Table 2 Results of numerical solutions

	Mars		Jupiter		Saturn	
Variable	0.0[a]	5.0[a]	0.0[a]	5.0[a]	0.0[a]	5.0[a]
r_{CO}, AU	1.034 (1.035)	1.026	1.176 (1.193)	1.146	1.214 (1.238)	1.180
v_{CO}, km/s	30.562	31.112	34.640	35.279	35.842	36.450
Δv_{circ}, km/s	4.581	4.069	7.333	6.951	6.521	6.286
t_B, day	11.521 (11.438)	10.014	26.163 (28.366)	23.614	28.810 (31.720)	26.189
t_T, yr	0.447	0.478	2.358	2.406	5.567	5.611

[a]α, deg.

estimate of each variable subject to constraints. For the first three cases, Earth–Mars, Earth–Jupiter, and Earth–Saturn, the goal of the NLP is to minimize the impulsive velocity change required to circularize the orbit at the various target planets by varying the burn time and choosing a constant thrust angle. For all cases, the thrust angle converged to the maximum allowed thrust angle of 5 deg. This is because the required velocity change to circularize the final orbit is reduced when any tangential acceleration is applied when compared to the pure radial thrust problem. Table 2 summarizes the results of the three example trajectories. The values in parentheses are the results of the analytical equations. The differences between the numerical and analytic results are due to mass flow being neglected in the analytical derivation. This effect is compounded for trajectories that reach farther out from the starting orbit.

In addition to these cases, a Pluto flyby mission was considered. In this case, the goal of the NLP was to minimize total flight time subject to a hyperbolic excess speed v_∞ of less than 20 km/s at Pluto. The first option considered was a direct transfer to Pluto starting from a circular orbit at 1 AU. This resulted in a cutoff radius of 1.428 AU after 40.461 days of burn and a total flight time of 5.953 years with a $v_\infty$of 20 km/s at Pluto. The second option considered was a burn to Venus, gravity assist at Venus, and then a burn to flyby Pluto. This was done at a close approach altitude of 300 km at Venus, and the resulting velocity vector was calculated using vector geometry and a total turning angle of 122 deg. This resulted in a burn time during the Earth–Venus transfer of 12.786 days with a cutoff radius of 1.043 AU and a flight time of 1.119 years. The second phase of the trajectory had a cutoff radius of 1.267 AU after 45.424 days of burn and an additional flight time of 6.087 years with a v_∞ of 20 km/s at Pluto. Even though the flight time and burn time are longer for the Venus–Pluto portion of the trajectory, the cutoff radius is smaller, which means the mission is more feasible in terms of using solar power to provide the power needed to operate M2P2.

Conclusions

Several outer planet missions were investigated using constant radial thrust propulsion and gravity assist trajectories. For the constant radial thrust missions, it was shown that a 5-deg thrust angle reduced the required impulsive velocity change to circularize the final orbit when compared to the pure radial thrust problem (0-deg thrust angle) and did so without a significant increase in burn time or total flight time. In addition, results indicate that a direct Earth–Pluto transfer was possible, but a Earth–Venus (flyby)–Pluto transfer produced a smaller cutoff radius than the direct transfer, thereby possibly suggesting a solar-powered propulsion system option. Further analysis using outer or multiple planet gravitational assists may provide a larger range of missions deemed relevant for radial thrust propulsion concepts.

Acknowledgment

This material is based on work supported under NASA Contract NAG8-1803.

References

[1]Winglee, R. M., Slough, J., Ziemba, T., and Goodson, A., "Mini-Magnetospheric Plasma Propulsion: Tapping the Energy of the Solar Wind for Spacecraft Propulsion," *Journal of Geophysical Research*, Vol. 105, No. A9, 2000, pp. 21,067–21,078.

[2]Boltz, F. W., "Orbital Motion Under Continuous Radial Thrust," *Journal of Guidance, Control, and Dynamics*, Vol. 14, No. 3, 1991, pp. 667–670.

[3]Prussing, J. E., and Coverstone-Carroll, V., "Constant Radial Thrust Acceleration Redux," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 3, 1998, pp. 516–518.

[4]Battin, R. H., *An Introduction to the Mathematics and Methods of Astrodynamics*, edited by J.S. Przemieniecki, AIAA Education Series, AIAA, New York, 1987, pp. 408–415.

[5]Betts, J. T., "Optimal Interplanetary Orbit Transfers by Direct Transcription," *Journal of the Astronautical Sciences*, Vol. 42, No. 3, 1994, pp. 247–268.

[6]Noca, M., Frisbee, R., Johnson, L., Kos, L., Gefert, L., and Dudzinski, L., "Evaluating Advanced Propulsion Systems for the Titan Explorer Mission," Electronic Rocket Propulsion Society, Paper IEPC-01-175, Oct. 2001.

Vehicle Motion Planning with Time-Varying Constraints

W. Todd Cerven,* Francesco Bullo,† and Victoria L. Coverstone‡
University of Illinois at Urbana–Champaign, Urbana, Illinois 61801

Introduction

WITH the growing emphasis on vehicle autonomy, the problem of planning a trajectory in an environment with obstacles has become increasingly important. This task has been of particular interest to roboticists and computer scientists, whose primary focus is on kinematic motion planning.[1] Typical kinematic planning methods fall into two main categories, roadmap methods and incremental search methods, both of which find collision-free paths in the state space. Roadmap methods generate and traverse a graph of collision-free connecting paths spanning the state space, whereas incremental search methods, including dynamic programming[2] and potential field methods,[3] perform an iterative search to connect the initial and goal states. For the purely geometric path-planning problem, deterministic algorithms have been created that are complete, that is, they will find a solution if and only if one exists. Unfortunately, these suffer from high computational costs, which are exponential in system degrees of freedom. This cost has motivated the development of iterative randomized path-planning algorithms that are probabilistically complete, that is, if a feasible path exists, the probability of finding a path from the initial to final conditions converges to one as the number of iterations goes to infinity. The introduction of the rapidly exploring random trees (RRTs) of LaValle and Kuffner[4] allowed both for computationally efficient exploration of a complicated space as well as incorporation of system dynamics.

Received 1 August 2003; revision received 19 January 2004; accepted for publication 20 January 2004. Copyright © 2004 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved. Copies of this paper may be made for personal or internal use, on condition that the copier pay the $10.00 per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923; include the code 0731-5090/04 $10.00 in correspondence with the CCC.

*Research Assistant, Coordinated Sciences Laboratory; currently Senior Member of Technical Staff, Performance Modeling and Analysis Department, The Aerospace Corporation, 15049 Conference Center Drive, Suite 1029, Chantilly, VA 20151. Member AIAA.

†Assistant Professor, Coordinated Sciences Laboratory, 1308 West Main Street.

‡Associate Professor, Department of Aerospace Engineering, 104 South Wright Street. Associate Fellow AIAA.

The RRT grows a tree of feasible trajectories from the initial condition, or root node. Each node, or waypoint, on the tree represents a system state and has possible trajectories branching from it. Through use of an embedded planning routine, the tree incrementally builds itself in random directions, node by node, until the final conditions are met (within accuracy bounds). Frazzoli et al.[5] demonstrated that a hybrid systems representation of vehicle dynamics, when coupled with the RRT, could be used to address moving obstacles and time-invariant final conditions in a real-time environment. This paper presents a similar approach that provides probabilistic completeness in the presence of both time-varying obstacles and final conditions while using a simpler algorithmic procedure. In addition, a novel approach to provide error mitigation of the embedded planner in a hybrid system-based RRT is presented. An example is then given in which the proposed algorithm is applied to the landing of a spacecraft on an idealized asteroid.

RRT-Based Approach

Background

The idea of this method is to incrementally build a tree of feasible trajectories to explore efficiently a reachable space, where a tree is a directed graph in which all nodes (excepting the root) have one parent node and an unspecified number of child nodes. The basic RRT algorithm[4] can be seen in Fig. 1a.

The original RRT algorithm, shown to be probabilistically complete, extended the tree by picking the closest (Euclidean metric ρ) node on the tree to the random point and choosing the best constant input from a finite predetermined set. For this simplistic embedded planner, the system equations of motion are propagated according to the input for a predetermined time. If no collisions are found, a new child node is added to the tree corresponding to the propagated state.

Frazzoli et al. adapted this method for control of autonomous vehicles where motion is represented by a concatenation of motion primitives.[5] They redefined the metric ρ as the cost-to-go function of an optimal control problem[2] and replaced the constant input set with an online planning algorithm that found solutions for the obstacle-free planning problem. Using the Extend routine in Fig. 1b, they were able to address moving obstacles by looping through successively close nodes on the tree until one was found that allowed an acceptable trajectory. If a collision-free trajectory was found, the routine would grow the tree and then try to connect to the final state. A completeness proof of this approach is in Ref. 5. A limitation of this method lies in its inability to address time-varying final conditions.

New Approach

We use a primitive-based hybrid system model that expands on that in Ref. 5, where the dynamics of the system, commonly described using an ordinary differential equation, are instead modeled by computed state flows in response to differing inputs. These can be separated into two types of primitives, reference trajectories and maneuvers, where reference trajectories are precomputed trajectories with a variable-time duration within a given interval and maneuvers are fixed-time primitives connecting reference trajectories. This is a hybrid system in that the state space is defined by the finite set of trajectory primitives, the continuous space over which each primitive can be applied, that is, where the dynamics of the system are invariant, and time. The control variables then consist of the reference trajectory time durations and the parameters defining each maneuver. Initial and final conditions are assumed to lie on reference trajectories. A more detailed description of this system is found in Ref. 6. Furthermore, we assume that there exists an embedded planner guaranteed to find motion planning solutions in an obstacle-free environment subject to time-varying final constraints and an upper time limit. The RRT-based algorithm for this approach follows the general loop in Fig. 1a, calling the Extend routine in Fig. 1c to try to connect the trajectories represented by the tree to a state defined by the function RandomState. Once again, the metric is defined as the cost-to-go function, but the state variable x now includes time as well. This addition both accommodates a time-dependent final condition and alleviates the need for cycling through the nodes in the tree to address time-varying obstacles. In this Extend routine, NearestNeighbor merely finds the closest node in the tree to x instead of sorting the tree nodes and cycling through them. InputFound then calls the embedded planner to find a feasible trajectory from the nearest node state to x and NoCollision incrementally checks the resulting trajectory for collisions. Provided a collision-free trajectory is found, the tree is extended by AddChildren. Unlike the approach of Frazzoli et al.,[5] this allows for the addition of single or multiple nested child nodes along the reference trajectories. Extend will repeat this process with the last created node until x is reached or a collision-free trajectory cannot be obtained. Once Extend exits, BuildRRT keeps looping until a solution is found. Probabilistic completeness for this algorithm can then be shown as follows.

Lemma: When it is assumed that no two RRT milestones lie within a specified $\epsilon > 0$ of one another for the given metric, this method is probabilistically complete.

Proof: When it is noted that the appropriate input is always generated by the online planning algorithm if it exists and is assigned a specific nonzero execution time, it follows from Theorem 3 of Ref. 4 that this method is probabilistically complete. □

```
BuildRRT($x_{start}$,$x_{final}$)
1: tree.Initialize($x_{start}$)
2: for iterations = 1 to maxiterations
3:     $x_{rand} \leftarrow$ RandomState()
4:     Extend(tree,$x_{rand}$,$x_{final}$)
5:     if $\rho(x_{rand}, x_{final}) < \epsilon$ break loop
6: return tree
```
a)

```
Extend(tree,$x$,$x_{final}$)
1: for all $x_{near}$ in SortedNodeList(tree,$x$)
2:     if InputFound($x_{near}, x, u$)
         and NoCollision($x_{near}, x, u$)
3:       tree.AddChild($x_{near}, x, u$)
4:       if $\rho(x, x_{final}) < \epsilon$ return success
5:       else let $x_{near} = x$, $x = x_{final}$ and goto 2:
6: return failure
```
b)

```
Extend(tree,$x$,$x_{final}$)
1: $x_{near} \leftarrow$ NearestNeighbor(tree,$x$)
2: if InputFound($x_{near}, x, u$)
     and NoCollision($x_{near}, x, u$)
3:   tree.AddChildren($x_{near}, x, u$)
4:   if $\rho(x, x_{final}) < \epsilon$ return connected
5:   else let $x_{near} = x$, $x = x_{final}$ and goto 2:
6: return not reachable or collision
```
c)

Fig. 1 Algorithm pseudocode, where ρ is a predefined metric: a) basic RRT loop, b) Frazzoli et al.[5] extend routine, and c) proposed extend routine.

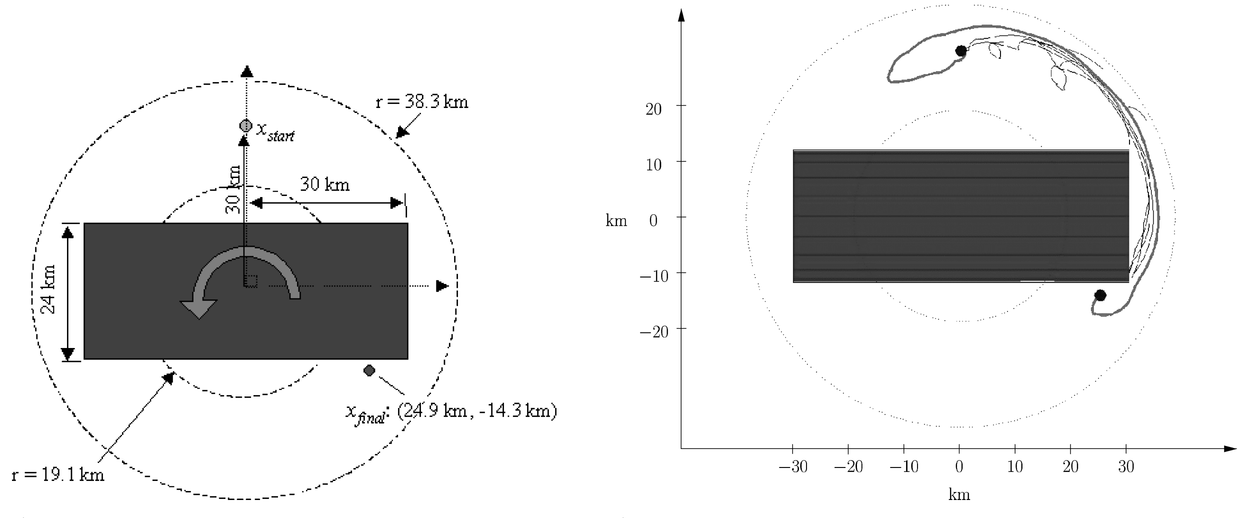

Fig. 2 Ida landing: a) setup and relative target location and b) tree and final path in rotating reference frame.

Embedded Planner Error Mitigation

The embedded planner naturally has a prescribed accuracy, and as a result, an error that can be propagated as trajectories are concatenated together. Although this effect could be troublesome, the framework of the RRT also allows for correction of errors from the underlying planning algorithm. For the initial incarnation of the RRT,[4] a constant input was chosen from a finite set and was highly unlikely to control the system to the intended final state. As a result, the actual final state (as found by integrating the system under the input) is that which is stored as the new node state rather than the targeted state. Thus, replanning from that node takes into account the error correction. When the hybrid system is addressed, the characteristics of these errors become important because replanning can only occur from nodes on reference trajectories. Thus, when integration is performed along the trajectory, nodes would only be added to the tree where the state matched the trajectory primitive within an acceptable error. Whereas this methodology is useful in practice, it is not a complete error correction because the correction is only the projection of the total error onto the trajectory primitive. Note that, although limitations of handling the error in this manner were not quantified, this method was shown to be consistently effective for the example problem.

Asteroid Landing Example

The example described here is that of a spacecraft landing on a celestial body similar to the asteroid Ida. We made simplifying assumptions by geometrically modeling the asteroid as a 60-km-long cylinder with radius 12 km and by modeling the gravity as a Newtonian point source. Although it is known that the gravity field about a nonspherical body is more complicated than the point source model, it is used here for simplicity. The algorithm could then be extended to other gravity models when needed. The setup of the problem is seen in Fig. 2, where the initial position of the spacecraft is 18 km above the surface of the asteroid with the final condition of landing at a point just off the surface on the other side of the asteroid. Although this problem could have been cast in a rotating reference frame, an inertial reference frame was chosen to show algorithm performance in relation to time-varying obstacles and final conditions. Additionally, artificial constraints are imposed to limit motion to Ida's plane of rotation and an annulus with radii between 19.1 and 38.3 km of the center of mass. An upper bound of 46 h was placed on the transfer time. The embedded planner used in the InputFound routine was based on a dynamic programming approach and may be found in Ref. 6. Note that, rather than sample the reachable set, which is computationally impractical to define, the sampling in the function RandomState was done over the free hybrid state space. Here, the reference trajectories in the hybrid system are defined as the set of all circular orbits, and the maneuvers are defined through an online local planning algorithm. From a start point at $[x^T, v^T]^T = [0, 30, 0, -0.0062, 0, 0]^T$ (km, km/s), the randomized algorithm grows a tree (Fig. 2), the result of 11 randomized planner iterations. The results of these iterations are given in Table 1.

Table 1 Randomized planner iterations: Extend results

Iteration	End condition	Greedy loops
0	Collision	0
1	Not Reachable	0
2	Not Reachable	0
3	Collision	0
4	Not Reachable	0
5	Not Reachable	0
6	Collision	1
7	Collision	1
8	Not Reachable	0
9	Not Reachable	0
10	Collision	1
11	Connected	2

The greedy loops column in Table 1 refers to the fact that the error correction mentioned earlier can cause a loop in the Extend routine. This occurred in step 11, where the Extend algorithm looped an extra time to reach the inertial final state. The overall solution was computed in 64.2 s, with a final state of $[-26.2, -12, 0, 0.0026, -0.0057, 0]^T$ (km, km/s) at 12.3 h. This is an error of 0.03% in position and 0.2% in velocity from the final condition.

Of course, this is a planner based on randomized methods, and as such, every solution to this algorithm will be slightly different with different run times. A batch of 50 runs of the aforementioned example were completed and yielded a median run time of 95.9 s (on an 850-MHz Pentium III computer) with 78% of the cases taking less than 200 s. The median number of randomized planner iterations was 11, with 90% taking less than 40 iterations. In every case tested, the algorithm successfully converged to a solution.

Conclusions

This paper presented a new variant of the RRT for use with a motion primitive-based planner. By the inclusion of time as a state, it is able to accommodate time-varying obstacles and final conditions. This method is shown to be probabilistically complete, finding a solution with a probability of one as the number of iterations goes to infinity. This method was then applied to the example of a spacecraft landing on an idealized asteroid, for which analysis of a batch of runs was completed. This method showed itself to be reliable with typical run times of less than 3 min. Although the randomized method shown is not optimal, there exist methods to refine the tree to increase optimality.

Acknowledgments

This research is supported in part by the U.S. Army Research Office under grant DAAD 190110716 and by National Science Foundation Grant IIS-0118146. The authors also gratefully acknowledge Emilio Frazzoli as well as Steven LaValle and his research group, whose comments were beneficial in the development of this paper.

References

[1]Latombe, J.-C., "Motion Planning: A Journey of Robots, Molecules, Digital Actors, and Other Artifacts," *International Journal of Robotics Research*, Vol. 18, No. 11, 1999, pp. 1119–1128.

[2]Bertsekas, D. P., *Dynamic Programming and Optimal Control*, 2nd ed., Vol. 1, Athena Scientific, Belmont, MA, 2001, Chap. 1, pp. 18–34.

[3]Sundar, S., and Shiller, Z., "Optimal Obstacle Avoidance Based on the Hamilton–Jacobi–Bellman Equation," *IEEE Transactions on Automatic Control*, Vol. 13, No. 2, 1997, pp. 305–310.

[4]LaValle, S. M., and Kuffner, J. J., "Randomized Kinodynamic Planning," *International Journal of Robotics Research*, Vol. 20, No. 5, 2001, pp. 378–400.

[5]Frazzoli, E., Daleh, M. A., and Feron, E., "Real-Time Motion Planning for Agile Autonomous Vehicles," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 1, 2002, pp. 116–129.

[6]Cerven, W. T., "Efficient Hierarchical Global Motion Planning for Autonomous Vehicles," Ph.D. Dissertation, Dept. of Aerospace Engineering, Univ. of Illinois at Urbana–Champaign, Urbana, IL, Oct. 2003.

Extended Kalman Filtering for Satellite Orbital Attitude Estimation Based on Gibbs Vector

YuRong Lin* and ZhengLong Deng†
Harbin Institute of Technology, 150001 Harbin, People's Republic of China

I. Introduction

THE stringent attitude pointing accuracy requirements of a nadir-pointing satellite demand real-time precise attitude determination from output information of the attitude measurement system. Current three-axis-stabilized satellite attitude measurement systems of the highest accuracy are typically composed of line-of-sight attitude sensors and gyros.[1–3] Conventional estimation techniques, such as the Kalman filter,[3–6] have been proven to be extremely useful for attitude estimation using vector measurements and gyro measurements. During the design of an attitude estimator, the parameters that describe the satellite's attitude should be selected first. The attitude of a three-axis-stabilized satellite is usually defined in the orbit-following coordinate system and called the orbital attitude.[1] The orbital attitude angles, which are small quantities when the satellite operates in a stable state, are suitable to be described by Euler angles.[6,7] By the Euler angle method only three parameters are needed; however, the direction cosine matrix evaluation based on the three angles as well as their dynamics equations involve a fair amount of trigonometric expressions, which can be time consuming, especially in real-time applications.

In the present study, a new optimal estimation algorithm is developed for determining the orbital attitude of a three-axis-stabilized satellite equipped with three-axis gyros as well as line-of-sight attitude sensors. The three-parameter Gibbs vector is used as the representation of the orbital attitude, whose kinematics equations are derived from the relation between the orbital attitude quaternion and the inertial attitude quaternion. Compared with the quaternion, the advantage of using the Gibbs vector results from the reduced number of parameters. In comparison to the Euler angles, the computation of the Gibbs vector does not involve trigonometric functions, but only simple algebraic expressions, and so the new algorithm possesses good real-time performance. Also, no singularity occurs when using the Gibbs vector to compute the orbital attitude because the orbital attitude angles are far less than 180 deg. Considering the measurement characteristics of the line-of-sight attitude sensor, the way to establish the attitude measurement equation is discussed separately for the case of one-vector observation and the case of multiple (more than two) vector observations. In the case of multiple-vector observations, simplification of the update process is achieved by combining the QUEST algorithm into the filter.

II. Sensor Models

The satellite being studied is assumed to be equipped with line-of-sight attitude sensors and three-axis gyros. The line-of-sight attitude sensor considered here is any sensor for which the measured quantity depends solely on the direction of some reference object in the sensor coordinate system.[3,8–10] The direction of a body in the sensor coordinate system $\boldsymbol{u}_0$ is related to the direction in the reference system $\boldsymbol{v}_0$ according to[9]

$$\boldsymbol{u}_0 = A\boldsymbol{v}_0 \tag{1}$$

where A is the attitude matrix. The actual measurements $\boldsymbol{u}$ and $\boldsymbol{v}$ of $\boldsymbol{u}_0$ and $\boldsymbol{v}_0$ are contaminated by additive measurement noise, respectively, such that

$$\boldsymbol{u} = \boldsymbol{u}_0 + \Delta\boldsymbol{u} \tag{2}$$

$$\boldsymbol{v} = \boldsymbol{v}_0 + \Delta\boldsymbol{v} \tag{3}$$

where $\Delta\boldsymbol{u}$ and $\Delta\boldsymbol{v}$ are assumed to be zero mean and white with covariance matrices $R_u = \sigma_u^2\boldsymbol{I}_{3\times3}(\boldsymbol{I}_{3\times3}$ is a 3×3 identity matrix) and $R_v = \sigma_v^2\boldsymbol{I}_{3\times3}$ (Ref. 11), respectively.

The gyro measures the satellite angular velocity $\boldsymbol{\omega}$. The gyro output $\boldsymbol{\omega}_g$ is related to $\boldsymbol{\omega}$ according to[1]

$$\boldsymbol{\omega}_g = \boldsymbol{\omega} + \boldsymbol{d} + \boldsymbol{b} + \boldsymbol{n}_g \tag{4}$$

where $\boldsymbol{d}$ is the correlative drift rate, $\boldsymbol{b}$ is the constant drift rate, and $\boldsymbol{n}_g$ is the gyro noise, and $\boldsymbol{d}$ and $\boldsymbol{b}$ are separately modeled as

$$\dot{\boldsymbol{d}} = -D_\tau\boldsymbol{d} + \boldsymbol{n}_d \tag{5}$$

$$\dot{\boldsymbol{b}} = 0_3 \tag{6}$$

where $\boldsymbol{n}_d$ is a zero-mean Gaussian white noise and D_τ is a diagonal matrix with inverse time constants as diagonal components.

III. Filter Design for Satellite Orbital Attitude Estimation

Attitude Kinematics

The Gibbs vector is a three-parameter attitude representation bearing direct relation to the attitude quaternion. The quaternion is defined as

$$\boldsymbol{q} = \begin{bmatrix} \boldsymbol{q}_{13} \\ q_4 \end{bmatrix} = [q_1 \quad q_2 \quad q_3 \quad q_4]^T \tag{7}$$

and the inverse of $\boldsymbol{q}$ is defined as $\boldsymbol{q}^{-1} = [-q_1 \ -q_2 \ -q_3 \ -q_4]^T$. Denoting the orbital attitude quaternion as $\boldsymbol{q}^o$, then the Gibbs vector describing the satellite orbital attitude is defined as

$$\boldsymbol{g} = \boldsymbol{q}_{13}^o \big/ q_4^o = [g_1 \quad g_2 \quad g_3]^T \tag{8}$$

The rotational angular velocity of the satellite body coordinate system relative to the orbit-following coordinate system is unknown,

Received 5 December 2002; revision received 14 August 2003; accepted for publication 25 November 2003. Copyright © 2004 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved. Copies of this paper may be made for personal or internal use, on condition that the copier pay the $10.00 per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923; include the code 0731-5090/04 $10.00 in correspondence with the CCC.

*Instructor, Teaching and Research Section 411, Department of Control Science and Engineering.

†Professor, Doctoral Tutor, Teaching and Research Section 411, Department of Control Science and Engineering.