F_n should be interpreted as zero blocks of the appropriate sizes. The least-squares problem can now be put into the standard SDP form of equation (10):

$$\begin{aligned} &\text{minimize} && \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ s \end{bmatrix} \\ &\text{subject to} && F_{LS}(\boldsymbol{x}') \geq 0 \end{aligned} \tag{17}$$

Once an optimization problem is posed as an SDP, it can be modified by imposing a variety of additional LMI constraints. Multiple constraints can be enforced by forming a block-diagonal concatenation of their LMI representations. In this way, linear and quadratic problems subject to a large class of convex constraints can be solved efficiently.

C. Quaternion Algebra

Quaternions form an algebra with a non-commutative binary product operation. It is often convenient to think of them as four-dimensional objects composed of a three-dimensional vector part $\boldsymbol{v}$ and a scalar part s.

$$q = \begin{bmatrix} \boldsymbol{v} \\ s \end{bmatrix} \tag{18}$$

This representation allows the quaternion product to be written in terms of scalar and vector products [9]:

$$q_1 q_2 = \begin{bmatrix} \boldsymbol{v_1} \times \boldsymbol{v_2} + s_1 \boldsymbol{v_2} + s_2 \boldsymbol{v_1} \\ s_1 s_2 - \boldsymbol{v_1} \cdot \boldsymbol{v_2} \end{bmatrix} \tag{19}$$

Note that, in general, $q_1 q_2 \neq q_2 q_1$. Throughout the paper, quaternion products are indicated by juxtaposition, while scalar and vector products are indicated in the usual way, with the $\cdot$ and $\times$ symbols respectively.

Rotations can be conveniently represented by unit-length quaternions. If $\boldsymbol{r}$ is a unit vector in $\mathbb{R}^3$ representing the axis of rotation and θ is the angle of rotation, then the quaternion representing the rotation is as follows:

$$q = \begin{bmatrix} \boldsymbol{r} \sin(\theta/2) \\ \cos(\theta/2) \end{bmatrix} \tag{20}$$

Both q and $-q$ correspond to the same rotation, making the unit quaternions a "double cover" of the group of rotations.

The conjugate of a quaternion is denoted with a superscript $\dagger$ and represents the rotation about the same axis $\boldsymbol{r}$ by $-\theta$.

$$q^{\dagger} = \begin{bmatrix} -\boldsymbol{v} \\ s \end{bmatrix} \tag{21}$$

Two rotations can be composed by multiplying their quaternion representations. A quaternion q_3 representing a rotation q_1 followed by a rotation q_2 is simply $q_3 = q_2 q_1$. The rotation of a three-dimensional vector $\boldsymbol{x}$ by a unit quaternion q is

$$\hat{\boldsymbol{x}}' = q \hat{\boldsymbol{x}} q^{\dagger} \tag{22}$$

where $\hat{\boldsymbol{x}}$ indicates the formation of a quaternion with zero scalar part from the vector $\boldsymbol{x}$:

$$\hat{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{x} \\ 0 \end{bmatrix} \tag{23}$$

Finally, the subsequent analysis requires some kinematic identities relating quaternion derivatives to vector quantities more familiar in rigid body dynamics. First, the time derivative of a body's attitude quaternion is related to its angular velocity in the following way:

$$\hat{\omega} = 2q^\dagger \dot{q} \tag{24}$$

Second, the quaternion generalized force corresponding to a torque on the body is [12, 13]

$$\mathcal{F} = 2q\hat{\boldsymbol{\tau}} \tag{25}$$

Schaub and Junkins provide a thorough discussion of rigid body dynamics using quaternions [14].

III. Discrete Gyrostat Mechanics

A gyrostat is a system of coupled rigid bodies whose relative motions do not change the total inertia tensor of the system. This abstraction serves as a practical mathematical model for a spacecraft with reaction wheels. The fundamental differential equation governing the motion of a gyrostat is,

$$J \cdot \dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (J \cdot \boldsymbol{\omega} + \boldsymbol{\rho}) + \dot{\boldsymbol{\rho}} = \boldsymbol{\tau} \tag{26}$$

where J is the inertia tensor of the gyrostat taken about its center of mass, $\boldsymbol{\omega}$ is the body angular velocity, $\boldsymbol{\rho}$ is the total angular momentum stored in the rotors, and $\boldsymbol{\tau}$ is the external torque applied to the gyrostat [15].

In principle, the inertia can be estimated using equation (26) if time histories of the variables $\boldsymbol{\omega}$, $\dot{\boldsymbol{\omega}}$, $\boldsymbol{\rho}$, and $\dot{\boldsymbol{\rho}}$ are available. In practice, however, the angular acceleration $\dot{\boldsymbol{\omega}}$ is not measured directly and must be obtained from noisy measurements of $\boldsymbol{\omega}$. Many solutions to this problem have been proposed, including various finite difference and filtering schemes [6, 16] and integrating both sides of equation (26) with respect to time [3–5], but all add additional complexity and suffer to varying degrees from numerical error and noise amplification. Here, a new approach is taken in which discrete variational mechanics is used to derive a discrete-time version of equation (26) that does not contain $\dot{\boldsymbol{\omega}}$.

Discrete mechanics is a mathematical framework for deriving discrete-time algebraic equations of motion for mechanical systems based on classical variational mechanics [17]. The essential idea is to approximate the derivatives and integrals encountered in the standard formulation of Lagrangian mechanics with finite differences and quadrature rules *before* deriving equations of motion, rather than after. Discretizations derived from variational principles offer many advantages over more traditional schemes like Runge-Kutta methods, including momentum and energy conservation [17]. The remainder of this section gives a brief derivation of the discrete-time gyrostat equation originally presented by the authors in [18].

A. Torque-Free Motion

The Lagrangian for a gyrostat is,

$$\mathcal{L} = \frac{1}{2}\boldsymbol{\omega}_B \cdot J_B \cdot \boldsymbol{\omega}_B + \sum_{r=1}^{N_r} \frac{1}{2}(\boldsymbol{\omega}_B + \boldsymbol{\omega}_r) \cdot J_r \cdot (\boldsymbol{\omega}_B + \boldsymbol{\omega}_r) \tag{27}$$

where J_B is the carrier body's inertia tensor (including rotor masses) taken about the body's center of mass, $\boldsymbol{\omega}_B$ is the carrier body's angular velocity, the J_r are the rotor inertia tensors taken about each rotor's center of mass, the $\boldsymbol{\omega}_r$ are the rotor angular velocities relative to the carrier body, and the $\boldsymbol{x}_r$ are the rotor positions relative to the carrier body's center of mass. Using equation (27), Hamilton's principle states that the equation of motion for the gyrostat, in the absence of external torques, can be found by setting the variation of the action equal to zero [19]:

$$\delta\mathcal{S} = \delta \int_{t_0}^{t_f} \mathcal{L}\, \mathrm{d}t = 0 \tag{28}$$

The transition to discrete mechanics begins by breaking the action integral into short segments of length h, with $t_k = t_0 + kh$:

$$\delta\mathcal{S} = \delta \int_{t_0}^{t_f} \mathcal{L}\, \mathrm{d}t = \delta \sum_{k=0}^{N-1} \int_{t_k}^{t_{k+1}} \mathcal{L}\, \mathrm{d}t = 0 \tag{29}$$

The integral over a single time step on the right hand side of equation (29) is then approximated using a quadrature rule. First, the body's angular velocity is approximated by a finite difference of quaternions,

$$\hat{\omega}_k = 2\, q_k^\dagger \dot{q}_k \approx 2\, q_k^\dagger \left(\frac{q_{k+1} - q_k}{h}\right) = 2\left(\frac{f_k - 1}{h}\right) \tag{30}$$

where $f_k = q_k^\dagger q_{k+1}$ is the rotation between adjacent time steps. The rectangle rule is then applied to arrive at the following discrete Lagrangian [18],

$$\mathcal{L}_d = \frac{2}{h}\left[f_k \cdot \hat{J}_B \cdot f_k + \sum_{r=1}^{N_r}(f_k + \frac{h}{2}\hat{\omega}_{r,k}) \cdot \hat{J}_r \cdot (f_k + \frac{h}{2}\hat{\omega}_{r,k})\right] \tag{31}$$

where $\hat{J}_B$ and $\hat{J}_r$ are augmented 4×4 equivalents of J_B and J_r:

$$\hat{J} = \begin{bmatrix} J_{11} & J_{12} & J_{13} & 0 \\ J_{21} & J_{22} & J_{23} & 0 \\ J_{31} & J_{32} & J_{33} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{32}$$

Substituting equation (31) into equation (29) results in the following discrete action sum:

$$\mathcal{S}_d = \sum_{k=0}^{N-1} \frac{2}{h}\left[f_k \cdot \hat{J}_B \cdot f_k + \sum_{r=1}^{N_r}(f_k + \frac{h}{2}\hat{\omega}_{r,k}) \cdot \hat{J}_r \cdot (f_k + \frac{h}{2}\hat{\omega}_{r,k})\right] \tag{33}$$

Hamilton's principle can then be applied by setting the variational derivative of equation (33) equal to zero using a constrained variation of f_k which respects the quaternion unit-norm constraint [18],

$$^{\epsilon}f_k = f_k + \epsilon(f_k\hat{\boldsymbol{\eta}}_{k+1} - \hat{\boldsymbol{\eta}}_k f_k) \tag{34}$$

resulting in

$$\sum_{k=0}^{N-1}\left[f_k \cdot \hat{J}_B \cdot (f_k\hat{\boldsymbol{\eta}}_{k+1} - \hat{\boldsymbol{\eta}}_k f_k) + \sum_{r=1}^{N_r}(f_k + \frac{h}{2}\hat{\omega}_{r,k}) \cdot \hat{J}_r \cdot (f_k\hat{\boldsymbol{\eta}}_{k+1} - \hat{\boldsymbol{\eta}}_k f_k)\right] = 0 \tag{35}$$

To ensure that the variations $\boldsymbol{\eta}$ in equation (35) have the same time index, a "discrete integration by parts" is performed, which amounts to some simple index manipulation:

$$\sum_{k=1}^{N-1}\left[f_{k-1} \cdot \hat{J}_B \cdot (f_{k-1}\hat{\boldsymbol{\eta}}_k - \hat{\boldsymbol{\eta}}_k f_k) + \sum_{r=1}^{N_r}(f_{k-1} + \frac{h}{2}\hat{\omega}_{r,k-1}) \cdot \hat{J}_r \cdot (f_{k-1}\hat{\boldsymbol{\eta}}_k - \hat{\boldsymbol{\eta}}_k f_k)\right] = 0 \tag{36}$$

At this point equation (36), which implicitly includes unit-norm constraints on the quaternions, is converted to an unconstrained vector equation by parameterizing f_k in the following way:

$$f_k = \begin{bmatrix} \boldsymbol{\phi}_k \\ \sqrt{1 - \boldsymbol{\phi}_k \cdot \boldsymbol{\phi}_k} \end{bmatrix} \tag{37}$$

This parameterization is only valid for $|\boldsymbol{\phi}_k| < 1$. Therefore, h must be chosen small enough to ensure that the incremental rotations between adjacent time steps are less than 180°. A number of other 3-parameter attitude representations could be used instead, however, equation (37) is a natural choice that leads to simple and elegant expressions.

After substituting equation (37) and recognizing that equation (36) must hold for all variations $\boldsymbol{\eta}_k$, the discrete-time gyrostat equation is obtained,

$$\begin{aligned} &\sqrt{1 - \boldsymbol{\phi}_{k+1} \cdot \boldsymbol{\phi}_{k+1}}\,(J \cdot \boldsymbol{\phi}_{k+1} + \frac{h}{2}\boldsymbol{\rho}_{k+1}) + \boldsymbol{\phi}_{k+1} \times (J \cdot \boldsymbol{\phi}_{k+1} + \frac{h}{2}\boldsymbol{\rho}_{k+1}) \\ &\qquad = \sqrt{1 - \boldsymbol{\phi}_k \cdot \boldsymbol{\phi}_k}\,(J \cdot \boldsymbol{\phi}_k + \frac{h}{2}\boldsymbol{\rho}_k) - \boldsymbol{\phi}_k \times (J \cdot \boldsymbol{\phi}_k + \frac{h}{2}\boldsymbol{\rho}_k) \end{aligned} \tag{38}$$

where $\boldsymbol{\rho}$ is the total momentum stored in the rotors and J is the gyrostat inertia:

$$J = J_B + \sum_{r=1}^{N_r} J_r \tag{39}$$

B. External Torques

External torques can be added to equation (38) using the integral form of the Lagrange-D'Alembert principle [20],

$$\delta \int_{t_0}^{t_f} \mathcal{L}\, \mathrm{d}t + \int_{t_0}^{t_f} \mathcal{F} \cdot \delta q \, \mathrm{d}t = 0 \tag{40}$$

where the second term is the integral of the virtual work done by a generalized force $\mathcal{F}$. The discrete form of equation (40) is

$$\delta \sum_{k=0}^{N} \mathcal{L}_d + \sum_{k=0}^{N} \mathcal{F}_d^- \cdot \delta q_k + \mathcal{F}_d^+ \cdot \delta q_{k+1} = 0 \tag{41}$$

$\mathcal{F}_d^-$ and $\mathcal{F}_d^+$, known as discrete generalized forces, are defined as follows [17]:

$$\mathcal{F}_d^- = \int_{t_k}^{t_{k+1}} \frac{1}{2}\mathcal{F}(q,\dot{q}) \cdot \frac{\partial q(t)}{\partial q_k} \mathrm{d}t \tag{42}$$

$$\mathcal{F}_d^+ = \int_{t_k}^{t_{k+1}} \frac{1}{2}\mathcal{F}(q,\dot{q}) \cdot \frac{\partial q(t)}{\partial q_{k+1}} \mathrm{d}t \tag{43}$$

Inserting the expression for the quaternion generalized force given in equation (25) and applying the rectangle rule results in the following discrete quaternion generalized forces:

$$\mathcal{F}_d^- \approx h q_k \hat{\boldsymbol{\tau}}_k \tag{44}$$

$$\mathcal{F}_d^+ \approx h q_{k+1} \hat{\boldsymbol{\tau}}_{k+1} \tag{45}$$

Substituting these terms into equation (41) and performing some additional algebra reveals the forced discrete gyrostat equation:

$$\sqrt{1-\boldsymbol{\phi}_{k+1}\cdot\boldsymbol{\phi}_{k+1}}\,(J\cdot\boldsymbol{\phi}_{k+1} + \frac{h}{2}\boldsymbol{\rho}_{k+1}) + \boldsymbol{\phi}_{k+1} \times (J\cdot\boldsymbol{\phi}_{k+1} + \frac{h}{2}\boldsymbol{\rho}_{k+1}) + \frac{h^2}{2}\boldsymbol{\tau}_{k+1} = \sqrt{1-\boldsymbol{\phi}_{k}\cdot\boldsymbol{\phi}_{k}}\,(J\cdot\boldsymbol{\phi}_{k} + \frac{h}{2}\boldsymbol{\rho}_{k}) - \boldsymbol{\phi}_{k} \times (J\cdot\boldsymbol{\phi}_{k} + \frac{h}{2}\boldsymbol{\rho}_{k}) \tag{46}$$

Most attitude determination filters used on board spacecraft, including the ubiquitous multiplicative extended Kalman filter (MEKF) [21, 22], directly estimate $\boldsymbol{\phi}_k$ or a closely related three-parameter relative rotation at each time step. Therefore, equation (46) provides an ideal starting point for an inertia estimation scheme. Unlike equation (26), data readily available from existing spacecraft attitude determination systems can be used directly as input with little or no additional preprocessing.

IV. Batch Inertia Estimation

In this section, estimation of the components of the inertia tensor from a time history of attitude observations, rotor momenta, and external torques is formulated as a constrained batch least-squares problem. The resulting problem is then transformed into an SDP which can be efficiently solved.

While equation (46) is nonlinear in $\boldsymbol{\phi}$, it is linear in J. To make this explicit, the matrix $G(\boldsymbol{\phi})$ and the vector $\boldsymbol{j}$ are defined as follows:

$$G(\boldsymbol{\phi}) = \begin{bmatrix} \phi_1 & 0 & 0 & \phi_2 & \phi_3 & 0 \\ 0 & \phi_2 & 0 & \phi_1 & 0 & \phi_3 \\ 0 & 0 & \phi_3 & 0 & \phi_1 & \phi_2 \end{bmatrix} \tag{47}$$

$$\boldsymbol{j} = \begin{bmatrix} J_{11} \\ J_{22} \\ J_{33} \\ J_{12} \\ J_{13} \\ J_{23} \end{bmatrix} \tag{48}$$

In terms of G and $\boldsymbol{j}$, the matrix vector product $J\boldsymbol{\phi}$ becomes $G(\boldsymbol{\phi})\boldsymbol{j}$, allowing equation (46) to be rewritten as,

$$N(\boldsymbol{\phi}_{k+1})\left(G(\boldsymbol{\phi}_{k+1})\boldsymbol{j} + \frac{h}{2}\boldsymbol{\rho}_{k+1}\right) + \frac{h^2}{2}\boldsymbol{\tau}_{k+1} = M(\boldsymbol{\phi}_k)\left(G(\boldsymbol{\phi}_k)\boldsymbol{j} + \frac{h}{2}\boldsymbol{\rho}_k\right) \tag{49}$$

where the matrices $M(\boldsymbol{\phi})$ and $N(\boldsymbol{\phi})$ are defined as follows:

$$M(\boldsymbol{\phi}) = \sqrt{1 - \boldsymbol{\phi} \cdot \boldsymbol{\phi}}\, I - S(\boldsymbol{\phi}) \tag{50}$$

$$N(\boldsymbol{\phi}) = \sqrt{1 - \boldsymbol{\phi} \cdot \boldsymbol{\phi}}\, I + S(\boldsymbol{\phi}) \tag{51}$$

After collecting terms, equation (49) can be written as

$$H_k \boldsymbol{j} = \boldsymbol{y}_k \tag{52}$$

where the 3×6 matrix H_k and the 3×1 vector $\boldsymbol{y}_k$ are defined as follows:

$$H_k = N(\boldsymbol{\phi}_{k+1})G(\boldsymbol{\phi}_{k+1}) - M(\boldsymbol{\phi}_k)G(\boldsymbol{\phi}_k) \tag{53}$$

$$\boldsymbol{y}_k = M(\boldsymbol{\phi}_k)\frac{h}{2}\boldsymbol{\rho}_k - N(\boldsymbol{\phi}_{k+1})\frac{h}{2}\boldsymbol{\rho}_{k+1} - \frac{h^2}{2}\boldsymbol{\tau}_{k+1} \tag{54}$$

Given a set of attitude measurements collected over time $\boldsymbol{\phi}_1 \dots \boldsymbol{\phi}_N$, along with corresponding time histories of reaction wheel momenta $\boldsymbol{\rho}_1 \dots \boldsymbol{\rho}_N$ and external torques $\boldsymbol{\tau}_1 \dots \boldsymbol{\tau}_N$, the batch inertia estimation problem can be stated as,

$$\begin{aligned} \underset{\boldsymbol{j}}{\text{minimize}} \quad & \|H_{1:N}\boldsymbol{j} - \boldsymbol{y}_{1:N}\|_2^2 \\ \text{subject to} \quad & \begin{cases} J > 0 \\ J_{11} + J_{22} - J_{33} \geq 0 \\ J_{11} + J_{33} - J_{22} \geq 0 \\ J_{22} + J_{33} - J_{11} \geq 0 \end{cases} \end{aligned} \tag{55}$$

where $H_{1:N}$ and $\boldsymbol{y}_{1:N}$ are concatenations of the H_k and $\boldsymbol{y}_k$ matrices corresponding to each measurement:

$$H_{1:N} = \begin{bmatrix} H_1 \\ \vdots \\ H_N \end{bmatrix} \qquad \boldsymbol{y}_{1:N} = \begin{bmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_N \end{bmatrix} \tag{56}$$

The constrained least-squares problem (55) can then be transformed into an SDP using the results presented in section II:

$$\begin{aligned} \text{minimize} \quad & \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{j} \\ s \end{bmatrix} \\ \text{subject to} \quad & \begin{cases} J - \epsilon I \geq 0 \\ J_{11} + J_{22} - J_{33} \geq 0 \\ J_{11} + J_{33} - J_{22} \geq 0 \\ J_{22} + J_{33} - J_{11} \geq 0 \\ \begin{bmatrix} s & (H_{1:N}\boldsymbol{j} - \boldsymbol{y}_{1:N})^\intercal \\ (H_{1:N}\boldsymbol{j} - \boldsymbol{y}_{1:N}) & I \end{bmatrix} \geq 0 \end{cases} \end{aligned} \tag{57}$$

While the five LMI constraints in (57) have been written separately due to space constraints, they are concatenated to form a single block-diagonal LMI in practice. The ϵI term in the first constraint, where ϵ is a small positive constant, ensures that J is strictly positive-definite rather than positive-semidefinite. This constant should be chosen slightly larger than the error tolerance of the SDP solver being used.

In addition to enforcing positive-definiteness and the triangle inequality, a variety of other constraints can be included in the SDP formulation. This is especially useful for incorporating *a priori* knowledge into the estimator. For example, if bounds can be placed on the elements of J from modeling or ground-based testing, they can be easily accounted for by adding additional diagonal elements to the LMI in (57).

For J to be uniquely determined, a few conditions on the observations must be met. First, $\boldsymbol{\phi}_k$ must be changing in time. Otherwise the corresponding H_k matrices will be zero. Physically, this means that the spacecraft cannot be stationary or in a spin about a principle axis with no nutation. Second, some non-zero internal rotor momentum or known external torque must be applied to the spacecraft. Otherwise, the vectors $\boldsymbol{y}_k$ will be zero. In that case, the least-squares problem reduces to finding a vector in the null space of H, which is only determined up to an arbitrary scale factor.

V. Recursive Inertia Estimation

The batch estimation scheme of section IV suffers from one critical problem: As measurements are added, the size of the matrices in the LMI constraint in equation (57) grow. As a result, the computational cost of the algorithm can become prohibitive with just a few hundred measurements. Additionally, the full SDP problem must be re-solved each time a new measurement is added. This section develops a recursive version of (57) that enables efficient updating and only requires the solution of a small SDP of fixed-size at each time step.

The key to the recursive algorithm is the QR decomposition. A given matrix A can always be factored into the product of an orthogonal matrix Q and an upper-triangular matrix R [23, 24]. If A is rectangular with more rows m than columns n, the lower $m - n$ rows of R are entirely filled with zeros:

$$A = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \end{bmatrix} \tag{58}$$

Only the first n columns of Q and first n rows of R, denoted Q_1 and R_1 above, are actually needed to reconstruct A. Standard linear algebra routines are available for calculating Q_1 and R_1 given an input matrix A, known as a "thin" or "economy" QR decomposition [23, 24]. In the rest of this section it is assumed that such a routine is available and the "thin" $m \times n$ Q_1 and $n \times n$ R_1 matrices will simply be denoted Q and R.

By applying the QR decomposition to $H_{1:N}$, the least-squares problem (55) can be rewritten as,

$$\begin{aligned} \underset{\boldsymbol{j}}{\text{minimize}} \quad & \|R_{1:N}\boldsymbol{j} - \boldsymbol{z}_{1:N}\|_2^2 \\ \text{subject to} \quad & \begin{cases} J > 0 \\ J_{11} + J_{22} - J_{33} \geq 0 \\ J_{11} + J_{33} - J_{22} \geq 0 \\ J_{22} + J_{33} - J_{11} \geq 0 \end{cases} \end{aligned} \tag{59}$$

where $\boldsymbol{z}_{1:N} = Q_{1:N}^{\mathsf{T}} \boldsymbol{y}_{1:N}$. If a new set of observations H_{N+1} and $\boldsymbol{y}_{N+1}$ becomes available, R and $\boldsymbol{z}$ can be easily updated:

$$[Q_{1:N+1},\, R_{1:N+1}] = \text{qr}\left(\begin{bmatrix} R_{1:N} \\ H_{N+1} \end{bmatrix}\right) \tag{60}$$

$$\boldsymbol{z}_{1:N+1} = Q_{1:N+1}^{\mathsf{T}} \begin{bmatrix} \boldsymbol{z}_{1:N} \\ \boldsymbol{y}_{N+1} \end{bmatrix} \tag{61}$$

Once $R_{1:N+1}$ and $\boldsymbol{z}_{1:N+1}$ have been calculated, the constrained minimization (59) can be re-solved using the SDP formulation of section IV. Unlike (57), the sizes of all matrices in equations (59)–(61) remain fixed as measurements are added. Algorithm 1 summarizes the implementation of the recursive estimator.

Algorithm 1 Recursive Inertia Estimator

```
 1: $H_1 \leftarrow$ Calculate using equation (53)
 2: $\boldsymbol{y}_1 \leftarrow$ Calculate using equation (54)
 3: $[Q_1, R_1] \leftarrow \text{qr}(H_1)$
 4: $\boldsymbol{z}_1 \leftarrow Q_1^{\mathsf{T}} \boldsymbol{y}_1$
 5: for $k = 2 : N$ do
 6:     $H_k \leftarrow$ Calculate using equation (53)
 7:     $\boldsymbol{y}_k \leftarrow$ Calculate using equation (54)
 8:     $[Q_{1:k},\, R_{1:k}] \leftarrow \text{qr}\left(\begin{bmatrix} R_{1:k-1} \\ H_k \end{bmatrix}\right)$
 9:     $\boldsymbol{z}_{1:k} \leftarrow Q_{1:k}^{\mathsf{T}} \begin{bmatrix} \boldsymbol{z}_{1:k-1} \\ \boldsymbol{y}_k \end{bmatrix}$
10:     $J_k \leftarrow$ Solve constrained least-squares problem (59) using SDP
11: end for
```

VI. Numerical Examples

This section presents two test cases that demonstrate the performance of the recursive inertia estimation algorithm. First, a three-axis stabilized spacecraft is simulated performing a series of short slew maneuvers. Second, a spin-stabilized spacecraft is simulated spinning about its major axis of inertia with some nutation. The spinning case is used to illustrate the benefits of incorporating additional constraints into the estimator by bounding one of the moments of inertia to within 10% of a predetermined value, as might be available from pre-flight testing. Comparisons are made to two other inertia estimation algorithms: a naive SDP algorithm based on equation (26) that uses finite differences to obtain the required $\dot{\omega}$ data, and the momentum-based recursive inertia estimation algorithm of Norman, Peck, and O'Shaughnessy [5].

All simulations use MATLAB's ODE45 solver to integrate equation (26) with white noise disturbance torques applied to the spacecraft with a spectral density of 10^{-8} N·m/$\sqrt{Hz}$ in each axis. Simulated gyro measurements are also corrupted with white noise with a spectral density of 10^{-5} °/s/$\sqrt{Hz}$ in each axis. The true inertia in all cases is

$$J_{true} = \begin{bmatrix} 1.0035 & 0.0368 & -0.0678 \\ 0.0368 & 2.0047 & -0.0755 \\ -0.0678 & -0.0755 & 2.9918 \end{bmatrix}$$

The MATLAB-based SeDuMi SDP solver [25] is used in these examples. A wide variety of other SDP solvers, both free and commercial, are available [26].

In the first test case, a series of short slew maneuvers which could be executed on a three-axis stabilized spacecraft are simulated. No *a priori* information is used to initialize the estimation algorithms. Figure 1 shows the commanded rotor momentum components in body-fixed axes while figure 2 shows the simulated body-fixed angular velocity components for the slew maneuvers. Figure 3 shows the relative Frobenius norm error in the estimated inertia for all three

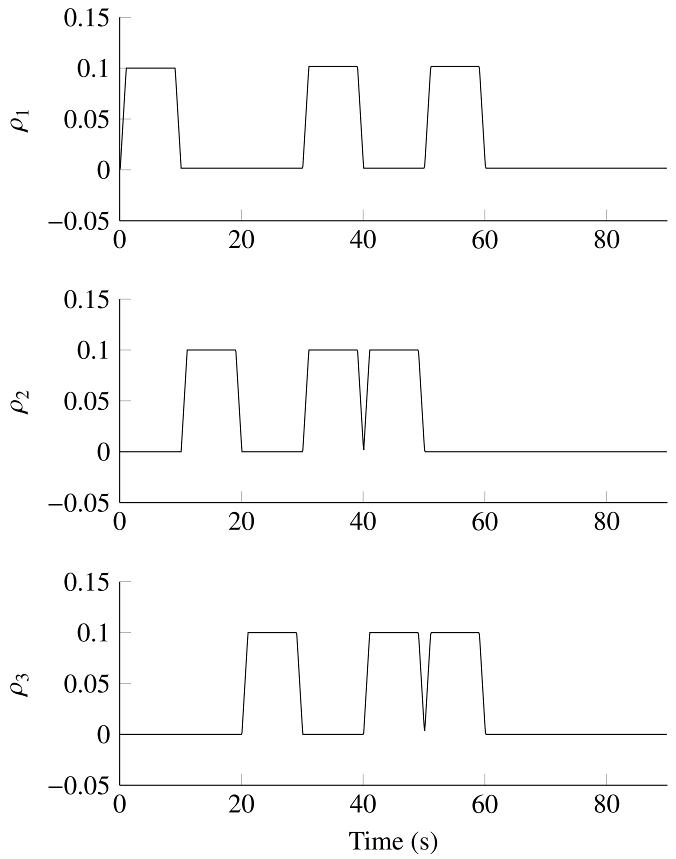

Figure 1. Rotor momentum components during slew maneuvers

algorithms. Figures 4 and 5 show the normalized errors in the estimated moments and products of inertia, respectively, over the course of the simulation.

Thanks to the additional constraints they take into account, the SDP-based algorithms have slightly better performance than the momentum-based scheme early in the simulation when little data is available. Later, the variational SDP algorithm and momentum-based algorithm have similar performance, while the SDP scheme based equation (26) has poor performance due to the use of finite differences to calculate $\dot{\omega}$.

The second test case demonstrates the advantages of including additional constraints in the SDP-based estimator. A spacecraft is simulated spinning about its major axis of inertia with some nutation. The only control input is a

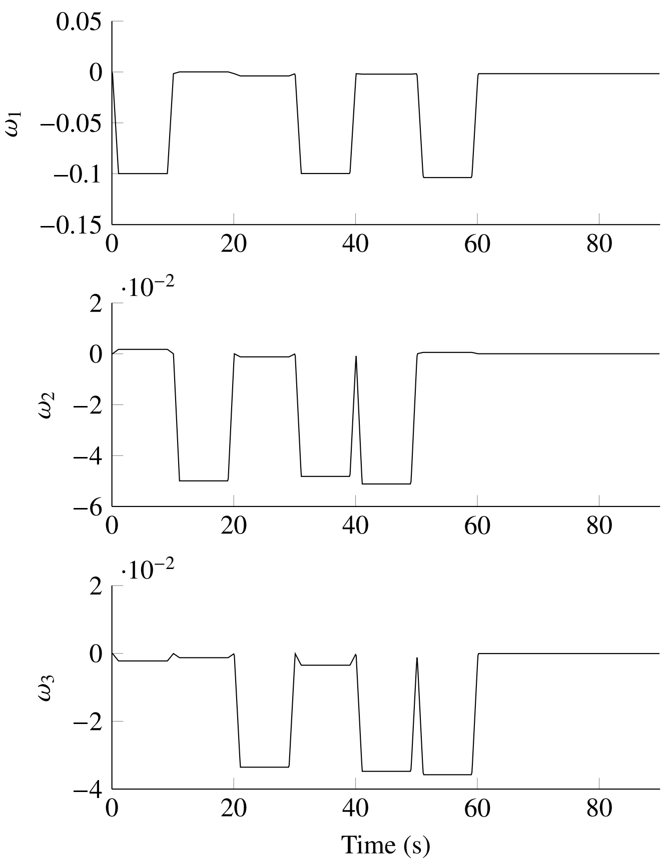

Figure 2. Body angular velocity components during slew maneuvers

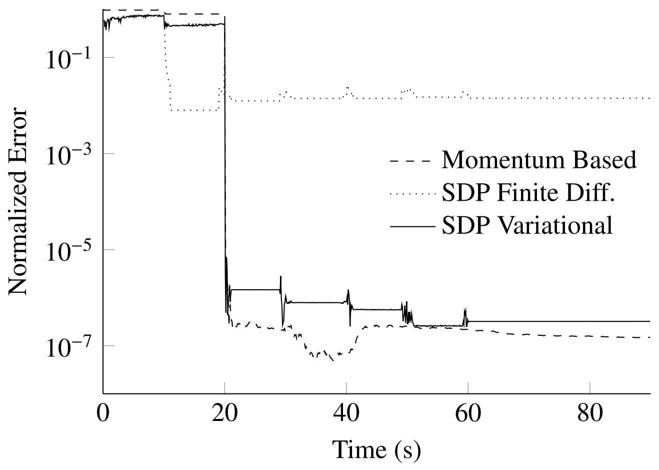

Figure 3. Normalized inertia error during slew maneuvers

short pulse applied to a single reaction wheel 30 seconds into the simulation. Figure 6 shows the simulated body-fixed angular velocity components, while figure 7 shows the rotor momentum components.

In the SDP-based estimators, the J_{33} component of the inertia matrix is constrained to be within 10% of a nominal value of 3 kg·m^2. Such an assumption is reasonable if, for example, a CAD model or ground-based experiment is used to calculate inertia components before launch. Figure 8 shows the relative Frobenius norm error in the estimated inertia for all three estimators. Normalized errors in the estimated moments of inertia are plotted in figure 9, while the normalized errors in the products of inertia are plotted in figure 10.

Here again, the SDP-based estimators achieve lower errors early in the simulation. Thanks to the inclusion of the 10% bound on J_{33}, they are able to estimate all of the other inertia components to within 10% as well. The momentum-based scheme, on the other hand, cannot provide any useful information until a control torque is applied. After the

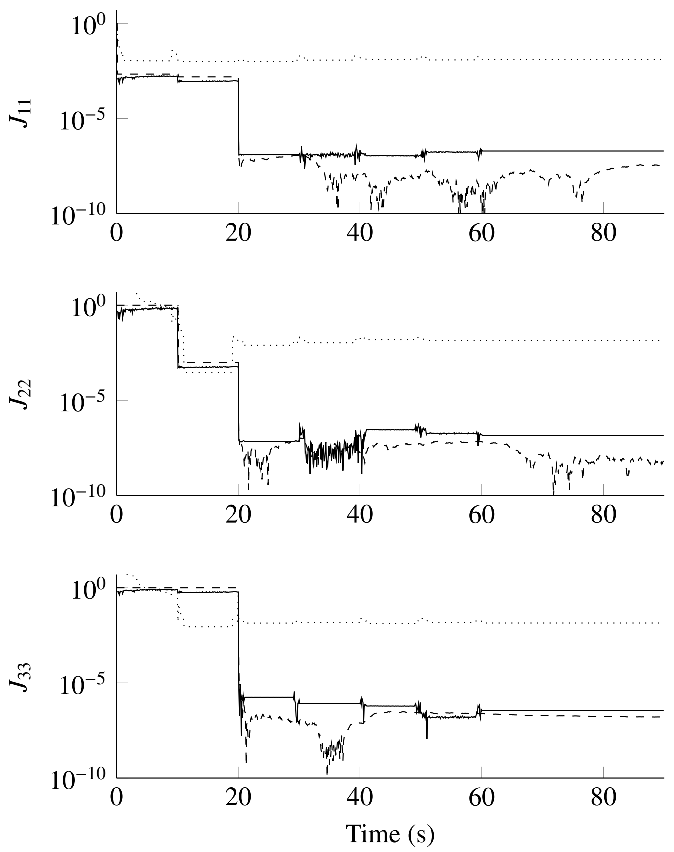

Figure 4. Normalized moment of inertia errors during slew maneuvers

torque is applied, both the variational SDP scheme and momentum-based scheme achieve similar performance.

VII. Conclusions

This study presents a recursive algorithm for spacecraft inertia estimation using semidefinite programming. A discrete mechanics formulation of the spacecraft dynamics allows data generated by standard attitude determination algorithms to be directly used as input to the estimator. By formulating the estimation problem as an SDP, the positive-definiteness and triangle-inequality constraints on the elements of the inertia tensor can be enforced, ensuring physically valid results. *A priori* information can also be incorporated into the estimator in the form of additional LMI constraints, enhancing convergence and leading to more accurate estimates when limited data is available. The availability of fast numerical solvers for SDPs makes the algorithm suited for both off-line analysis and real-time implementation.

References

[1] Bergmann, E. V., Walker, B. K., and Levy, D. R., "Mass property estimation for control of asymmetrical satellites," *Journal of Guidance, Control, and Dynamics*, Vol. 10, No. 5, 1987, pp. 483–491.

[2] Ahmed, J., Coppola, V. T., and Bernstein, D. S., "Adaptive asymptotic tracking of spacecraft attitude motion with inertia matrix identification," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 5, 1998, pp. 684–691.

[3] Psiaki, M. L., "Estimation of a spacecraft's attitude dynamics parameters by using flight data," *Journal of Guidance, Control, and Dynamics*, Vol. 28, No. 4, 2005, pp. 594–603.

[4] Tanygin, S. and Williams, T., "Mass property estimation using coasting maneuvers," *Journal of Guidance, Control, and Dynamics*, Vol. 20, No. 4, 1997, pp. 625–632.

[5] Norman, M. C., Peck, M. A., and O'Shaughnessy, D. J., "In-orbit estimation of inertia and momentum-actuator alignment parameters," *Journal of Guidance, Control, and Dynamics*, Vol. 34, No. 6, 2011, pp. 1798–1814.

[6] Keim, J. A., Behcet Acikmese, A., and Shields, J. F., "Spacecraft inertia estimation via constrained least squares," *Aerospace Conference, 2006 IEEE*, IEEE, 2006, pp. 6–pp.

[7] Peck, M. A., "Uncertainty models for physically realizable Inertia Dyadics," *The Journal of the Astronautical Sciences*, Vol. 54, No. 1, 2006, pp. 1–16.

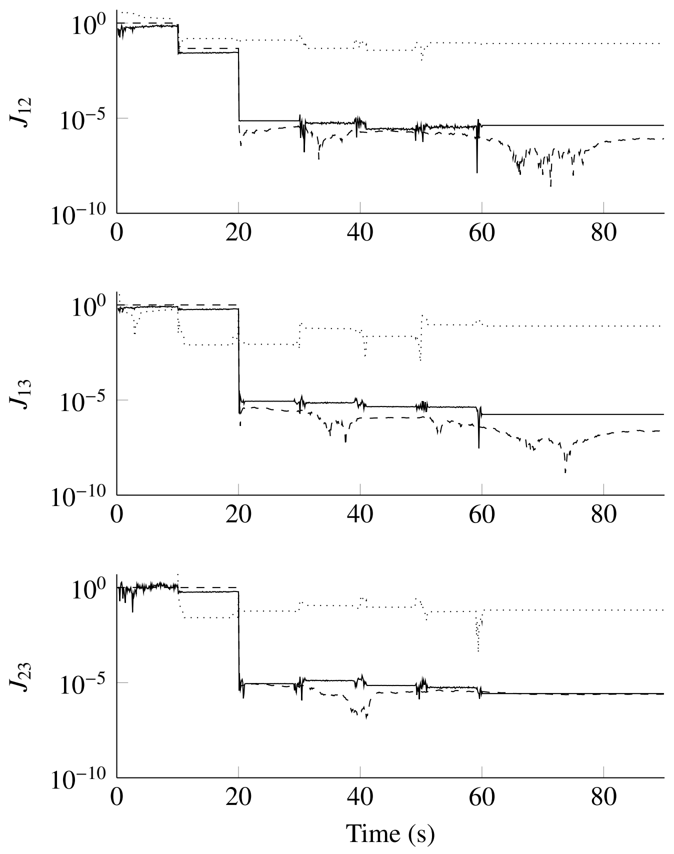

Figure 5. Normalized product of inertia errors during slew maneuvers

[8] Boyd, S. and Vandenberghe, L., *Convex Optimization*, Cambridge university press, 2009.

[9] Altmann, S., *Rotations, Quaternions, and Double Groups*, Oxford science publications, Clarendon Press, 1986.

[10] Boyd, S. P., El Ghaoui, L., Feron, E., and Balakrishnan, V., *Linear Matrix Inequalities in System and Control Theory*, Vol. 15, Society for Industrial and Applied Mathematics, 1994.

[11] Boyd, S. and Vandenberghe, L., "Semmidefinite Programming," *SIAM Review*, Vol. 38, No. 1, 1996, pp. 49–95.

[12] Nikravesh, P., Wehage, R., and Kwon, O., "Euler Parameters in Computational Kinematics and Dynamics. Part 1," *Journal of Mechanical Design*, Vol. 107, No. 3, 1985, pp. 358–365.

[13] Udwadia, F. E. and Schutte, A. D., "An Alternative Derivation of the Quaternion Equations of Motion for Rigid-Body Rotational Dynamics," *Journal of Applied Mechanics*, Vol. 77, No. 4, 2010, pp. 044505.

[14] Schaub, H. and Junkins, J. L., *Analytical Mechanics of Space Systems*, AIAA Education Series, Reston, VA, 2nd ed., October 2009.

[15] Hughes, P., *Spacecraft Attitude Dynamics*, Dover Books on Aeronautical Engineering, Dover Publications, Mineola, New York, 2004.

[16] Fosbury, A. M. and Nebelecky, C. K., "Spacecraft actuator alignment estimation," *Proceeding of the AIAA Guidance, Navigation and Control Conference (AIAA-2009-6316), Chicago, Illinois*, 2009, pp. 10–13.

[17] Marsden, J. E. and West, M., "Discrete mechanics and variational integrators," *Acta Numerica 2001*, Vol. 10, 5 2001, pp. 357–514.

[18] Manchester, Z. R. and Peck, M. A., "Quaternion Variational Integrators for Spacecraft Dynamics," *Journal of Guidance, Control, and Dynamics*, 2015.

[19] Goldstein, H., Poole, C., and Safko, J., *Classical Mechanics*, Addison Wesley, San Francisco, 3rd ed., 2001.

[20] Marsden, J. and Ratiu, T., *Introduction to Mechanics and Symmetry*, Texts in Applied Mathematics, Springer-Verlag, New York, 1994.

[21] Lefferts, E. J., Markley, F. L., and Shuster, M. D., "Kalman filtering for spacecraft attitude estimation," *Journal of Guidance, Control, and Dynamics*, Vol. 5, No. 5, 1982, pp. 417–429.

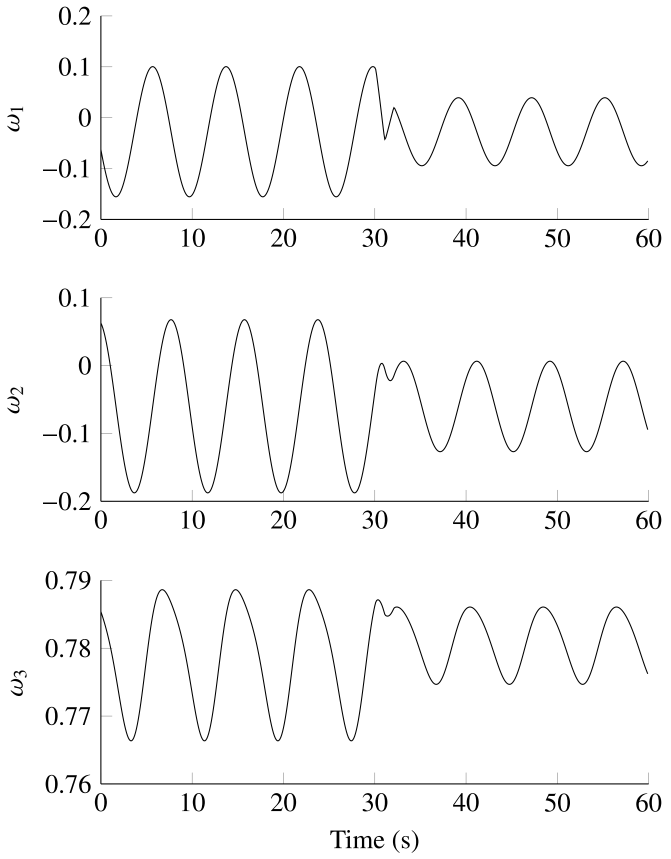

Figure 6. Body angular velocity components for spinning case

[22] Markley, F. L., "Attitude error representations for Kalman filtering," *Journal of guidance, control, and dynamics*, Vol. 26, No. 2, 2003, pp. 311–317.

[23] Trefethen, L. and Bau, D., *Numerical Linear Algebra*, Society for Industrial and Applied Mathematics, 1997.

[24] Golub, G. and Van Loan, C., *Matrix Computations*, Johns Hopkins University Press, 2012.

[25] Sturm, J. F., "Using SeDuMi 1.02, A Matlab toolbox for optimization over symmetric cones," *Optimization Methods and Software*, Vol. 11, No. 1-4, 1999, pp. 625–653.

[26] Mittelmann, H. D., "An independent benchmarking of SDP and SOCP solvers," *Mathematical Programming*, Vol. 95, No. 2, 2003, pp. 407–430.

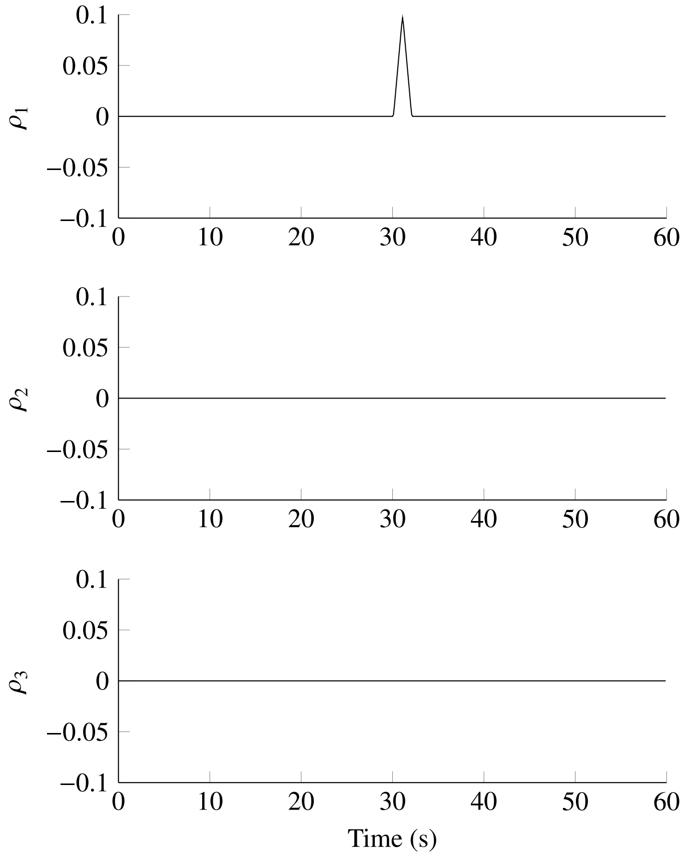

Figure 7. Rotor momentum components for spinning case

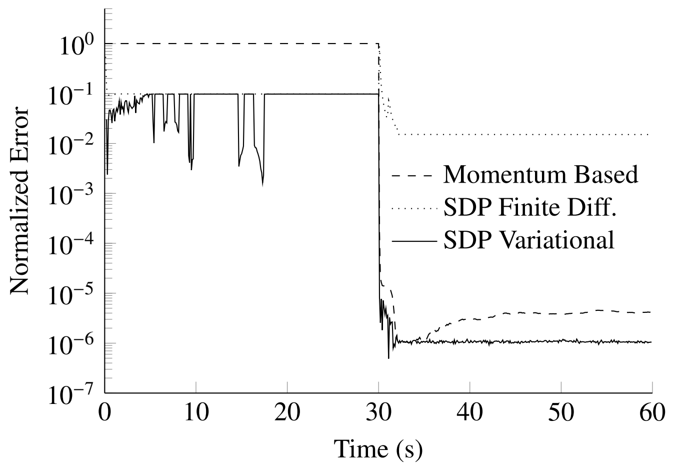

Figure 8. Total normalized error for spinning case

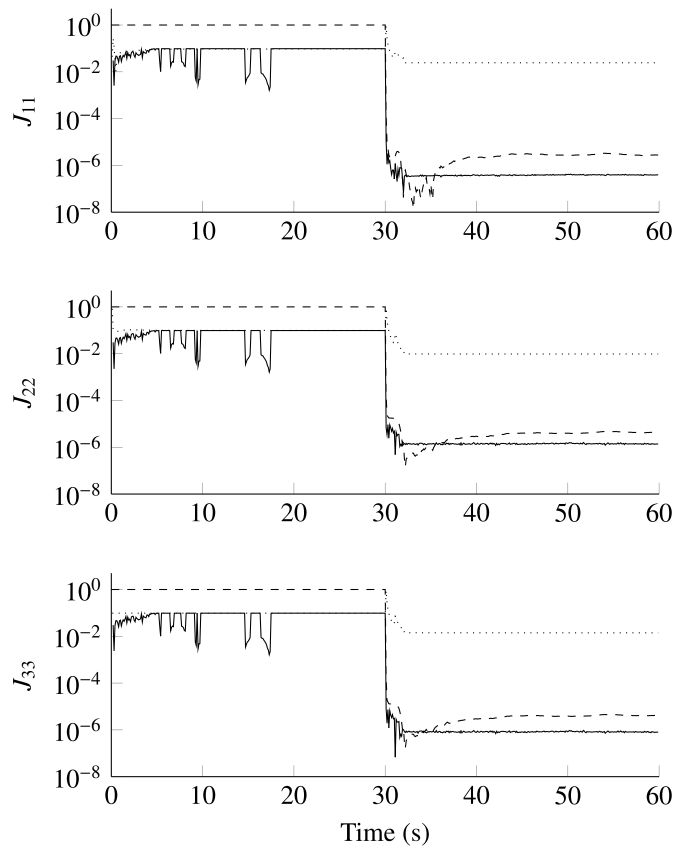

Figure 9. Normalized moment of inertia errors for spinning case

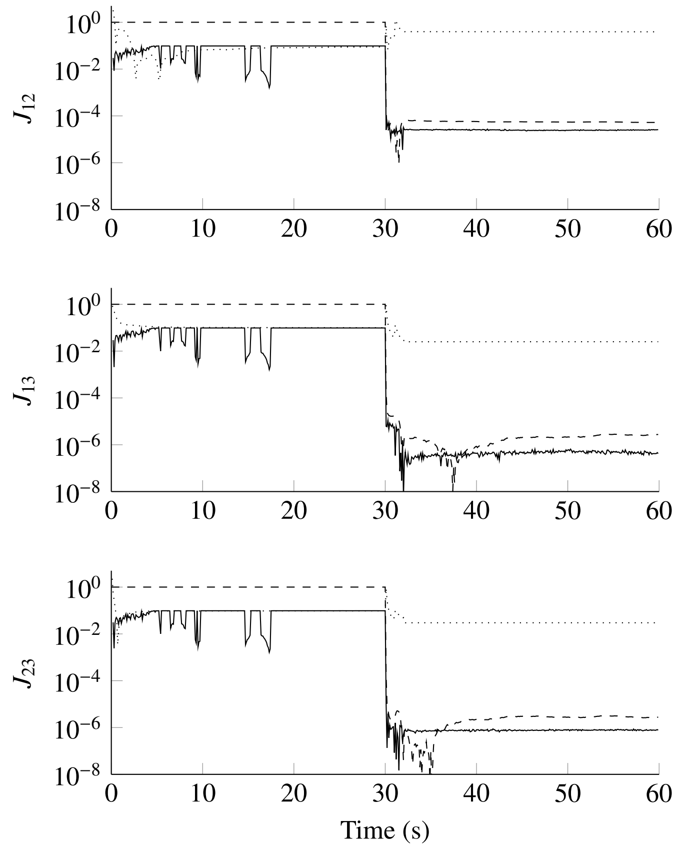

Figure 10. Normalized product of inertia errors for spinning case