AIAA Guidance, Navigation, and Control Conference
10 - 13 August 2009, Chicago, Illinois

AIAA 2009-6300

A Quaternion-based Inverse Dynamics Model for Real-time UAV Trajectory Generation

Rick G. Drury * and James F. Whidborne †

Cranfield University, Cranfield, Bedfordshire, MK43 0AL, England

This paper presents a computationally-fast inverse dynamics model based on unit quaternion orientation representation for use in the generation of unmanned aircraft trajectories when the flight envelope may include all flight path orientations, a necessary attribute for aerobatics or unmanned air-to-air combat vehicles. The model removes the inherent singularities in Euler-angle attitude representation, is as computationally efficient as an Euler-angle point mass model, permits smooth attitude parameterization and interpolation, and has a linear orientation state equation. Implementation details are described which show that the assumptions underlying both quaternion and Euler-angle models may be maintained, and path constraints evaluated, efficiently. Examples of trajectory optimization by discretizing the model in a direct method, using derivative-free optimization and 4-dimensional optimization vectors, show feasible control vectors for $\mathcal{O}(10^2)$ trajectory evaluations, making the method and model attractive for real-time trajectory generation.

Nomenclature

$\mathbf{a}$	=	acceleration
c_i	=	i-th inequality constraint
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)
$\mathbf{l}_z$	=	normal load factor: load factor component in wind $-z$ axis
l_z	=	normal load factor magnitude
M	=	aircraft mass
m	=	number of inequality constraints
N	=	number of nodes per trajectory
$\mathcal{P}$	=	penalty function
p, q, r	=	wind frame roll, pitch and yaw angular rates
$\mathbf{r}$	=	position vector $(x, y, z)^T$
s	=	path length
T	=	thrust
v	=	airspeed
x, y, z	=	position vector components
$\mathbf{x}, \mathbf{y}, \mathbf{z}$	=	coordinate axes
α	=	angle of attack; penalty function smoothing parameter
β	=	sideslip angle
γ	=	flight path angle
δs_j	=	Euclidean distance, node j–1 to node j
$\delta\tau$	=	segment virtual arc
δt_j	=	segment duration, node j–1 to node j

*PhD student, School of Engineering, Cranfield University, Cranfield, MK43 0AL, England and Member AIAA.
†Senior Lecturer, School of Engineering, Cranfield University, Cranfield, MK43 0AL, England, and Member AIAA.

$\hat{\epsilon}$ = unit vector of Frenet frame
η = maximum of constraint violations
κ = curvature
λ = speed factor
ρ = radius of curvature; penalty weighting parameter
τ = virtual arc
μ = bank angle
χ = track angle
Ω = 4x4 quaternion propagation matrix
$\tilde{\omega}$ = angular velocity cross product equivalent matrix
ω = wind frame angular velocity magnitude
$\|\cdot\|$ = 2-norm
$\times$ = vector cross product

Accent
$\hat{}$ = unit vector

Superscript or subscript
E = inertial flat-Earth frame, with north-east-down axes
j = node
seg = segment variable
t, n, b = Frenet frame tangential, normal and binormal axes
V = velocity frame
W = wind frame
$0, f$ = initial and final boundary points
$'$ = derivative with respect to τ

All quantities are expressed in the International System of Units.

I. Introduction

Unmanned air vehicles are in regular use for environmental, security, and meteorological data gathering, and for military surveillance and targeting. However, unmanned aerobatic vehicles, in particular air-to-air combat vehicles, remain at the research and development stages. The aim of this work is to present a computationally-fast inverse dynamics model capable of representing all flight path orientations without singularities, of admitting smooth orientation parameterization and interpolation, and which is applicable to direct methods of trajectory optimization.

Indirect methods of trajectory generation and optimization (using the calculus of variations and Pontryagin's Maximum Principle) are not currently feasible for real time implementation by on-board 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 by parameterization of a subset of the state and control variables and discretization of the trajectory, are not guaranteed to result in an optimal solution but compared to 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. Comparisons of direct and indirect methods can be found in Betts[1,2] and von Stryk and Bulirsch[3].

Hull[4] 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[5] 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 nodes in the trajectory and not between nodes. Hargraves and Paris[6] described a collocation method in which the defects were calculated at each node and at the center of each segment. Betts[1] described approaches to solving the system of differential-algebraic equations that arises when algebraic path constraints are included.

In the methods referenced above, the dynamical system differential state equations are explicitly or implicitly integrated using parameterized state and control vectors to generate defect and objective values for the NLP optimization. This may be considered a 'forward dynamics' approach. Lane and Stengel[7], and Sentoh and Bryson[8] 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[9] described an 'inverse dynamics' approach to trajectory generation in which the control vector was calculated from the parameterized output trajectory by inverting the state equations. Lu[10] 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 which balances fidelity against computational load; simple computationally-cheap particle models of conventional aircraft flight dynamics have been well known since at least the 1960's (e.g. Miele[11]); Menon[12] described a particle dynamical model for aircraft pursuit-evasion modeling using Cartesian coordinates; Lou and Bryson[13] further investigated point mass models for precision aerobatic maneuvers, using angle of attack, bank angle and thrust as the control vector with spherical coordinates for velocity and horizontal cylindrical coordinates for position. These models all used Euler-angle attitude representation. The inverse dynamics approach depends on the dynamical system being differentially flat (Fliess et al.[14,15]) which is a necessary condition for validity of the inverse dynamics model.

For high-maneuverability applications such as air combat or aerobatics a trajectory generator must process all orientations without singularities or ill-conditioning. In this work an inverse dynamics model using unit quaternion orientation representation is presented. Quaternions have the advantages of allowing smooth parameterization and interpolation[16–20] of attitude, a linear propagation equation, and of being easy to re-normalize during numerical integration. Stevens and Lewis[21] give a clear description of quaternion algebra and its application to aircraft navigation and dynamic simulation. In[22,23] Kaminer et al. described path following and coordinated control of multiple unmanned aircraft by trajectory following flight controllers using pitch and yaw rates as the control vector. Knoebel et al.[24] described a flight controller for an unmanned aircraft using quaternion feedback. The quaternion-based system model has the advantage over an Euler-angle model of producing angular rate and/or quaternion attitude control vectors suitable for controllers of the types described in References[22–24]; hence the quaternion model facilitates the imposition of path constraints on angular velocity which improves confidence in the feasibility of the generated trajectories.

In Section II the derivation of the inverse dynamics equations of the quaternion-based particle model is presented, together with a brief description of an Euler-angle model as a benchmark for computational load comparison. In Section III implementation issues arising when the model is discretized in an inverse dynamics method are discussed, the application of path constraints across segments as well as at nodes is shown to be necessary and readily implemented, and comparative results for the Euler-angle and quaternion models in representative trajectories are presented. In Section IV the application of the quaternion model in a particular direct method of trajectory optimization is described with example results. Section V highlights areas for future research and Section VI contains concluding remarks.

II. Models

In this work the following assumptions are applied

Assumption 1:

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

Assumption 2:

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

Assumption 3:

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

i.e. positive speed, body z axis aligned with the negative direction of the non-zero normal load factor, and zero sideslip. Zero wind is also assumed.

A. Euler-angle Model

Many variations of Euler-angle-based particle systems to model the flight dynamics of fixed-wing aircraft have been published (e.g.[9,10,12,13,25]). Using the aerospace convention of a yaw-pitch-roll sequence to

represent attitude by the set of Euler angles $(\chi, \gamma, \mu)^T$, 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

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

with controls given by

$$a_x = \frac{T-D}{M} - g\sin\gamma \tag{5}$$
$$\mu = \arctan\left(\frac{v\dot{\chi}\cos\gamma}{v\dot{\gamma} + g\cos\gamma}\right) \tag{6}$$
$$l_z = \|\mathbf{l}_z\| = \left((v\dot{\gamma} + g\cos\gamma)^2 + (v\dot{\chi}\cos\gamma)^2\right)^{1/2} \tag{7}$$

which exhibits a singularity at $\gamma = \pm\pi/2$. (Note arctan is defined as the 4-quadrant inverse tangent.)

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

B. Unit Quaternion-based Kinematic Model

In this subsection the state equations for the quaternion model introduced by the authors[26] are presented, the inverse dynamics analytic expressions for the control vector as a function of the output vector are derived, and the model is shown to be differentially flat.

1. 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[21, 25]. 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

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

2. Inverse Dynamics

Frenet's formulae, rotational transformations and the transport theorem lead to inverse dynamics analytic functions for orientation $\mathbf{e}$ and control vector $(a_x, p, q, r)^T$ as follows. (For clarity, functions of time t are abbreviated by omitting (t) in the rest of this section).

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

$$\dot{\mathbf{r}} \triangleq (\dot{x}, \dot{y}, \dot{z})^T \tag{14}$$
$$\dot{s} \triangleq \|\dot{\mathbf{r}}\| = \sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2} \tag{15}$$
$$\ddot{s} \triangleq \frac{d}{dt}(\dot{s}) = \frac{1}{\dot{s}}(\dot{x}\ddot{x} + \dot{y}\ddot{y} + \dot{z}\ddot{z}) \tag{16}$$

Tangential acceleration a_x is given by

$$a_x = \frac{T - D}{M} + \frac{g\dot{z}}{\dot{s}} \tag{17}$$

Using intrinsic curvilinear coordinates of the path and Frenet's formulae (following[27]), the tangential and normal unit vectors of the path $\hat{\epsilon}_t$, $\hat{\epsilon}_n$, and normal acceleration $\mathbf{a}_n$ are related to $\mathbf{r}$ and its time derivatives by

$$\hat{\epsilon}_t = \dot{\mathbf{r}}/\dot{s}\ ; \quad \hat{\epsilon}_n \triangleq \rho \frac{d\hat{\epsilon}_t}{ds}$$

differentiating and canceling $\hat{\epsilon}_t$

$$\begin{aligned}
\ddot{\mathbf{r}} &= \ddot{s}\hat{\epsilon}_t + \frac{\dot{s}^2}{\rho}\hat{\epsilon}_n \\
\hat{\epsilon}_n &= \frac{\rho}{\dot{s}^3}\left(\dot{s}\ddot{\mathbf{r}} - \ddot{s}\dot{\mathbf{r}}\right) \\
\mathbf{a}_n &= \frac{\dot{s}^2}{\rho}\hat{\epsilon}_n = \ddot{\mathbf{r}} - \ddot{s}\dot{\mathbf{r}}/\dot{s}
\end{aligned} \tag{18}$$

The velocity frame V is the flat-Earth frame rotated through yaw angle χ and flight path angle γ only, hence

$$\begin{aligned}
\hat{\mathbf{x}}_V &= \dot{\mathbf{r}}/\dot{s} \\
\hat{\mathbf{y}}_V &= \left(\frac{-\dot{y}}{\sqrt{\dot{x}^2 + \dot{y}^2}}, \quad \frac{\dot{x}}{\sqrt{\dot{x}^2 + \dot{y}^2}}, \quad 0\right)^T \\
\hat{\mathbf{z}}_V &= \hat{\mathbf{x}}_V \times \hat{\mathbf{y}}_V \\
&= \left(\frac{-\dot{x}\dot{z}}{\dot{s}\sqrt{\dot{x}^2 + \dot{y}^2}}, \quad \frac{-\dot{y}\dot{z}}{\dot{s}\sqrt{\dot{x}^2 + \dot{y}^2}}, \quad \frac{\dot{x}^2 + \dot{y}^2}{\dot{s}\sqrt{\dot{x}^2 + \dot{y}^2}}\right)^T
\end{aligned} \tag{19}$$

From Eq. (18) and $\hat{\mathbf{x}}_W = \hat{\epsilon}_t$ and $\hat{\mathbf{z}}_V \perp \hat{\mathbf{x}}_V$ and $\hat{\epsilon}_n \perp \hat{\epsilon}_t$ then

$$\hat{\mathbf{z}}_V \times \hat{\epsilon}_n = k\hat{\mathbf{x}}_W \qquad \text{where } k \text{ is a non-zero scalar}$$

and clearly

$$\hat{\mathbf{x}}_W = \hat{\mathbf{x}}_V \tag{20}$$

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. (19)

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

therefore

$$\begin{aligned}
\mathbf{l}_z &= \mathbf{a}_n - (\mathbf{g} \cdot \hat{\mathbf{z}}_V)\hat{\mathbf{z}}_V \\
&= \ddot{\mathbf{r}} - \frac{\ddot{s}\dot{\mathbf{r}}}{\dot{s}} - \frac{g}{\dot{s}^2}\left(-\dot{x}\dot{z}, \quad -\dot{y}\dot{z}, \quad \dot{x}^2 + \dot{y}^2\right)^T
\end{aligned} \tag{21}$$

and applying assumption 2

$$\hat{\mathbf{z}}_W = -\mathbf{l}_z/\|\mathbf{l}_z\| \tag{22}$$

$$\hat{\mathbf{y}}_W = \hat{\mathbf{z}}_W \times \hat{\mathbf{x}}_W \tag{23}$$

The relationship between the transformation matrix and the wind frame angular velocity vector is given by the strapdown (Poisson kinematic) equation

$$\tilde{\boldsymbol{\omega}} = \mathbf{H}_E^W (\dot{\mathbf{H}}_E^W)^T \tag{24}$$

which can be derived from the transport theorem, see for example[21,27], and where the angular velocity cross product equivalent matrix $\tilde{\omega}$ is

$$\tilde{\omega} = \begin{pmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{pmatrix} \tag{25}$$

and $\dot{\mathbf{H}}_E^W = ({}_E\dot{\hat{\mathbf{x}}}_W,\ {}_E\dot{\hat{\mathbf{y}}}_W,\ {}_E\dot{\hat{\mathbf{z}}}_W)^T$. Hence p, q, and r can be obtained by equating elements of Eqs. (24) and (25):

$$p_W = -\hat{\mathbf{y}}_W \cdot \dot{\hat{\mathbf{z}}}_W \qquad \text{or} \qquad p_W = \hat{\mathbf{z}}_W \cdot \dot{\hat{\mathbf{y}}}_W \tag{26}$$

$$q_W = -\hat{\mathbf{z}}_W \cdot \dot{\hat{\mathbf{x}}}_W \qquad \text{or} \qquad q_W = \hat{\mathbf{x}}_W \cdot \dot{\hat{\mathbf{z}}}_W \tag{27}$$

$$r_W = \hat{\mathbf{y}}_W \cdot \dot{\hat{\mathbf{x}}}_W \qquad \text{or} \qquad r_W = -\hat{\mathbf{x}}_W \cdot \dot{\hat{\mathbf{y}}}_W \tag{28}$$

Note that the first expressions of each of Eqs. (26), (27) and (28) avoid the need to calculate $\dot{\hat{\mathbf{y}}}_W$, and that p depends on $\ddot{\mathbf{r}}$ whilst a_x, q, r, and the Euler-angle model controls depend only on the first 2 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[16] which requires no more than 1 square root, 5 multiplications, 1 division and 6 additions or subtractions.

3. Differential Flatness

To confirm that the unit quaternion system Eqs. (8–12) is differentially flat, define an output vector $\mathbf{y} = (x, y, z)^T$. Eqs. (26), (27), and (28) show that p, q, and r are real analytic functions of $\hat{\mathbf{x}}_W$, $\hat{\mathbf{y}}_W$, and $\hat{\mathbf{z}}_W$; Eqs. (21), (22), and (23) express $\hat{\mathbf{x}}_W$, $\hat{\mathbf{y}}_W$, and $\hat{\mathbf{z}}_W$ as real analytic functions of $\mathbf{r}$ and its derivatives, and since $\mathbf{y} = \mathbf{r}$, p, q, and r are then real analytic functions of $\mathbf{y}$ and its derivatives.

To show that $\mathbf{e}$ can be expressed as a real analytic function of $\mathbf{y}$ and its derivatives, let $\mathbf{H}_E^W = [h_{ij}]$, $i = 1, 2, 3; j = 1, 2, 3$ and re-arrange to give the standard[21] result

$$e_0^2 = (1 + h_{11} + h_{22} + h_{33})/4 \qquad e_1^2 = (1 + h_{11} - h_{22} - h_{33})/4$$
$$e_2^2 = (1 - h_{11} + h_{22} - h_{33})/4 \qquad e_3^2 = (1 - h_{11} - h_{22} + h_{33})/4$$

$$e_0 e_1 = (h_{23} - h_{32})/4\,; \quad e_1 e_2 = (h_{12} + h_{21})/4$$
$$e_0 e_2 = (h_{31} - h_{13})/4\,; \quad e_2 e_3 = (h_{23} + h_{32})/4$$
$$e_0 e_3 = (h_{12} - h_{21})/4\,; \quad e_1 e_3 = (h_{13} + h_{31})/4$$

and applying the quaternion property that $e_0^2 + e_1^2 + e_2^2 + e_3^2 = 1$, $\mathbf{e}$ is a real analytic function of the elements of $\mathbf{H}_E^W$, hence of $\hat{\mathbf{x}}_W$, $\hat{\mathbf{y}}_W$, and $\hat{\mathbf{z}}_W$. Hence, subject to assumptions 1, 2, and 3, the system of Eqs. (8–12) is differentially flat.

III. Implementation

Validity of the model is dependent on maintaining the validity of the assumptions; it is also shown below that when the model is discretized in an inverse dynamics method it is necessary to consider path constraints across segments as well as at nodes. Methods to satisfy these requirements, and an expression to efficiently compute $\dot{\hat{\mathbf{z}}}_W$ (and hence p), are presented in this section. The resulting computational performance of the quaternion model is compared to that of the Euler-angle model, and examples are given of loop, horizontal helix, vertical helix and varying curvature spiral trajectories (Figures 2–9) to demonstrate the feasibility of the control vectors for these maneuvers.

A. Maintaining Assumptions

Assumption 1 is required physically for aerodynamic forces and moments and mathematically for excluding singularities. In most direct methods $v \equiv \dot{s}$ and Eq. (15) gives $\dot{s}(t)$ as a function of $\mathbf{r}$ such that $\dot{s}(t) \geq 0$. In general no closed-form analytic solution for the roots of $\dot{s}(t) = 0$ exists, hence root-finding is computationally expensive. However, $\dot{s}(t_j) = 0$ if and only if $\dot{x}(t_j) = 0$ and $\dot{y}(t_j) = 0$ and $\dot{z}(t_j) = 0$; if $\dot{x}, \dot{y}$, and $\dot{z}$

are parameterized by functions with known finite maximum numbers of real roots b_x, b_y, b_z (e.g. algebraic polynomials), then there can exist no more than $b = \min(b_x, b_y, b_z)$ zeros of $\dot{s}(t)$. Three possible actions to take at any such node (i.e. $\{j \,|\, \dot{s}(t_j) = 0\}$) are to: 1) skip the node entirely; or 2) add a small arbitrary quantity to $\dot{s}(t_j)$; or 3) move the node by iterating $t_j \leftarrow t_j \pm \zeta\delta t$ where $0 < \zeta < 1$ until $\dot{s}(t_j) > 0$. The first approach effectively doubles the segment length between the preceding and succeeding nodes which reduces the confidence level of trajectory feasibility. The second approach can only be implemented by adding an arbitrary quantity to $\dot{x}(t_j)$, $\dot{y}(t_j)$ and/or $\dot{z}(t_j)$ which introduces an unnecessary error and may make the inverse dynamics ill-conditioned (e.g. adding machine precision (eps) to $\dot{x}(t_j)$ only, compared to adding it to $\dot{y}(t_j)$ only, causes χ_j to change by $\pi/2$). The third approach adds no more than b nodes per trajectory to the computational load, and does not reduce confidence level or introduce an unnecessary error; this approach has been adopted in this work.

In the direct method described in Section IV speed v is parameterized independently of $\dot{\mathbf{r}}$, which admits the possibilities that

$$\exists j \in \{j \,|\, v(t_j) = 0, \dot{s}(t_j) > 0, j = 1, \ldots, N\} \tag{29}$$

$$\exists j \in \{j \,|\, v(_j) < 0, j = 1, \ldots, N\} \tag{30}$$

these are specific to that direct method and are addressed in Section IV.

For assumption 2 ($l_z > 0$), Eq. (7) shows that $l_z = 0$ only during 'free fall' or linear vertical flight. Apart from the Euler-angle singularity both 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

Evaluating path constraints at a node using only instantaneous values at that node ('node-based' evaluation) is insufficient to ensure feasibility: consider the normal load factor l_z in 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 maximum load factor in the segment (l_z^{seg}) exceeds the values at the nodes therefore node-based evaluation is insufficient to ensure feasibility. As noted in Section I path constraints can be incorporated into shooting and collocation methods but in general are only satisfied at discrete points in the trajectory and not across inter-node segments, although they apply for all $t \,|\, 0 \leq t \leq t_f$; in these methods constraint satisfaction may be improved by increasing the number of nodes[4], and in practice if the segment duration is sufficiently short an infeasible trajectory will violate control constraints. However, in the direct method of Section IV this is not necessarily true: this is addressed in Section IV A.

Although curvature $\kappa = 1/\rho$ is an obvious parameter for evaluating load factor, manipulating Eq. (18) gives[27]

$$\kappa = \frac{\sqrt{\dot{s}^2(\ddot{\mathbf{r}}.\ddot{\mathbf{r}}) - (\dot{\mathbf{r}}.\ddot{\mathbf{r}})^2}}{\dot{s}^3}$$

hence calculating $\max(\kappa(t) \,|\, t_0 \leq t \leq t_f)$ is computationally expensive. Shanmugavel et al. investigated Dubins sets extended to 3 dimensions[28,29], Pythagorean hodographs[30,31], and clothoid curves[32] as functions which facilitate the evaluation of curvature-based constraints. However, the inverse dynamics approach admits the evaluation of a normal load factor constraint across segments with low computational overhead: linear interpolation between the nodes gives the following expression for l_z across each segment

$$l_z^{seg} \approx \|0.5(v_j + v_{j-1})\omega^{seg}\hat{\epsilon}_n^{seg} - (\mathbf{g} \cdot \hat{\mathbf{z}}_V)\hat{\mathbf{z}}_V\| \tag{31}$$

where

$$\omega^{seg} = \arccos(\dot{\hat{\mathbf{x}}}_{Wj} \cdot \dot{\hat{\mathbf{x}}}_{Wj-1}) \,/\delta t$$
$$\hat{\epsilon}_n^{seg} \approx (\dot{\hat{\mathbf{x}}}_{Wj} - \dot{\hat{\mathbf{x}}}_{Wj-1})/\|(\dot{\hat{\mathbf{x}}}_{Wj} - \dot{\hat{\mathbf{x}}}_{Wj-1})\|$$

In principle it is necessary to evaluate other path constraints across each segment for both Euler-angle and quaternion models; segment-based expressions for translational and torsional variables (e.g. T and p) are similarly straightforward.

Figure 1 shows the difference between computed values of l_z and l_z^{seg} for the level course reversal trajectory described above. Other l_z^{seg} approximations are possible: using mean values for gravitational force; assuming a constant radius arc in the plane of $\mathbf{r}'_j$ and $\mathbf{r}'_{j-1}$ and computing the worst case load factor on that arc; and combinations of these approaches. The values of l_z^{seg} computed using these different approximations typically vary by less than 0.1 ms^{-2} for the trajectories of Figures 2–10; near l_{zmax} this variation is likely to be comparable with the tolerance on l_{zmax} and hence acceptable.

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 in practice be subsumed into an l_z constraint for non-stalled flight, α is required for transforming between wind and body frames. Expressions for T, D, and α at many levels of fidelity are well-known (e.g.[21,25]).

C. Computation of $\hat{\mathbf{z}}_W$

$\dot{\hat{\mathbf{x}}}_W$, $\dot{\hat{\mathbf{y}}}_W$, and $\dot{\hat{\mathbf{z}}}_W$ can be obtained by differentiating Eqs. (20), (23), and (22) but for $\dot{\hat{\mathbf{y}}}_W$ and $\dot{\hat{\mathbf{z}}}_W$ the equations, which include $\ddot{\mathbf{r}}$ and $\ddot{s}$, become computationally expensive to implement; $\dot{\hat{\mathbf{y}}}_W$ can be avoided by using the first expressions in Eqs. (26), (27) and (28) and $\dot{\hat{\mathbf{z}}}_W$ may be evaluated as follows. Define $\mathbf{l}$ as a vector and $\mathbf{z}$ as the unit vector parallel to $\mathbf{l}$, $\mathbf{dl}$ to be the total differential of $\mathbf{l}$, $\mathbf{dl_t}$ to be the component of $\mathbf{dl}$ parallel to $\mathbf{l}$, $\mathbf{dz}$ to be the total differential of $\mathbf{z}$, $\mathbf{dz}_t$ to be the component of $\mathbf{dz}$ parallel to $\mathbf{z}$, and $\mathbf{dz}_n$ to be the component of $\mathbf{dz}$ perpendicular to $\mathbf{z}$. Then

$$\mathbf{dz} = \frac{\mathbf{dl}}{\|\mathbf{l}\|}\,; \quad \mathbf{dl}_t = \frac{\mathbf{dl}.\mathbf{l}}{\|\mathbf{l}\|}\hat{\mathbf{z}} = (\mathbf{dl}.\hat{\mathbf{z}})\hat{\mathbf{z}}$$
$$\mathbf{dz}_t = \frac{\|\mathbf{dl}_t\|}{\|\mathbf{l}\|} = \frac{(\mathbf{dl}.\hat{\mathbf{z}})\hat{\mathbf{z}}}{\|\mathbf{l}\|}$$
$$\mathbf{dz}_n = \mathbf{dz} - \mathbf{dz}_t = \frac{1}{\|\mathbf{l}\|}\left(\mathbf{dl} - (\mathbf{dl}.\hat{\mathbf{z}})\hat{\mathbf{z}}\right)$$

Therefore, substituting $\mathbf{l}_z = \mathbf{l}$ and $\hat{\mathbf{z}}_W = -\mathbf{z}$,

$$\dot{\hat{\mathbf{z}}}_W = -\frac{1}{\|\mathbf{l}_z\|}\left(\dot{\mathbf{l}}_z - (\dot{\mathbf{l}}_z.\hat{\mathbf{z}}_W)\hat{\mathbf{z}}_W\right) \tag{32}$$

where $\dot{\mathbf{l}}_z$ is obtained by differentiating Eq. (21).

D. 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. Each trajectory was run in Matlab$^{\text{TM}}$ on a 32-bit PC using the quaternion model then using the Euler-angle model, for $N = 2^n+1, n = \{17, 16, ...7\}$, and the ratio of CPU time consumed by the quaternion model to CPU time consumed by the Euler-angle model was 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 from the Matlab$^{\text{TM}}$ CPU timer quantization. The times include the processing necessary to extract $\mathbf{r}$ and its derivatives from the parameterization functions at each node, but exclude trajectory parameterization setup, constraint evaluations, and transforming controls to physical variables. Controls were calculated using analytic derivatives at each node. As tested, the Euler-angle inverse dynamics routine performed, at each node: 1 square root, 3 inverse trigonometric functions, 1 cosine, 22 multiplications, 3 divisions and 15 additions/subtractions (no functions were evaluated more than once) compared to: 1 square root, 52 multiplications, 20 divisions and 44 additions/subtractions for the quaternion inverse dynamics routine; in both cases assignments and simple conditions were also performed. CPU times depend not only on the model but on the efficiency with which the compiler and run-time environment implement the square root and trigonometric functions and on memory management; for the Matlab$^{\text{TM}}$ environment 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.

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

E. Example Results

Figures 2 and 3 show a loop trajectory and the control vector produced by the quaternion model using Matlab^{TM} on a 32-bit PC (machine precision 2.22E–16). The trajectory was defined as a vertical loop through 360° with constant normal acceleration and constant speed, starting and ending in level flight tracking 45°, and discretized into 129 nodes. The control values are clearly feasible.

Since the evaluation of the control vector is analytic, errors in the control vector should arise only from discretization of the trajectory or machine floating point round-off error. Further, since the control expressions of Section II are functions of values at discrete nodes, and therefore are not dependent on the interval between nodes, discretization error should not be significant. For the loop trajectory of Figure 2 a_x, p and r should be zero and q should be constant. Figure 3 shows the error deviation of the computed values of the control vector $(a_x, p, q, r)^T$ to be comparable to machine precision. Repeating the calculation with varying N from $2^{17}+1$ to 2^4+1 (equivalent to a node interval of 1.1 seconds) confirmed that the error variation is not dependent on N and is comparable to machine precision over this range. However, since the control vector is held constant between nodes, decreasing N will affect the ability of a flight control system to track the desired trajectory thus imposing a practical lower limit on N for a given combination of aircraft and flight control system. Segment-based controls for a loop trajectory produce, since the curvature is constant, a constant q error which reduces as δt reduces; for $N = 129$, ($\delta t = 0.15$ secs), the q error was $\sim 10^{-4}$ rad/sec and the maximum position error was 0.2 meters. Similar results have been obtained for horizontal helix, vertical helix, and varying curvature climbing/descending turns. Figures 4–10 show various trajectories and computed control vectors - all the control vectors are feasible and the error in the computed values is comparable to machine precision.

IV. Application to a Direct Method

The objective of this Section is to demonstrate the quaternion inverse dynamics model in a particular direct method for which published performance results[33,34] suggest suitability for real-time aircraft trajectory generation. Two examples are given that take different approaches to parameterizing the output vector.

A. Direct Method

A direct method that transforms the optimal control problem into a nonlinear programming problem of low dimension (typically < 10) potentially suitable for real-time trajectory generation was described in Refs[22,23,33–36]. Taranenko and Momdzhi[35] (as described in[33]) introduced variables τ (monotonically increasing 'virtual arc') and λ ('speed factor'). Position $\mathbf{r}$ was parameterized with respect to τ and λ was

defined as

$$\lambda \triangleq d\tau/dt \tag{33}$$

Then as described by Kaminer et al.[23]

$$v = \|d\mathbf{r}/dt\| = \lambda\|d\mathbf{r}/d\tau\| \tag{34}$$
$$\lambda = \|\dot{\mathbf{r}}\|/\|\mathbf{r}'\|$$

Since λ can be varied independently of $\mathbf{r}$

$$\dot{\mathbf{r}} \neq f(\mathbf{r}')$$

thus decoupling speed from the spatial parameterization.

Taranenko and Momdzhi applied polynomial and trigonometric parameterization of the state variables, collocation, and inverse dynamics to generate near-optimal aircraft trajectories. In References[22,23,33,36] Yakimenko and colleagues showed that spatial parameterization by seventh degree algebraic polynomials provides a large set of trajectories that are guaranteed to meet state, velocity and acceleration boundary conditions, and allows the third derivatives of the position vector at the initial and final boundary points ($\mathbf{r}'''_{0,f}$) as a vector of free optimization variables. Although computation of the coefficients of high-degree interpolating polynomials is known to be subject to ill-conditioning (e.g. Press[37]), the referenced literature describes good results with this approach.

Speed may be parameterized explicitly, implicity by parameterizing λ, or by parameterizing a control (e.g. a_x) and integrating an equation of motion. Yakimenko[33] parameterized one control (thrust) and integrated one equation of motion; Kaminer et al.[22,23] applied the method to generate trajectories for multiple unmanned aircraft and described the option of parameterizing v explicitly, or implicitly via λ, thus parameterizing all states and obviating the need for integration of any equations of motion. Since v is a physical variable and subject to path constraints (e.g. nominal stall and 'never exceed' speeds), in this section v is parameterized explicitly.

Given the spatial and speed parameterizations with respect to τ, for an arbitrary scalar or vector variable ζ time derivatives are related to derivatives with respect to τ by the definition of λ, Eq. (33), giving

$$\begin{aligned}
&\dot{\zeta} = \lambda\zeta' \,; \quad \ddot{\zeta} = \lambda(\zeta''\lambda + \zeta'\lambda') \\
&\dddot{\zeta} = \lambda^3\zeta''' + 3\lambda^2\lambda'\zeta'' + (\lambda^2\lambda'' + \lambda\lambda'^2)\zeta'
\end{aligned} \tag{35}$$

To express λ as a function of the parameterizations of $\mathbf{r}'$ and v for substitution in Eqs. (35), several expressions are possible; Yakimenko[33] used

$$\lambda_j = \frac{(v(\tau_j) + v(\tau_{j-1}))/2}{\delta s_j}\ \delta\tau \tag{36}$$

where

$$\delta s_j = \sqrt{((x_j - x_{j-1})^2 + (y_j - y_{j-1})^2 + (z_j - z_{j-1})^2)}$$

An alternative expression using only node variables is

$$\lambda_j = v(\tau_j)/s'(\tau_j) \tag{37}$$

Eq. (36) has a singularity at $\delta s_j = 0$ and Eq. (37) has a singularity at $s'(\tau_j) = 0$ hence an approach to handling these singularities is required. Section III showed that adjusting t_j is a practical engineering approach to handling $\dot{s}(t_j) = 0$; this can be applied to either of $\delta s_j = 0$ or $s'(\tau_j) = 0$ in this method. Since δs_j is dependent on the segment duration, the associated discretization error would vary with N whereas $s'(\tau_j)$ is not segment dependent. Hence in this work Eq. (37) with the approach of adjusting τ_j when $s'(\tau_j) = 0$ is used.

Since $\delta t_j = \delta\tau/\lambda$ singularities in δt_j arise when $\lambda = 0$, i.e. when $v(\tau_j) + v(\tau_{j-1}) = 0$ for Eq. (36) or when $v(\tau_j) = 0$ for Eq. (37). A trajectory containing $v \leq 0$ is infeasible since assumption (1) is invalidated; however, a candidate trajectory demanding $v \leq 0$ may arise from the speed parameterization $v(\tau)$ independently of the value of $s'(\tau_j)$ or δs_j (Eqs. (29) and (30)). Two possible approaches to detect and handle this condition are:

1. test for roots of the parameterization $v(\tau)$ in the range $\tau_0 \leq \tau \leq \tau_f$; or

2. test $v(\tau_j)$ as each node is evaluated and if $v(\tau_j) \leq 0$ set $v = \xi$ where ξ is a small positive arbitrary scalar such that $0 < \xi \ll v_{stall}$ and calculate a stall speed constraint violation (using the parameterized value of v).

The first approach requires the roots of the speed parameterization function; if no closed-form analytic solution for the roots of $v(\tau)$ exists a computationally expensive iterative approach is required. The second approach requires little additional computational load (1 condition and 1 assignment) since a stall speed constraint is required for any conventional fixed-wing aircraft. The second approach can also be applied to $v(\tau_j) + v(\tau_{j-1}) = 0$. In this work the second approach to handle $v(\tau_j) \leq 0$ is used.

Specific to this particular method, in the course reversal example of Section III v can be set constant. In this event node-based evaluations of a_x, p, q, r, μ, and l_z will satisfy node-based constraints at t_j and t_{j+1} (bracketing the turn) as $\delta t_j \to 0$ even though the trajectory is infeasible. However, the infeasibility can be detected by evaluating l_z^{seg}, (or $\delta\chi$ in this example).

B. Optimization and Penalty Functions

A robust optimization method is required for real-time on-board aircraft trajectory generation. A constrained optimization method, the active set method that is implemented in the Matlab™ *fmincon* command for optimization with nonlinear inequality constraints, was applied but consistent results were not obtained. (A similar result was reported by Bevilacqua et al.[38].)

Direct search 'derivative-free' unconstrained optimization methods are characterized by not requiring approximations of the gradients or Hessians of the objective or constraints. Although derivative-free methods have been published since at least the 1950's, until recently there were few published mathematical examinations of their convergence; in the last decade however, significant research has been carried out into their convergence properties (e.g.[39–42]). In this work a provably-convergent variant[42] introduced by Price et al. of the Nelder–Mead algorithm[43] was used. Sequential application of an unconstrained optimization method to a nonlinearly constrained problem by incorporating penalty functions is well documented (e.g.[44,45]) and in this work the sequential implementation follows Griffin and Kolda[46]: the maximum allowable constraint violation η_{max} and convergence tolerances are specified initially then at each major iteration in the sequence an optimization of $t_f + \mathcal{P}$ is performed using penalty parameter ρ, smoothing parameter α and convergence tolerances; if the optimization passes the convergence tests and $\eta < \eta_{max}$ the sequence ends; otherwise ρ is increased, α is reduced, convergence tolerances are reduced and the sequence re-runs the optimizer.

A minimum-time objective was used in all optimizations and normal load factor constraints were evaluated at nodes and across segments.

The deviation of the generated trajectories from optimality is dependent, inter alia, on the nonlinear optimization algorithm and the penalty function. Since the aim was to demonstrate the quaternion model rather than any particular optimization algorithm, the effectiveness of the optimization algorithm as a function of the initial guess, and the deviation from optimality of the final trajectory, were not measured.

C. Optimization Results

For these examples aircraft data representative of a small unmanned aircraft were used: mass 11 kg, maximum thrust 20 N, $15 \leq v \leq 60$ ms^{-1}, and $l_{zmax} < 3g$. A simple drag function was used of the form $D = C_{D0} + k(C_L - C_{LminD})^2$ where k is the reciprocal of the product of π, aspect ratio and Oswald efficiency coefficient. Each trajectory uses $N = 200$.

The first example problem is the generation of a trajectory from an initial point in straight and level unaccelerated flight to a final point in straight and level unaccelerated flight with the same track angle and speed, vertically above the initial point. The spatial parameterization was by seventh degree polynomials defined by $\mathbf{r}_{0,f}, \mathbf{r}'_{0,f}, \mathbf{r}''_{0,f}$, and $\mathbf{r}'''_{0,f}$, and speed parameterization was $v = 23$ ms^{-1}. The optimization vector was reduced to 4 dimensions $(\tau_f, \mathbf{r}_0'''^T)^T$ by setting $\mathbf{r}_f''' \triangleq (0,0,0)^T$. Thus the trajectory was guaranteed to meet position, speed and acceleration boundary conditions. Since the modeled aircraft had insufficient thrust for a sustained vertical climb, the result should be a spiral trajectory.

Figure 8 shows the spiral trajectory produced after 113 calls to the trajectory function generator, and Figure 9 shows the corresponding control vector which can be seen to be feasible. As a confidence check on the quaternion model, the state equations Eqs. (8)–(12) were numerically integrated using the Matlab™

ode45 function; visually the result of the integration was virtually indistinguishable from the parameterized demanded trajectory and is therefore not shown superimposed on Figure 8. The mean and standard deviation of the errors in position (x, y, z) were $(-9.6, 0.45, -2.46)$ and $(7.18, 5.3, 2.08)$ with a final position error of $(-0.02, -1.08, 0.01)$ meters, demonstrating that the forward dynamics state equations closely tracked the trajectory that was generated using the inverse dynamics model.

The second example problem is the generation of a constant-radius vertical loop, to demonstrate that the model handles aerobatic maneuvers within an optimization algorithm. For this example the maximum thrust limit was increased to 120 N. Trigonometric spatial parameterization was used to fix the shape of the trajectory with the loop radius available as an optimization variable. Speed was parameterized by a fifth degree polynomial defined by $(v_{0,f}, v'_{0,f}, v''_{0,f})$. By setting $v_f = v_0$, $v'_f = v'_0$, and $v''_f = v''_0$ the 4-dimensional optimization vector became $(radius, v_0, v'_0, v''_0)$.

After 109 function calls the optimization produced a loop with zero constraint violations. Figure 10 shows the control vector which can be seen to be feasible. The numerical integration of the state equations again produced a trajectory visually virtually indistinguishable from the demanded trajectory; the mean and standard deviation of the errors in position (x, y, z) were $(-0.02, 0, 0.17)$ and $(0.13, 0, 0.14)$ with a final position error of $(-0.04, 0, 0.05)$ meters.

These two examples demonstrate that the quaternion inverse dynamics model can be applied within a potential real-time direct method, with a 4-dimensional optimization vector, to produce feasible trajectories for $\mathcal{O}(10^2)$ trajectory evaluations using both polynomial and trigonometric output parameterization functions.

V. Future Work

The quaternion model explicitly calculates roll rate p and methods exist (for example[16–20]) for smooth parameterization and interpolation of quaternion attitude $\mathbf{e}$; the Euler-angle model does not calculate p or $\dot{\mu}$ and Euler-angle parameterization does not usually produce smooth attitude changes[47]. Accordingly, maneuvers such as high-g level-off from a near-vertical climb or 'negative-g' flight are difficult to represent smoothly using the Euler-angle model but could be represented by the quaternion model with an appropriate specification to relax the assumption of roll alignment with positive normal load factor.

The computational load of optimizing a trajectory in a direct method depends on the number of calls to the trajectory function generator, the number of nodes N per trajectory, and the computational load of each node. Hence reducing N would improve overall computational performance, but this increases δt_j which may reduce the ability of a trajectory tracking controller to track the trajectory and in the presence of obstacles may reduce confidence in feasibility. The unit quaternion state equation Eq. (12) is linear if controls are held constant between nodes in which case Eq. (12) is time-invariant within each segment which therefore admits a closed-form forward dynamics evaluation at time t_{j-1} of attitude at t_j

$$\mathbf{e}_j = \exp(\mathbf{\Omega}\delta t)\,\mathbf{e}_{j-1} \tag{38}$$

Eq. (38) is a candidate merit function for increasing δt_j during trajectory evaluation to reduce the computational load whilst retaining trajectory tracking and feasibility.

VI. Conclusion

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. Application of path constraints across segments when the inverse dynamics model is discretized must be considered and can be readily implemented. The model is suitable for application to aerobatic maneuvers and examples of loop, horizontal helix, and vertical helix maneuvers show that the model produces feasible control vectors for these maneuvers. Examples of trajectory generation using a direct method with a derivative-free optimization algorithm demonstrate that the quaternion inverse dynamics model can be applied within a candidate real-time direct method with a 4-dimensional optimization vector to produce feasible trajectories, including inverted positive-g flight, using both polynomial and trigonometric output parameterization functions.

Acknowledgments

This work is supported by a United Kingdom Engineering and Physical Sciences Research Council Industrial CASE award; the award is 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.

[2]Betts, J., *Practical Methods for Optimal Control Using Nonlinear Programming*, SIAM, Philadelphia,USA, 2001.

[3]von Stryk, O. and Bulirsch, R., "Direct and Indirect Methods for Trajectory Optimization," *Annals of Operational Research*, Vol. 37, 1992, pp. 357–373.

[4]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.

[5]Seywald, H., "Trajectory Optimization Based on Differential Inclusions," *Journal of Guidance, Control, and Dynamics*, Vol. 17, No. 3, 1994, pp. 480–487.

[6]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.

[7]Lane, S. and Stengel, R., "Flight Control Design Using Nonlinear Inverse Dynamics," *Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 4, 1988, pp. 471–483.

[8]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.

[9]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.

[10]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.

[11]Miele, A., *Flight Mechanics*, Addison-Wesley, Massachusetts,USA, 1962.

[12]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.

[13]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.

[14]Fliess, M., Levine, J., Martin, P., and Rouchon, P., "On Differentially Flat Nonlinear Systems," *Proceedings of the IFAC Symposium NOLCOS92*, Bordeaux, France, 1992, pp. 408–412.

[15]Fliess, M., Levine, J., Martin, P., and Rouchon, P., "Flatness and Defect of Nonlinear Systems: Introductory Theory and Examples," *International Journal of Control*, Vol. 61, No. 6, Jan. 1995, pp. 1327–1361.

[16]Shoemake, K., "Animating Rotation with Quaternion Curves," *Computer Graphics*, Vol. 19, No. 3, July 1985, pp. 245–254.

[17]Barr, A., Currin, B., Gabriel, S., and Hughes, J., "Smooth Interpolation of Orientations with Angular Velocity Constraints using Quaternions," *Computer Graphics*, Vol. 26, No. 2, July 1992, pp. 313–320.

[18]Schmidt, J. and Niemann, H., "Using Quaternions for Parameterizing 3-D Rotations in Unconstrained Nonlinear Optimization," *Proceedings of the Vision Modeling and Visualization Conference*, 2001, pp. 399–406.

[19]Ramamoorthi, R. and Barr, A., "Fast construction of Accurate Quaternion Splines," *Proceedings of the International Conference on Computer Graphics and Interactive Techniques*, ACM Press/Addison-Wesley Publishing Co., New York, 1997, pp. 287–292.

[20]Johnstone, J. and Williams, J., "A Rational Quaternion Spline of Arbitrary Continuity," Tech. rep., Department of Computer and Information Sciences, University of Alabama at Birmingham, Aug. 1999.

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

[22]Kaminer, I., Yakimenko, O., and Pascoal, A., "Coordinated Control of Multiple UAVs for Time-Critical Applications," *Proceedings of the 27th IEEE Aerospace Conference*, Big Sky, Montana,, March 2006.

[23]Kaminer, I., Yakimenko, O., Pascoal, A., and Ghabcheloo, R., "Path Generation, Path Following and Coordinated Control for Time-Critical Missions of Multiple UAVs," *Proceedings of the American Control Conference*, June 2006, pp. 4906–4913.

[24]Knoebel, N., Osborne, S., Snyder, D., McLain, T., Beard, R., and Eldredge, A., "Preliminary Modeling, Control, and Trajectory Design for Minature Autonomous Tailsitters," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Aug. 2006.

[25]Stengel, R., *Flight Dynamics*, Princeton University Press, Princeton, 2004.

[26]Drury, R. and Whidborne, J., "Quaternion-based Inverse Dynamics Model for Evaluating Aerobatic Aircraft Trajectories," *Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 4, July 2009, pp. 1388–1391.

[27]Baruh, H., *Analytical Dynamics*, McGraw-Hill, Boston, 1999.

[28]Shanmugavel, M., Tsourdos, A., Zbikowski, R., and White, B., "Path Planning of Multiple UAVs Using Dubins Sets," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, San Francisco, Aug. 2005.

[29]Shanmugavel, M., Tsourdos, A., Zbikowski, R., and White, B., "3D Dubins Sets Based Coordinated Path Planning for Swarm of UAVs," *AIAA Guidance, Navigation and Control Conference and Exhibit*, Keystone,Co., Aug. 2006.

[30]Shanmugavel, M., Tsourdos, A., Zbikowski, R., White, B., Rabbath, C., and Lechevin, N., "A Solution to Simultaneous Arrival of Multiple UAVs using Pythagorean Hodograph Curves," *Proceedings of the American Control Conference*, June 2006, pp. 2813–2818.

[31]Shanmugavel, M., Tsourdos, A., Zbikowski, R., and White, B., "3D Path Planning for Multiple UAVs using Pythagorean Hodograph Curves," *AIAA Guidance, Navigation and Control Conference and Exhibit*, Hilton Head, Aug. 2007.

[32]Shanmugavel, M., Tsourdos, A., Zbikowski, R., and White, B., "Path Planning of Multiple UAVs with Clothoid Curves," *17th IFAC Symposium on Automatic Control in Aerospace*, Toulouse, France, Aug. 2007.

[33]Yakimenko, O., "Direct Method for Rapid Prototyping of Near-Optimal Aircraft Trajectories," *Journal of Guidance, Control, and Dynamics*, Vol. 23, No. 5, Oct. 2000, pp. 865–875.

[34]Yakimenko, O.A., X. Y. and Basset, G., "Computing Short-Time Aircraft Maneuvers Using Direct Methods," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Hawaii, Aug. 2008.

[35]Taranenko, V. T. and Momdzhi, V. G., *Direct Variational Method in Boundary Problems of Flight Dynamics*, Mashinostroenie Press, 1986, (in Russian).

[36]Yakimenko, O., "Direct Method for Real-Time Prototyping of Optimal Control," *Proceedings of the UKACC International Control Conference*, Glasgow, Sept. 2006.

[37]Press, W., Teukolsky, S., Vetterling, W., and Flannery, B., *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, New York, 3rd ed., 2007.

[38]Bevilacqua, R., Yakimenko, O., and Romano, M., "On-line Generation of Quasi-Optimal Docking Trajectories," *Proceedings of the 7th Dynamics and Control of Systems and Structures in Space Conference*, Greenwich, England, July 2006, pp. 203–218.

[39]Kolda, T., Lewis, R., and Torczon, V., "Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods," *SIAM Review*, Vol. 45, No. 3, 2003, pp. 385–482.

[40]Hough, P., Kolda, T., and Torczon, V., "Asynchronous Parallel Pattern Search for Nonlinear Optimization," *SIAM Journal of Scientific Computing*, Vol. 23, 2001, pp. 134–156.

[41]Lagarias, J., Reeds, J., Wright, M., and Wright, P., "Convergence Properties of the Nelder-Mead Simplex Algorithm in Low Dimensions," Tech. Rep. 96-4-07, Computing Sciences Research Center, Bell Laboratories, New Jersey, May 1997.

[42]Price, C., Coope, I., and Byatt, D., "A Convergent Variant of the Nelder-Mead Algorithm," *Journal of Optimization Theory and Applications*, Vol. 113, No. 1, April 2002, pp. 5–19.

[43]Nelder, J. and Mead, R., "A Simplex Method for Function Minimization," *The Computer Journal*, Vol. 8, No. 7, 1965, pp. 308–313.

[44]Nocedal, J. and Wright, S., *Numerical Optimization*, Springer, New York, 2nd ed., 2006.

[45]Fletcher, R., *Practical Methods of Optimization*, Wiley, Chichester, 2nd ed., 1987.

[46]Griffin, J. and Kolda, T., "Nonlinearly-Constrained Optimization Using Asynchronous Parallel Generating Set Search," Tech. Rep. SAND2007-3257, Sandia National Laboratories, Albuquerque, May 2007.

[47]Watt, A. and Watt, M., *Advanced Animation and Rendering Techniques: Theory and Practice*, ACM Addison-Wesley, 1992.

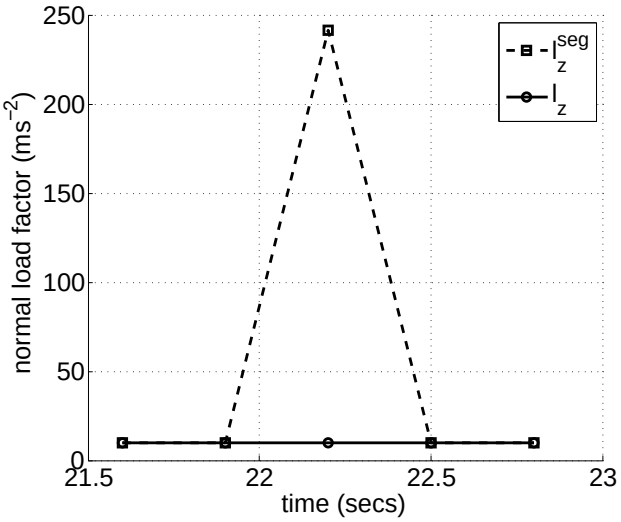

Figure 1. Comparison of l_z and l_z^{seg} for 23 $\mathbf{ms}^{-1}$ and $\delta t = 0.3$ secs

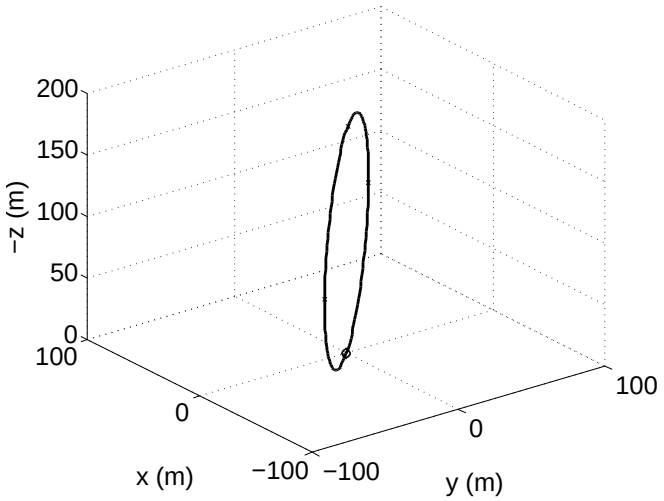

Figure 2. Vertical Loop Trajectory

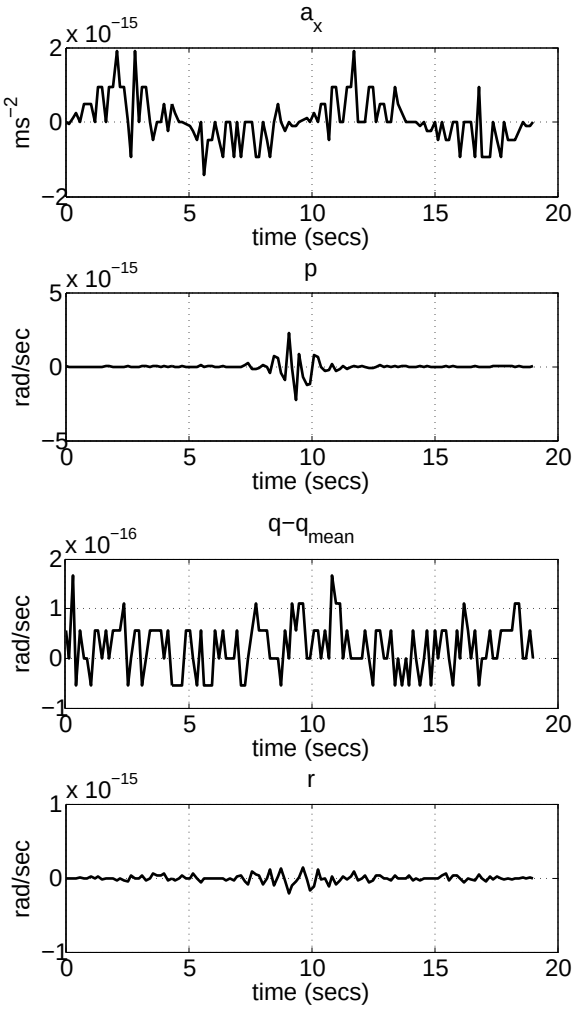

Figure 3. Control Vector for a Vertical Loop

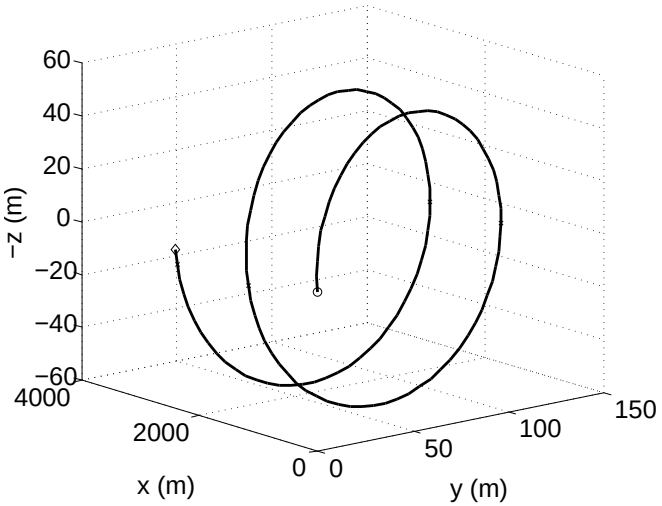

Figure 4. Horizontal Helix Trajectory

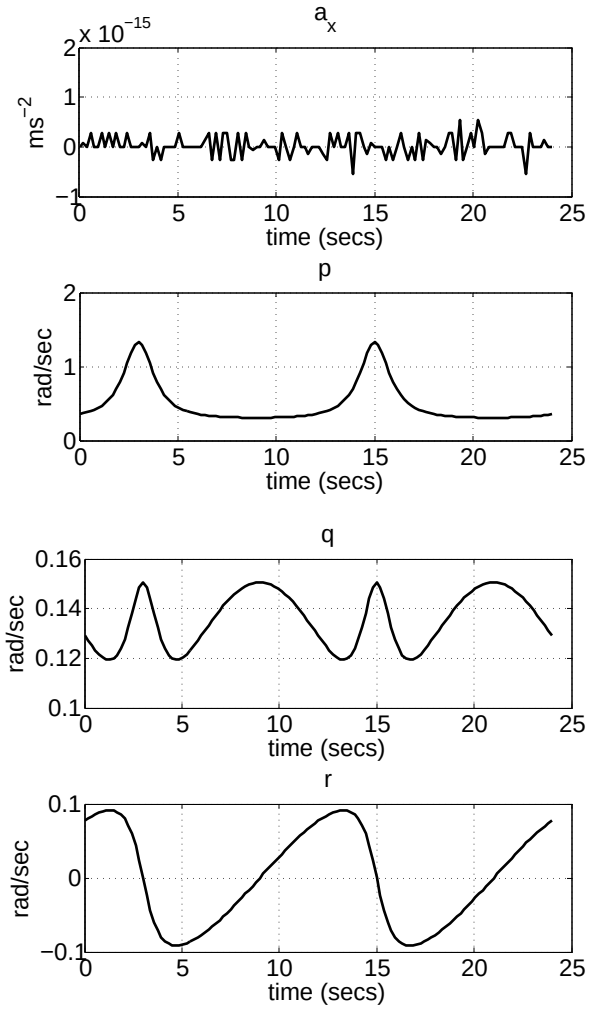

Figure 5. Control Vector for a Horizontal Helix

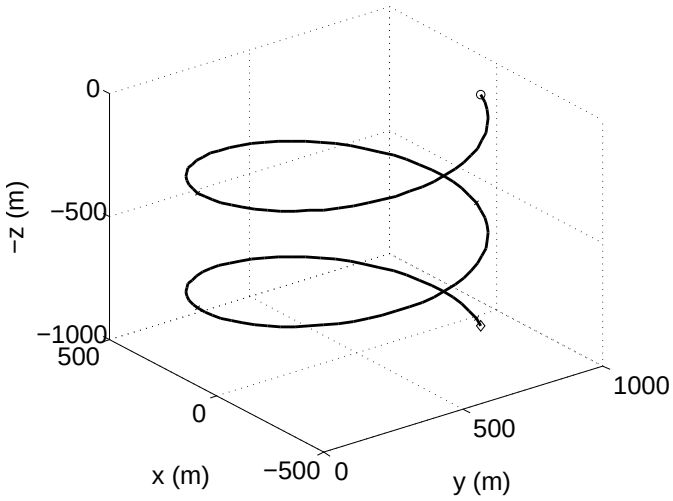

Figure 6. Vertical Helix Trajectory

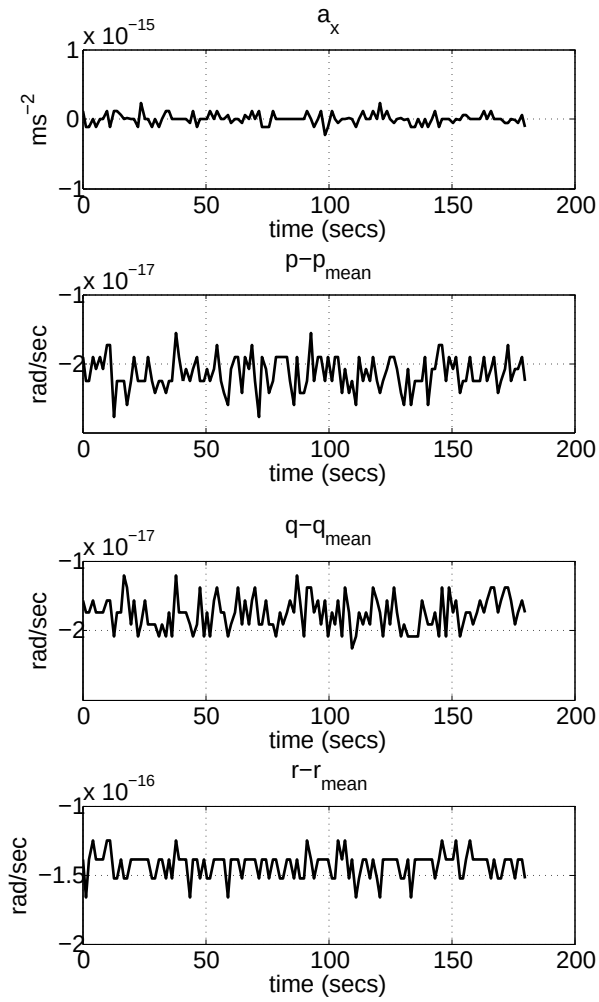

Figure 7. Control Vector for a Vertical Helix

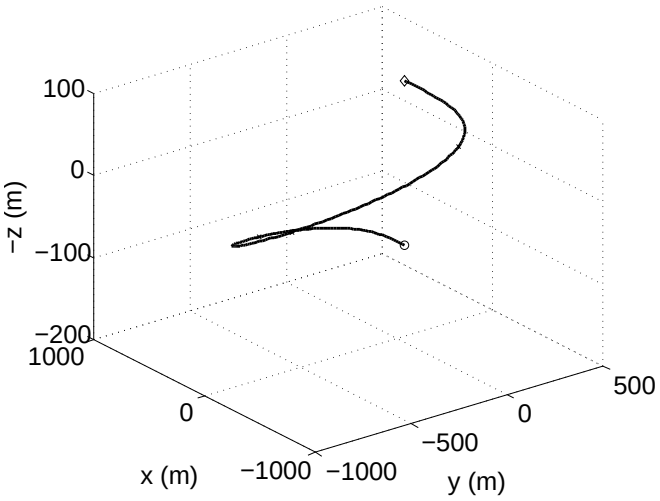

Figure 8. Spiral Trajectory from Optimization

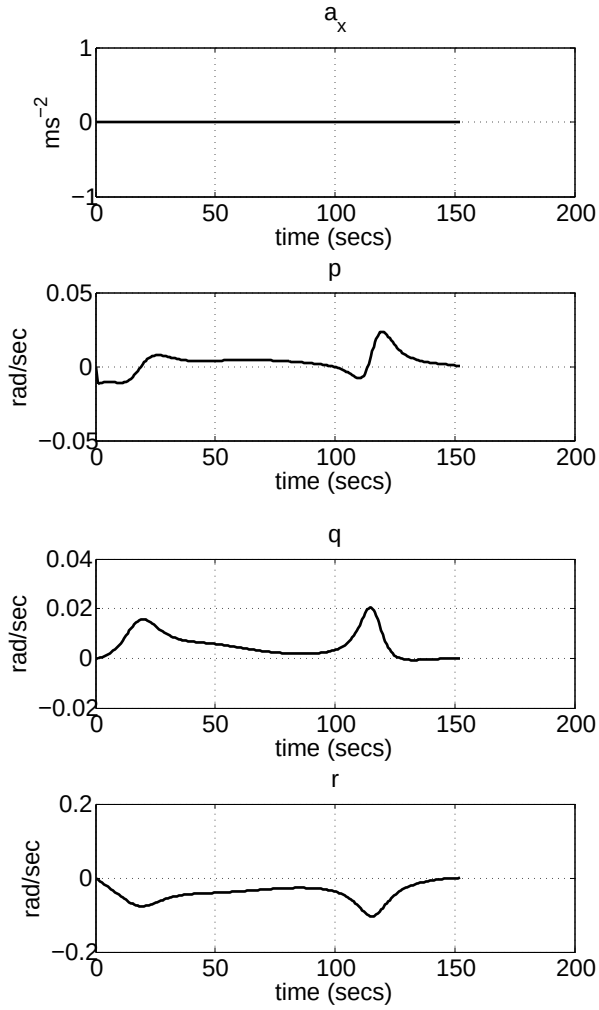

Figure 9. Control Vector for Spiral of Figure ??

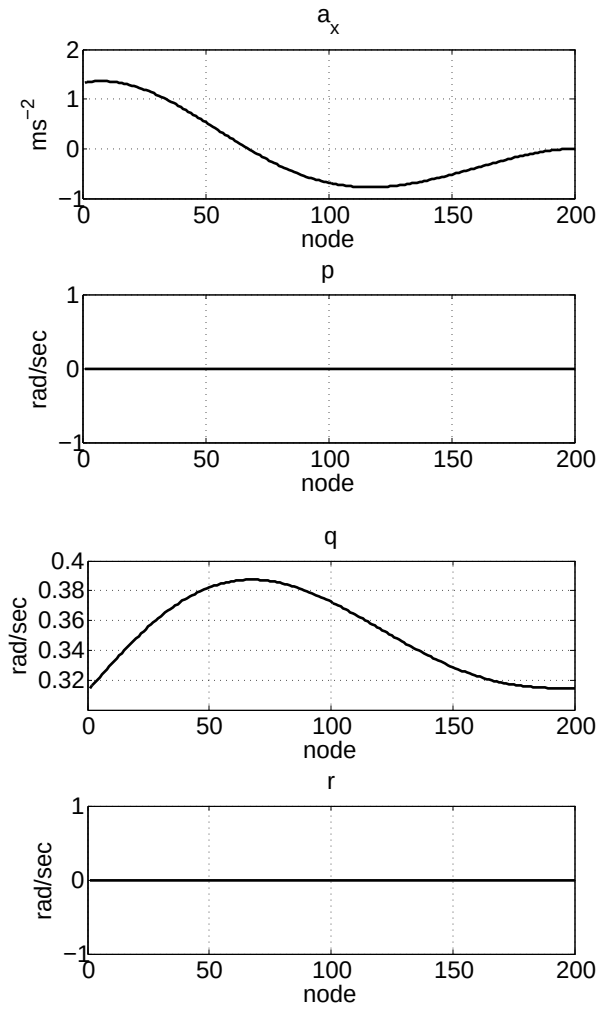

Figure 10. Control Vector for Varying Speed Constant Radius Loop