JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 32, No. 4, July–August 2009

Engineering Notes

ENGINEERING NOTES are short manuscripts describing new developments or important results of a preliminary nature. These Notes should not exceed 2500 words (where a figure or table counts as 200 words). Following informal review by the Editors, they may be published within a few months of the date of receipt. Style requirements are the same as for regular contributions (see inside back cover).

Quaternion-Based Inverse Dynamics Model for Expressing Aerobatic Aircraft Trajectories

Rick G. Drury* and James F. Whidborne†
Cranfield University, Cranfield, Bedfordshire, England MK43 0AL, United Kingdom

DOI: 10.2514/1.42883

Nomenclature

$\mathbf{a}$	=	acceleration
D	=	drag
$\mathbf{e}$	=	quaternion $(e_0, e_1, e_2, e_3)^T$
$\mathbf{g}$	=	gravity vector $(0, 0, g)^T$
g	=	gravitational acceleration magnitude
$\mathbf{H}_E^W$	=	transformation matrix, Earth to wind frame
$\boldsymbol{l}_z$	=	normal load factor
l_z	=	normal load factor magnitude
M	=	aircraft mass
N	=	number of nodes per trajectory
p, q, r	=	roll, pitch, and yaw angular rates
$\mathbf{r}$	=	position vector $(x, y, z)^T$
s	=	path length
T	=	thrust
t	=	time
v	=	airspeed
$\mathbf{x}, \mathbf{y}, \mathbf{z}$	=	coordinate axes
x, y, z	=	position vector components
α	=	angle of attack
β	=	sideslip angle
γ	=	flight-path angle
δt	=	segment duration
$\hat{\boldsymbol{\epsilon}}$	=	unit vector of Frenet frame
μ	=	bank angle
χ	=	heading angle
Ω	=	4×4 quaternion propagation matrix
$\tilde{\omega}$	=	angular velocity cross product equivalent matrix
ω	=	wind frame angular velocity magnitude
$\|\cdot\|$	=	2-norm
$\times$	=	vector cross product
$\hat{}$	=	unit vector

Subscripts

E	=	inertial flat-Earth frame, north-east-down axes
j	=	node
t, n, b	=	Frenet frame tangential, normal, and binormal axes
V	=	velocity frame
W	=	wind frame

I. Introduction

THE aim of this Note is to present a computationally fast point-mass inverse dynamics model capable of representing all flight-path orientations without singularities and admitting smooth parameterization and interpolation for use in direct methods of trajectory generation for conventional fixed-wing aircraft.

Indirect methods of trajectory generation and optimization are not currently feasible for real-time implementation by onboard aircraft systems for the reasons given by Betts [1,2]. Direct methods, which rely on transforming the optimal control problem into a finite-dimensional nonlinear programming (NLP) problem, are not guaranteed to result in an optimal solution, but compared with indirect methods will usually generate a near-optimal solution more robustly (it is not necessary to solve potentially ill-conditioned combinations of state, adjoint, and stationarity/Pontryagin conditions) and have a larger convergence radius.

Hull [3] has classified methods for transforming optimal control problems into NLP problems by the set of parameterized variables. If only control and some of the state variables are parameterized, then numerical integration of the remaining state equations is required (e.g., by direct shooting or direct multiple shooting), but if all state variables are parameterized, then explicit numerical integration is not required (e.g., collocation, differential inclusion [4], or inverse dynamics). In general, optimal control problems for aircraft include nonlinear path inequality constraints and control constraints; Hull noted that path constraints could be incorporated into shooting and collocation methods but, in general, were only satisfied at discrete points in the trajectory and not between nodes. Hargraves and Paris [5] described a collocation method in which the defects were calculated at each node and at the center of each segment.

Lane and Stengel [6] and Sentoh and Bryson [7] described "inverse control" methods for designing flight control systems by expressing the control vector as a function of the trajectory to be tracked. Kato and Sugiura [8] described an "inverse dynamics" approach to trajectory generation using an aircraft model in which the state variables were calculated from the parameterized output trajectory by inverting the state equations. Lu [9] described a similar inverse dynamics approach using a point-mass model in polar coordinates applied to a low-Earth-orbit ascent trajectory. In all of the inverse dynamics methods, a dynamical model is required that balances fidelity against computational load. Simple, computationally cheap particle models of conventional fixed-wing flight dynamics have been well known since at least the 1960s; Menon [10] described a particle dynamical model for aircraft pursuit-evasion modeling using Cartesian coordinates, and Lou and Bryson [11] further investigated point-mass models for precision aerobatic maneuvers using (α, μ, T) as the control vector with spherical coordinates for velocity and horizontal cylindrical coordinates for position. These models all used Euler angles to represent path orientation.

For high-maneuverability applications such as air combat or aerobatics, the trajectory generator must process all path orientations without singularities or ill conditioning. Unit quaternions represent

Received 22 December 2008; revision received 17 March 2009; accepted for publication 22 March 2009. Copyright © 2009 by R.G. Drury. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. 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 0731-5090/09 $10.00 in correspondence with the CCC.

*Ph.D. Student, School of Engineering. Member AIAA.
†Senior Lecturer, School of Engineering. Member AIAA.

orientations without singularities and can be smoothly interpolated. In this work, an inverse dynamics model using unit quaternions is introduced.

II. Unit Quaternion-Based Model

The following assumptions are applied.

Assumption 1:

$$v > 0 \tag{1}$$

Assumption 2:

$$\hat{\mathbf{z}}_W = -\boldsymbol{l}_z/\|\boldsymbol{l}_z\|; \quad \|\boldsymbol{l}_z\| \neq 0 \tag{2}$$

Assumption 3:

$$\beta = 0 \tag{3}$$

Zero wind is also assumed.

A well-known Euler-angle system with state vector $(x, y, z, v, \gamma, \chi)^T$ and control vector $(a_x, \mu, l_z)^T$ may be defined as the set of state equations:

$$\begin{gathered}\dot{x} = v\cos\gamma\cos\chi \qquad \dot{y} = v\cos\gamma\sin\chi \qquad \dot{z} = -v\sin\gamma \\ \dot{v} = a_x \qquad \dot{\gamma} = \frac{(l_z\cos\mu - g\cos\gamma)}{v} \qquad \dot{\chi} = \frac{l_z\sin\mu}{v\cos\gamma}\end{gathered} \tag{4}$$

with controls given by

$$a_x = [(T-D)/M] - g\sin\gamma \tag{5}$$

$$\mu = \arctan\left(\frac{v\dot{\chi}\cos\gamma}{v\dot{\gamma} + g\cos\gamma}\right) \tag{6}$$

where arctan is defined as the four-quadrant inverse tangent, and

$$l_z = \|\boldsymbol{l}_z\| = ((v\dot{\gamma} + g\cos\gamma)^2 + (v\dot{\chi}\cos\gamma)^2)^{1/2} \tag{7}$$

which exhibits a singularity at $\gamma = \pm\pi/2$.

Although the effects of the singularity can be mitigated by suitable implementation, the issue can be overcome by using a differentially flat quaternion model that does not have the singularity and for which the computational load is comparable with that of the Euler-angle model.

A. Quaternion Model State Equations

The orthogonal transformation matrix between flat-Earth axes and wind axes is well known and can be expressed in both Euler-angle and quaternion forms [12,13]. Defining a state vector $(x, y, z, v, \mathbf{e}^T)^T$ and control vector $(a_x, p, q, r)^T$, equating corresponding elements of the two forms of the transformation matrix, and incorporating the quaternion kinematic equation $\dot{\mathbf{e}} = \Omega\mathbf{e}$ gives the following system of state equations:

$$\dot{x} = v(e_0^2 + e_1^2 - e_2^2 - e_3^2) \tag{8}$$

$$\dot{y} = 2v(e_1e_2 + e_0e_3) \tag{9}$$

$$\dot{z} = 2v(e_1e_3 - e_0e_2) \tag{10}$$

$$\dot{v} = a_x \tag{11}$$

$$\dot{\mathbf{e}} = \Omega\mathbf{e} \tag{12}$$

where

$$\Omega \triangleq \frac{1}{2}\begin{pmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{pmatrix} \tag{13}$$

B. Inverse Dynamics

Let x, y, and z be parameterized by functions that are at least 3 times continuously differentiable with respect to t and

$$\dot{s} \triangleq \|\dot{\mathbf{r}}\| = \sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2} \tag{14}$$

$$\ddot{s} \triangleq \tfrac{\mathrm{d}}{\mathrm{d}t}(\dot{s}) = \tfrac{1}{\dot{s}}(\dot{x}\,\ddot{x} + \dot{y}\,\ddot{y} + \dot{z}\,\ddot{z}) \tag{15}$$

where

$$a_x = [(T-D)/M] + (g\dot{z}/\dot{s}) \tag{16}$$

Using Frenet's formulas [14], $\hat{\boldsymbol{\epsilon}}_t$, $\hat{\boldsymbol{\epsilon}}_n$, and $\mathbf{a}_n$ are given by

$$\hat{\boldsymbol{\epsilon}}_t = \dot{\mathbf{r}}/\dot{s}, \qquad \hat{\boldsymbol{\epsilon}}_n \triangleq \rho\frac{\mathrm{d}\hat{\boldsymbol{\epsilon}}_t}{\mathrm{d}s} \tag{17}$$

$$\mathbf{a}_n = (\dot{s}^2/\rho)\hat{\boldsymbol{\epsilon}}_n = \ddot{\mathbf{r}} - \ddot{s}\,\dot{\mathbf{r}}/\dot{s} \tag{18}$$

The velocity frame V is the flat-Earth frame rotated through χ and γ only; hence,

$$\hat{\mathbf{x}}_V = \dot{\mathbf{r}}/\dot{s} \tag{19}$$

$$\hat{\mathbf{y}}_V = \left(\tfrac{-\dot{y}}{\sqrt{\dot{x}^2+\dot{y}^2}},\quad \tfrac{\dot{x}}{\sqrt{\dot{x}^2+\dot{y}^2}},\quad 0\right)^T \tag{20}$$

$$\hat{\mathbf{z}}_V = \hat{\mathbf{x}}_V \times \hat{\mathbf{y}}_V = \left(\tfrac{-\dot{x}\,\dot{z}}{\dot{s}\sqrt{\dot{x}^2+\dot{y}^2}},\quad \tfrac{-\dot{y}\,\dot{z}}{\dot{s}\sqrt{\dot{x}^2+\dot{y}^2}},\quad \tfrac{\dot{x}^2+\dot{y}^2}{\dot{s}\sqrt{\dot{x}^2+\dot{y}^2}}\right)^T \tag{21}$$

Clearly $\hat{\mathbf{x}}_W = \hat{\mathbf{x}}_V$. From Eq. (18) and because $\hat{\mathbf{x}}_W = \hat{\boldsymbol{\epsilon}}_t$ and $\hat{\mathbf{z}}_V \perp \hat{\mathbf{x}}_V$ and $\hat{\boldsymbol{\epsilon}}_n \perp \hat{\boldsymbol{\epsilon}}_t$, then $\hat{\mathbf{z}}_V \times \hat{\boldsymbol{\epsilon}}_n = k\hat{\mathbf{x}}_W$, where k is a nonzero scalar; thus, the plane containing $\hat{\mathbf{z}}_V$ and $\mathbf{a}_n$ is perpendicular to $\hat{\mathbf{x}}_W$, and the gravity vector in the plane is given by $(\mathbf{g}\cdot\hat{\mathbf{z}}_V)\hat{\mathbf{z}}_V$. From $\mathbf{g} = (0, 0, g)^T$ and Eq. (21),

$$(\mathbf{g}\cdot\hat{\mathbf{z}}_V)\hat{\mathbf{z}}_V = (g/\dot{s}^2)(-\dot{x}\,\dot{z},\quad -\dot{y}\,\dot{z},\quad \dot{x}^2 + \dot{y}^2)^T \tag{22}$$

Therefore,

$$\begin{aligned}\boldsymbol{l}_z &= \mathbf{a}_n - (\mathbf{g}\cdot\hat{\mathbf{z}}_V)\hat{\mathbf{z}}_V \\ &= \ddot{\mathbf{r}} - (\ddot{s}\,\dot{\mathbf{r}}/\dot{s}) - (g/\dot{s}^2)(-\dot{x}\,\dot{z},\quad -\dot{y}\,\dot{z},\quad \dot{x}^2 + \dot{y}^2)^T\end{aligned} \tag{23}$$

and, applying Assumption 2,

$$\hat{\mathbf{z}}_W = -\boldsymbol{l}_z/\|\boldsymbol{l}_z\| \tag{24}$$

$$\hat{\mathbf{y}}_W = \hat{\mathbf{z}}_W \times \hat{\mathbf{x}}_W \tag{25}$$

The relationship between the angular velocity of a frame and the transformation matrix is given [13,14] by the strap-down (Poisson kinematic) equation:

$$\tilde{\omega} = \mathbf{H}_E^W(\dot{\mathbf{H}}_E^W)^T \tag{26}$$

where $\mathbf{H}_E^W = (\hat{\mathbf{x}}_W,\ \hat{\mathbf{y}}_W,\ \hat{\mathbf{z}}_W)^T$. Equating elements on each side of Eq. (26) gives

$$p_W = -\hat{\mathbf{y}}_W \cdot \dot{\hat{\mathbf{z}}}_W \quad \text{or} \quad p_W = \hat{\mathbf{z}}_W \cdot \dot{\hat{\mathbf{y}}}_W \tag{27}$$

$$q_W = -\hat{\mathbf{z}}_W \cdot \dot{\hat{\mathbf{x}}}_W \quad \text{or} \quad q_W = \hat{\mathbf{x}}_W \cdot \dot{\hat{\mathbf{z}}}_W \tag{28}$$

$$r_W = \hat{\mathbf{y}}_W \cdot \dot{\hat{\mathbf{x}}}_W \quad \text{or} \quad r_W = -\hat{\mathbf{x}}_W \cdot \dot{\hat{\mathbf{y}}}_W \tag{29}$$

Note that the first expressions of Eqs. (27–29) avoid the need to calculate $\dot{\hat{\mathbf{y}}}_W$ and that p depends on $\dddot{\mathbf{r}}$ whereas a_x, q, r, and the Euler-angle model controls depend only on the first two derivatives of $\mathbf{r}$.

If $\mathbf{e}$ is required, then it can be derived from $\hat{\mathbf{x}}_W$, $\hat{\mathbf{y}}_W$, and $\hat{\mathbf{z}}_W$ without singularities by applying the algorithm described by Shoemake [15], which requires no more than one square root, five multiplications, one division, and six additions or subtractions.

III. Implementation and Example

When the model is discretized, the assumptions, path constraints, and controls can be evaluated from analytic instantaneous values at the nodes and from finite differences across the segments. Although analytic values are exact except for machine errors, it is shown next that segment-based values must also be considered.

A. Assumptions

Parameterization of $\mathbf{r}$ may invalidate Assumptions 1 and/or 2. Assumption 1 is required physically for aerodynamic lift and mathematically for excluding singularities in $\dot{\gamma}$, $\dot{\chi}$, $\hat{\mathbf{x}}_W$, and $\hat{\mathbf{z}}_W$. Provided that the parameterization ensures that $\dot{s}(t)$ has a finite number b of real roots, iterating $t_j \leftarrow t_j + \zeta\delta t$, where $-1 < \zeta < 1$ at any node j at which $\dot{s}(t_j) = 0$, until $\dot{s}(t_j) > 0$, ensures that $\dot{s}(t_j) > 0$ within at most b steps, thus avoiding the singularity at the node but not guaranteeing that $\dot{s}(t) > 0$ between nodes. Given $\dot{s}(t_j) > 0$, $\forall j \in (1, N)$, either applying the constraint $\dot{s}(t_j) + a_x(t_j)\delta t > 0$ or using $a_x(t_j) = (\dot{s}(t_{j+1}) - \dot{s}(t_j))/\delta t$ ensures that the assumption is satisfied between nodes. Note that finding the zeros of $\dot{s}(t)$ would be computationally expensive and that adding a small quantity to $\dot{s}(t_j)$ would introduce an unnecessary error.

For Assumption 2 ($l_z > 0$), Eq. (7) shows that $l_z = 0$ during "free fall" or linear vertical flight. Apart from the Euler-angle singularity, both the Euler-angle and quaternion models remain valid when $l_z = 0$, except that μ and $\hat{\mathbf{z}}_W$ become indeterminate. Hence, for both models, specifying that μ and $\hat{\mathbf{z}}_W$ do not change when $l_z = 0$ is sufficient.

B. Path Constraints and Controls

Evaluating path constraints based only on node values is insufficient to ensure feasibility. Consider a trajectory in which, at nodes $j-1$ and j, the rotational velocity and acceleration are zero but the path curves between nodes (e.g., a level course reversal occurring wholly between t_{j-1} and t_j). The load factor in the segment (l_z^{seg}) exceeds the values at the nodes. Therefore node values alone are insufficient to ensure feasibility. Interpolation between the nodes gives the following expression for l_z across each segment:

$$l_z^{\text{seg}} \approx \|0.5(v_j + v_{j-1})\omega^{\text{seg}}\hat{\boldsymbol{\epsilon}}_n^{\text{seg}} - (\mathbf{g}\cdot\hat{\mathbf{z}}_V)\hat{\mathbf{z}}_V\| \tag{30}$$

where

$$\omega^{\text{seg}} = \arccos(\dot{\hat{\mathbf{x}}}_{Wj} \cdot \dot{\hat{\mathbf{x}}}_{Wj-1})/\delta t$$

$$\hat{\boldsymbol{\epsilon}}_n^{\text{seg}} \approx (\dot{\hat{\mathbf{x}}}_{Wj} - \dot{\hat{\mathbf{x}}}_{Wj-1})/\|(\dot{\hat{\mathbf{x}}}_{Wj} - \dot{\hat{\mathbf{x}}}_{Wj-1})\|$$

Segment-based expressions for translational and torsional variables (e.g., T and p) and for Euler-angle model variables are similarly straightforward.

To generate a feasible and usable trajectory, a subset of the model state and control variables has to be transformed into the control vector of the particular flight control system, and the physical constraints on the particular aircraft need to be applied during optimization, usually including constraints on α, l_z, T, and p. Although an α constraint can often, in practice, be subsumed into an l_z constraint for nonstalled flight, α is required for transformation between the wind and body frames. Expressions for T, D, and α at many levels of fidelity are well known (e.g., [12,13]).

C. Computational Load

To assess computational performance, each model was applied to 19 different trajectories for which the parameterization function types were as follows: three linear, four vertical helices, two vertical loops, three horizontal helices, one quadratic polynomial, five fifth-degree polynomials, and one seventh-degree polynomial. To handle the singularity in the Euler-angle model, $\dot{\chi}_j \leftarrow 0$ if $\gamma_j = \pm\pi/2$. Each trajectory was discretized into N nodes and run in MATLAB® on a 32-bit PC for $N = 2^n + 1$, $n = \{17, 16, \ldots, 7\}$, first using the quaternion model then using the Euler-angle model, and the CPU times were recorded. The measurements were then repeated, but running the Euler-angle model first. For $N \leq 2^{13} + 1$, each trajectory was repeated $2^{14}/(N-1)$ times to reduce any effects of MATLAB® CPU timer quantization. The times included evaluating $\mathbf{r}$, its derivatives, and the inverse dynamics expressions, but excluded parameterization setup (common to both versions), constraint evaluations, and transforming controls to physical variables (specific to a particular aircraft/application). Controls were calculated using analytic derivatives at each node.

The Euler-angle inverse dynamics routine required, at worst, the following arithmetic operations at each node: 1 square root, 3 inverse trigonometric functions, 1 cosine, 22 multiplications, 3 divisions, and 15 additions/subtractions. For the quaternion model the corresponding operation count was as follows: 1 square root, 52 multiplications, 20 divisions, and 44 additions/subtractions. In each case, no functions were evaluated more than once; assignments and simple conditions were also performed.

Table 1 shows the mean, standard deviation, and range of the results, from which it can be seen that the mean quaternion model CPU time was comparable with that of the Euler-angle model.

If controls are calculated using across-segment finite differences instead of analytic derivatives at the nodes, reductions of 12 multiplications, 2 divisions, and 6 additions/subtractions for the Euler-angle model and 27 multiplications, 6 divisions, and 15 additions/subtractions for the quaternion model are possible. Figure 1 shows the percentage of 2160 arbitrary trajectories, each with time-varying angular and translational velocities, for which segment-based controls resulted in smaller worst-case and final position errors than node-based controls, plotted against δt. Each trajectory was parameterized by seventh-degree polynomials, and the state equations were numerically integrated using the MATLAB® ODE45 function.

Table 1 Ratio of quaternion model CPU time to Euler-angle model CPU time

N	Mean	Std. dev.	Max.	Min.
131073	0.42	0.01	0.45	0.38
65537	0.85	0.04	0.90	0.80
32769	0.95	0.02	1.00	0.91
16385	1.01	0.05	1.10	0.91
8193	1.01	0.05	1.15	0.93
4097	1.02	0.04	1.13	0.94
2049	1.03	0.06	1.14	0.94
1025	1.03	0.04	1.13	0.93
513	1.06	0.06	1.17	0.96
257	1.03	0.05	1.13	0.96
129	1.03	0.05	1.12	0.93
65	1.04	0.05	1.17	0.94
Overall	0.96	0.18	1.17	0.38

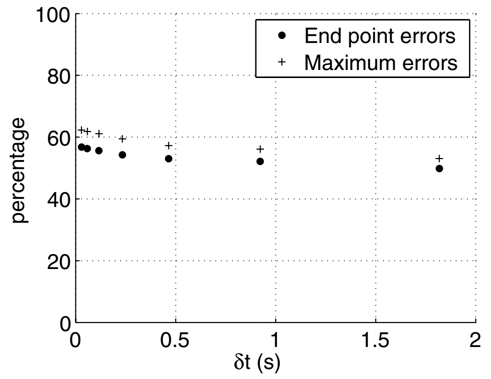

Fig. 1 Percentage of trajectories with segment-based errors less than node-based errors.

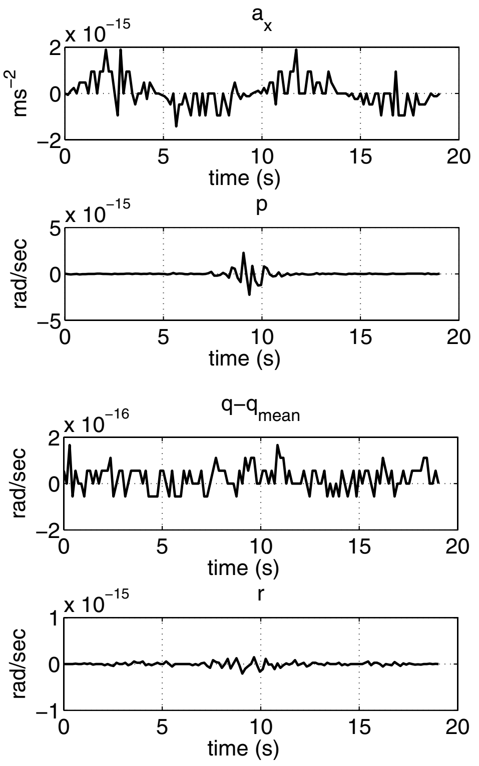

Fig. 2 Control vector for vertical loop, $a_n = 1$ g.

D. Trajectory Example

Figure 2 shows the control vector produced for a vertical loop with a constant normal acceleration of 1 g and a constant speed of 30 ms^{-1}, discretized into 129 nodes; a_x, p, and r should be zero and q should be constant. The error of the control vector $(a_x, p, q, r)^T$ computed using analytic derivatives is seen to be comparable to machine precision. Repeating the calculation with varying N from 131,073 to 17 (equivalent to $\delta t = 1.1$ s) confirmed that the error remained comparable to machine precision over this range, as expected because the analytic control expressions are independent of segment duration. Segment-based controls for a loop trajectory produce, because the curvature is constant, a constant q error that reduces as δt reduces; for $N = 129$ ($\delta t = 0.15$ s), the q error was $\sim 10^{-4}$ rad/s and the maximum position error was 0.2 m.

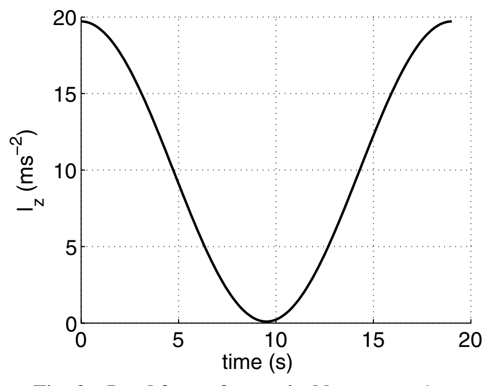

Fig. 3 Load factor for vertical loop, $a_n = 1$ g.

As shown in Fig. 3, the load factor varied from 0 to 2 g through the loop; the maximum difference between the segment-based load factor [Eq. (30)] and the node-based load factor [Eq. (23)] was 0.17 ms^{-2}, with a mean difference of 0.015 ms^{-2}.

IV. Conclusions

A point-mass inverse dynamics model based on unit quaternion path orientation representation, which does not have the inherent singularities of Euler-angle representation, is suitable for evaluating aircraft trajectories when the flight path includes all orientations. The model uses tangential acceleration and angular velocities as controls and has a linear orientation state equation. The computational speed performance of the model is comparable with that of a point-mass model using Euler-angle representation. When either model is discretized in an inverse dynamics direct method, an evaluation of assumptions and path constraints solely from analytic instantaneous values at the nodes is not sufficient to guarantee trajectory feasibility, and values derived from finite differences across the segments must be considered.

Acknowledgment

This work was supported by a U.K. Engineering and Physical Sciences Research Council Industrial Collaborative Awards in Science and Engineering award; the award was in collaboration with BAE Systems.

References

[1] Betts, J., "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 2, March 1998, pp. 193–207.
doi:10.2514/2.4231
[2] Betts, J., *Practical Methods for Optimal Control Using Nonlinear Programming*, Society for Industrial and Applied Mathematics, Philadelphia, 2001.
[3] Hull, D., "Conversion of Optimal Control Problems into Parameter Optimization Problems," *Journal of Guidance, Control, and Dynamics*, Vol. 20, No. 1, Jan. 1997, pp. 57–61.
doi:10.2514/2.4033
[4] Seywald, H., "Trajectory Optimization Based on Differential Inclusions," *Journal of Guidance, Control, and Dynamics*, Vol. 17, No. 3, 1994, pp. 480–487.
doi:10.2514/3.21224
[5] Hargraves, C., and Paris, S., "Direct Trajectory Optimization using Nonlinear Programming and Collocation," *Journal of Guidance, Control, and Dynamics*, Vol. 10, No. 4, July 1987, pp. 338–342.
doi:10.2514/3.20223
[6] Lane, S., and Stengel, R., "Flight Control Design Using Nonlinear Inverse Dynamics," *Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 4, July 1988, pp. 471–483.
doi:10.1016/0005-1098(88)90092-1
[7] Sentoh, E., and Bryson, A., "Inverse and Optimal Control for Desired Outputs," *Journal of Guidance, Control, and Dynamics*, Vol. 15, No. 3, 1992, pp. 687–691.
doi:10.2514/3.20892
[8] Kato, O., and Sugiura, I., "An Interpretation of Airplane General Motion and Control as Inverse Problem," *Journal of Guidance, Control, and Dynamics*, Vol. 9, No. 2, April 1986, pp. 198–204.
doi:10.2514/3.20090
[9] Lu, P., "Inverse Dynamics Approach to Trajectory Optimization for an Aerospace Plane," *Journal of Guidance, Control, and Dynamics*, Vol. 16, No. 4, 1993, pp. 726–732.
doi:10.2514/3.21073

[10] Menon, P., “Short-Range Nonlinear Feedback Strategies for Aircraft Pursuit-Evasion,” *Journal of Guidance, Control, and Dynamics*, Vol. 12, No. 1, Jan. 1989, pp. 27–32. doi:10.2514/3.20364

[11] Lou, K., and Bryson, A., “Inverse and Optimal Control for Precision Aerobatic Maneuvers,” *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 2, April 1996, pp. 483–488. doi:10.2514/3.21643

[12] Stengel, R., *Flight Dynamics*, Princeton Univ. Press, Princeton, NJ, 2004.

[13] Stevens, B., and Lewis, F., *Aircraft Control and Simulation*, 2nd ed., Wiley, Hoboken, NJ, 2003.

[14] Baruh, H., *Analytical Dynamics*, McGraw–Hill, Boston, MA, 1999.

[15] Shoemake, K., “Animating Rotation with Quaternion Curves,” *Computer Graphics*, Vol. 19, No. 3, July 1985, pp. 245– 254. doi:10.1145/325165.325242