Mechatronics 21 (2011) 844–860

Contents lists available at ScienceDirect

Mechatronics

journal homepage: www.elsevier.com/locate/mechatronics

Autonomous terrain-following for unmanned air vehicles

Raza Samar [*], Abdur Rehman

Centre for Control & Instrumentation, Engineering & Scientific Commission, Islamabad, Pakistan

ARTICLE INFO

Article history:
Available online 25 October 2010

Keywords:
Terrain-following
Decision making and autonomy
Altitude control
Mission planning
Guidance and control of UAVs
Trajectory tracking and path following
robust control applications
Eectromechanical actuators

ABSTRACT

This paper presents an integrated guidance and control design scheme for an unmanned air vehicle (UAV), and its flight test results. The paper focuses on the longitudinal control and guidance aspects, with particular emphasis on the terrain-following problem. An introduction to the mission, and the terrain-following problem is given first. Waypoints for climb and descent are defined. Computation of the reference trajectory in the vertical plane is discussed, including a terrain-following (TF) algorithm for real-time calculation of climb/descent points and altitudes. The algorithm is particularly suited for online computation and is therefore useful for autonomous flight. The algorithm computes the height at which the vehicle should fly so that a specified clearance from the underlying terrain is always maintained, while ensuring that the vehicle's rate of climb and rate of descent constraints are not violated. The output of the terrain-following algorithm is used to construct a smooth reference trajectory for the vehicle to track. The design of a robust controller for altitude tracking and stability augmentation of the vehicle is then presented. The controller uses elevators for pitch control in the inner loop, while the reference pitch commands are generated by the outer altitude control loop. The controller tracks the reference trajectory computed by the terrain-following algorithm. The design of an electromechanical actuator for actuating the control surfaces of the vehicle during flight is also discussed. The entire guidance and control scheme is implemented on an actual experimental vehicle and flight test results are presented and discussed.

1. Introduction

The notion of unmanned vehicles capable of autonomous flight is fast becoming a reality, with the level of autonomy continuously on the increase. The availability of high performance embedded computing platforms combined with the development of sophisticated algorithms has transformed the realm of unmanned vehicles within a span of a few years. In this paper we present an integrated approach to the design of a practical and autonomous terrain-following guidance and control system for an experimental aircraft, and discuss its in-flight performance.

The UAV under consideration is a high performance vehicle, shown in Fig. 1. It is propeller driven in a push-configuration. Pitch control is provided by a set of canards located forward of the main wing, roll control is provided by ailerons on the main wing, and the two vertical tails have rudders. The cruising speed is about 50 m/s. The main tasks of the planning, guidance and control system are to:

- given a mission in terms of geographical waypoints (latitude and longitude information), generate a series of altitude change (or climb/descent) points while satisfying vehicle constraints and providing a minimum clearance above the terrain,
- use the altitude change points to generate a feasible terrain-following reference trajectory for the vehicle,
- fly the vehicle on the reference path with minimum altitude error, and
- provide robust stabilization and control during flight.

Various approaches for terrain-following and avoidance have been discussed in the literature, see for example [1–4]. In [1] an optimization problem is set up in which cubic splines are optimized to lie close to the underlying terrain. Constraints are satisfied at node points which are also the optimization parameters. The larger the number of node points, the more the number of optimization parameters and hence the greater the computational cost. In [5] the vertical path is generated by using a large number of parabolic segments called the *pullup* and *pushover* parabolas. The parabolas are designed to satisfy the acceleration limits of the vehicle; terrain peaks are crossed horizontally, with a given clearance. Lu and Peirson [2] formulate the terrain-following problem as an optimal control problem and suggest a numerical method for its solution based on an inverse dynamics approach. The time of flight is included in the objective function and the dynamics of the aircraft are considered. The analysis however focuses on *offline* trajectory planning only and the results are not readily applicable to real-time onboard implementation. A nonlinear flight control law for terrain-following is discussed in [3] based on the predictive

[*] Corresponding author.
E-mail addresses: raza.samar@gmail.com, raza@mail.comsats.net.pk (R. Samar).

0957-4158/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mechatronics.2010.09.010

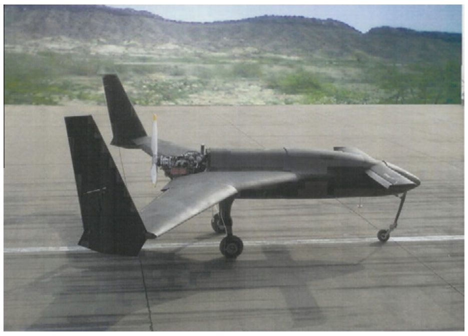

Fig. 1. A photograph of the experimental UAV.

control method. The controller commands the throttle setting and the angle of attack directly, hence a separate inner-loop controller for the angle of attack is required to be implemented. Terrain-following control of unmanned underwater vehicles has also been considered using neural networks, where the vehicle is controlled over varying seabed terrain [6]. In [7] the best turn or heading angle is computed to optimize the integrated terrain altitude and threats, which are modeled for a mission area. The time of flight can also be included in the cost function. A genetic algorithm approach for generation of the vehicle's trajectory, specially in the event of encountering a threat is discussed in [8]. Graph-search algorithms are discussed in [4] in which a search over the set of feasible trajectories is performed; Dijkstra's shortest path algorithm is used. Speedups of the algorithm are also discussed, however these remain computationally prohibitive for onboard application in an autonomous mode.

The main contribution of this paper is to provide an integrated framework for the design of practical algorithms capable of autonomous terrain-following flight. Real-time implementation of these algorithms on available computational engines is easily possible. The terrain-following algorithm is developed first; the algorithm takes as an input the geographical route (latitude, longitude information) of the mission track and generates a sequence of altitude change points and corresponding altitudes. The algorithm is simple and fast and has a performance approaching that of more sophisticated algorithms [9]. The output of the terrain-following algorithm is used to generate smooth reference trajectories for the altitude tracking controller to follow. The design of the altitude controller is then presented followed by a discussion of the flight test results. The emphasis throughout stays on having real-time capable algorithms suitable for autonomous application.

This paper is organized as follows. Section 2 defines the mission plan on which we intend the UAV to fly, and gives definitions of waypoints and altitude change points. Here we also formulate the terrain-following *planning* and *control* problems in more specific terms. Section 3 develops the basic algorithm for autonomous terrain-following flight. A sequence of altitude change points with corresponding leg altitudes are computed. Comparison with an optimal algorithm based on cubic splines is also given. A smooth reference trajectory needs to be generated from the sequence of computed altitude points, this is also discussed here. Section 4 describes the design of a terrain-following controller that provides stabilization and tracking control of the altitude reference trajectory. The controller needs to be robust and provide precise tracking of the commanded trajectory. Section 5 presents the design of an electromechanical actuator employed for control surface actuation of the vehicle. Flight test results are presented and discussed in Section 6; Section 7 concludes the paper.

2. Problem formulation

2.1. Assumptions

It is assumed that a two-dimensional route or plan is available beforehand which may consist of a sequence of turning waypoints (will be referred to as simply waypoints henceforth). Each of these waypoints is defined by its geographical coordinates, i.e., latitude and longitude. In other words we assume that a route in the horizontal plane (over the surface of the earth to be precise) which the vehicle needs to fly over, is given in terms of its geographical coordinates. The given route may involve straight flight throughout or straight and turning segments with loiter, depending on the requirements of a particular mission. The vehicle's rate of ascent and descent may be different for straight and turning parts of its flight. It is also assumed that a digital elevation map (DEM) of the surface of the earth for the proposed mission area is also available. The DEM is a map of elevations of points on the surface of the earth referenced to mean sea level. The dynamical constraints of the air vehicle in terms of its rates of climb and descent are also assumed to be known. The vehicle is assumed to be equipped with an inertial navigation unit (INU), aided by GPS,[1] for determining the position of the vehicle during flight. The INU comprises of inertial sensors (gyroscopes and accelerometers), which have drift and bias errors. These errors cause an increase in the navigation error with time, and therefore aiding with GPS measurements is required. We assume here that the INU error growth rate is fixed and known. To summarize we consider the following information as given:

- the mission track in terms of latitude and longitude information,
- the digital elevation map for the mission track,
- the rate of climb (ROC) and rate of descent (ROD) constraints (these may not be equal, and may also vary for different parts of the mission), and
- the INU error growth rate.

2.2. Terrain elevation profile

An elevation profile of the terrain is required to be generated over which the vehicle is being planned to fly (i.e., which corresponds to the given mission track). However as indicated above the vehicle may not be able to fly *exactly* over the given track. Winds, gusts and other disturbances may cause lateral cross-track errors. Also navigation system errors imply an uncertainty in the knowledge of the horizontal position of the vehicle. The terrain elevation for the mission must be generated by giving due consideration to these factors. In other words the planning in the vertical dimension should be robust and insensitive to possible perturbations in the given two-dimensional (horizontal) mission. Terrain profile information is to be computed as elevation versus range data.

2.3. The terrain-following problem

The terrain-following problem is to find a trajectory in the vertical plane (flight altitude versus range) that follows the contour of the terrain underneath with a given clearance while respecting the dynamical constraints of the vehicle. This then serves as the reference trajectory for the vehicle to fly on. The idea is to generate a reference altitude profile for the vehicle that is feasible with regards to the rates of climb and descent, and also maintains a given clearance above the underlying terrain. The terrain profile as dis-

[1] The Global Positioning System.

cussed in Section 2.2 above along with the climb and descent rate constraints serve as inputs to this problem.

Different performance measures can be used to define the optimality of the solution to the terrain-following problem. Here we take the total error squared over the entire range as the performance measure:

$$J = \int_0^{R_F} e^2 dR, \tag{1}$$

where R_F is the final range value (the total distance to be traveled). The error e is defined as: $e = h - T - C_{min}$, where T is terrain elevation, C_{min} is minimum ground clearance and h is the height to be computed (on which we would like the vehicle to fly). The error therefore represents the closeness of the computed altitude h to the underlying terrain. We define the slope s, the curvature k and the kink p to be first, second and third derivatives respectively, of h with respect to range R. Assuming the speed of the UAV to be approximately constant, the rate of climb (or descent) can be expressed as: $s = \frac{dh}{dR}$ or equivalently: $s = \frac{dh/dt}{V}$. The curvature k is expressed as: $k = \frac{ds}{dR} = \frac{d^2h/dt^2}{V^2}$. The kink p is defined as: $p = \frac{dk}{dR}$ or as: $p = \frac{d^3h/dt^3}{V^3}$. The objective function (1) is to be minimized subject to the following constraints:

$$0 \leqslant e, \tag{2}$$
$$s_{min} \leqslant s \leqslant s_{max}, \tag{3}$$
$$k_{min} \leqslant k \leqslant k_{max}, \tag{4}$$
$$p_{min} \leqslant p \leqslant p_{max}, \tag{5}$$

where the subscripts '*min*' and '*max*' refer to minimum and maximum allowed values for these variables. The curvature and the kink defined as above in the range domain are analogous to 'normal acceleration' and 'jerk' in the time domain. Trajectories that do not comply with system dynamics and constraints can place impractical demands on the tracking controller, and therefore must be avoided.

The terrain-following problem is therefore to compute the best height h, and at the same time satisfy the above constraints. The solution to this problem must be capable of online (autonomous) implementation. Furthermore if the lateral mission plan gets changed for some reason, it must be possible to generate new vertical trajectories quickly and autonomously onboard the vehicle.

2.4. The tracking problem

The terrain-following algorithm takes into consideration the dynamical constraints of the vehicle, and thus provides the tracking controllers with feasible reference trajectories. Once a reference altitude profile for the vehicle is available, the next step is to make the vehicle fly on the prescribed trajectory. The tracking problem is to design an altitude control system that uses the control surfaces of the air vehicle (elevators) to control the altitude as close as possible to the reference value. This altitude tracking must be achieved in the presence of disturbances such as winds and gusts, and across the flight envelope of the vehicle. The controller therefore must exhibit good robustness properties. Furthermore the control system should also provide stability augmentation during various flight maneuvers.

3. Terrain-following algorithms

We shall discuss two terrain-following algorithms here, the *Stair* algorithm and the *Optimal Spline* algorithm. But first we discuss generation of terrain elevation data for the mission track.

3.1. Terrain elevation profile generation

If the two-dimensional (horizontal plane) mission track is available, an elevation profile of the terrain underneath can be generated easily using the digital elevation map. However as indicated in Section 2.2 above, there is always a possibility of deviation (cross-track errors) of the vehicle from the proposed path. This could be due to imperfect lateral track control of the vehicle, disturbances such as winds or gusts, and uncertainty or errors in the knowledge of the position of the vehicle. The positional (navigation) errors come about because of the inertial navigation unit (INU) onboard the vehicle. GPS aiding prevents the errors from growing too large but in times of GPS outage, the inertial system errors tend to diverge. Low cost INUs usually employed in UAVs have low accuracy inertial sensors and the rate of error growth with time is therefore high. Uncertainty in the knowledge of the position of the air vehicle thus generally increases with time. This is shown in Fig. 2 where the corridor of uncertainty centered around the nominal path of the vehicle is shown to increase with time (and distance traveled). Each circle indicates the uncertainty in the position of the vehicle at that range (or distance). The radii are continuously increasing since the error in the navigation solution grows with time (and range), and therefore the uncertainty in position also grows.

Terrain profile is data containing terrain elevation versus range information. This information is used by the terrain-following algorithms. However due to the growing position uncertainty of the vehicle as indicated in Fig. 2, we need to find a robust elevation profile so that no peaks in the probable region of presence of the vehicle are missed out. Therefore for a given range, one should pick the highest value in the region so that chances of ground collision are ruled out. This is done as follows:

- The entire route is divided into a number of small segments and arcs. For straight part of the route, a length of say 100 m can be used. For turns a resolution of $100/R_t$ radians (equivalent to 100 m) can be used, R_t being the turn radius of the vehicle.
- At the end point of each segment (or arc), a circle is drawn with radius equal to the navigation error (INU error) expected while traveling up to that point.
- A maximum of all elevations (for all points inside the circle) is computed and this value is assigned to the centre of the circle.
- The maximum elevations are stored as elevation data versus range.

Thus terrain elevation data is obtained which is based on worst case elevation values.

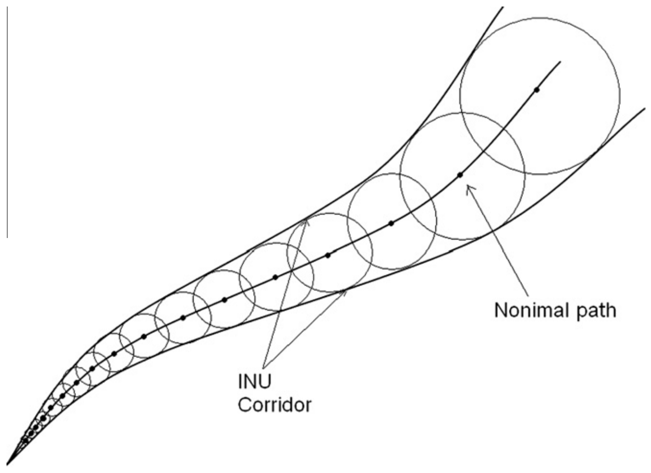

Fig. 2. INU corridor for terrain elevation profile generation.

3.2. Stair algorithm

It is a simple and computationally fast algorithm, ideally suited for online application. It automatically decides altitude change points and corresponding heights. The algorithm works on the given terrain elevation data and generates a set of altitude change points as a function of distance from the initial or start point. The inputs to the algorithm are:

- terrain elevation versus range data,
- minimum ground clearance,
- rate of ascent and descent of the vehicle (may be fixed or varying),
- length of a section or patch (user-defined parameter, discussed below), and
- minimum separation between a descent and a subsequent climb (user-defined parameter, discussed below).

The entire route is divided into sections, and the algorithm works out section heights according to the constraints imposed. The algorithm consists of the following steps:

Step 1: Divide the entire route into sections (or patches) of equal length ΔR (say 5 km); note that the sections we consider here are not the same as the segments we considered when generating the elevation profile in Section 3.1 above. Using terrain elevation data, compute the maximum elevation of all points in a section; we call this h_i^{max} for the i^{th} section. Set the section altitude as:

$$h_i = h_i^{max} + C_{min}, \tag{6}$$

where h_i is the section height and C_{min} is the minimum clearance. Collect all section heights $h_0, h_1, \ldots, h_n$ and find their differences as:

$$dh_i = h_{i+1} - h_i. \tag{7}$$

A positive dh_i indicates the start of a climb, a negative dh_i indicates a descent (Fig. 3).

Step 2: For the ascent case (positive dh_i), select the first point of the next patch (section) as a node; for descent (negative dh_i) select the end point of the current patch as the node. These nodes are the ascent end and descent start points, indicated by dark dots in Fig. 3.

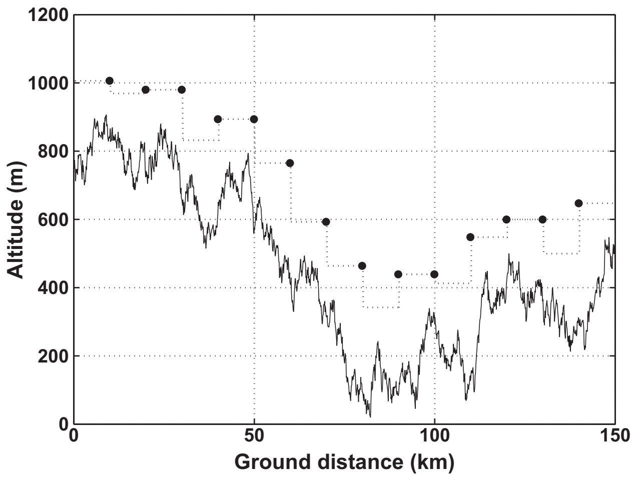

Fig. 3. Section heights and node points versus range for a sample terrain profile.

Step 3: For ascent, join the node (start point of next section) with the current section by a line with a slope equal to the rate of climb of the vehicle. Find the point of intersection of this line with the current patch, if it does not intersect the current patch, look for intersection with previous sections. This point is the ascent start or climb start point. Collect all ascent start points and embed them in the node point list in order. Note that if the ascent start point does not lie on the current patch but on some previous patch, then all intermediate patches and nodes will be deleted. All ascent start points become node points.

Step 4: For descent, find the intersection point of a line passing through the descent node point with the next patch/section. This line should have a slope equal to the rate of descent of the vehicle. If there is no intersection with the next adjacent section, go to the next section and so on. The intersection point is the descent stop point, all intermediate sections (and nodes) where an intersection cannot be found will be removed. Collect all descent stop points and embed them in the vector of node points. In this way the node list is updated with deletion of some node points, and with appropriate ascent start and descent stop points, as a function of distance from the start point.

Step 5: In a valley where the terrain elevation decreases and then increases again, the scenario becomes complex and two situations are likely to occur. It may happen that the descent stop point occurs later (at a greater distance) than the ascent start point as shown in Fig. 4. In order to rectify this, and also to have a gap (g is the specified gap) between descent and the next ascent, the following procedure is adopted. In Fig. 4 (d_1,h_1), (d_2,h_2), (d_3,h_3) and (d_4,h_4) are the ranges and heights of the node points and are known. The first two nodes are the descent start and stop nodes, the last two are the ascent start and stop nodes, respectively. The gradients $\tan\theta_1$ and $\tan\theta_2$ can be readily computed. Also we can write:

$$b = d_2 - d_3,$$

$$H_1 = c\tan\theta_1 \Rightarrow c = \frac{H_1}{\tan\theta_1},$$

$$H_1 = e\tan\theta_2 \Rightarrow e = \frac{H_1}{\tan\theta_2},$$

Now

$$b = c + e = \frac{H_1}{\tan\theta_1} + \frac{H_1}{\tan\theta_2},$$

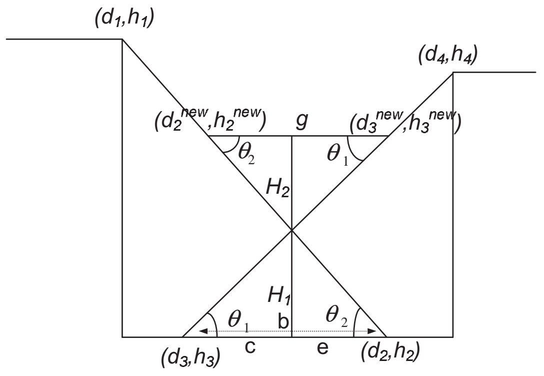

Fig. 4. A descent followed by an ascent with overlap.

and therefore

$$H_1 = b\frac{\tan\theta_1 \tan\theta_2}{\tan\theta_1 + \tan\theta_2}.$$

H_2 is calculated as:

$$H_2 = g\frac{\tan\theta_1 \tan\theta_2}{\tan\theta_1 + \tan\theta_2}. \tag{8}$$

New node points (d_2^{new}, h_2^{new}) and (d_3^{new}, h_3^{new}) can be found as:

$$\begin{aligned} d_2^{new} &= d_2 - \frac{H_1 + H_2}{\tan\theta_2}, \\ h_2^{new} &= h_2 + (H_1 + H_2), \\ d_3^{new} &= d_3 + \frac{H_1 + H_2}{\tan\theta_1}, \\ h_3^{new} &= h_3 + (H_1 + H_2), \end{aligned} \tag{9}$$

Old node points (d_2, h_2) and (d_3, h_3) are replaced with new node points (d_2^{new}, h_2^{new}) and (d_3^{new}, h_3^{new}) in the node list.
A different scenario occurs when a descent–ascent gap exists but is less than the specified value g. Now we raise the descent stop and ascent start points along their slope lines as follows, see Fig. 5. Here $b = d_3 - d_2$, $c = \frac{H}{\tan\theta_2}$, $e = \frac{H}{\tan\theta_1}$ and it is seen from Fig. 5 that:

$$g = b + c + e = (d_3 - d_2) + \frac{H}{\tan\theta_2} + \frac{H}{\tan\theta_1}. \tag{10}$$

Hence H can be found as:

$$H = (g - d_3 + d_2)\frac{\tan\theta_1 \tan\theta_2}{\tan\theta_1 + \tan\theta_2}, \tag{11}$$

and the new node points are:

$$\begin{aligned} d_2^{new} &= d_2 - \frac{H}{\tan\theta_2}, \\ h_2^{new} &= h_2 + H, \\ d_3^{new} &= d_3 + \frac{H}{\tan\theta_1}, \\ h_3^{new} &= h_3 + H. \end{aligned}$$

The new node points now replace the old ones in the node list. Note that such scenarios cannot arise in an 'ascent-followed-by-descent' case because of the way the ascent and descent nodes are defined.

Step 6: For the last node point (say F) which could be the landing point, find the intersection of a line (with the vehicle's descent rate as its slope) joining F to the previous section, and if an intersection with the previous section does not exist, move to the section before it and so on until an intersection is found. This intersection point is the final descent start point for landing. Delete all other node points between this point and F.

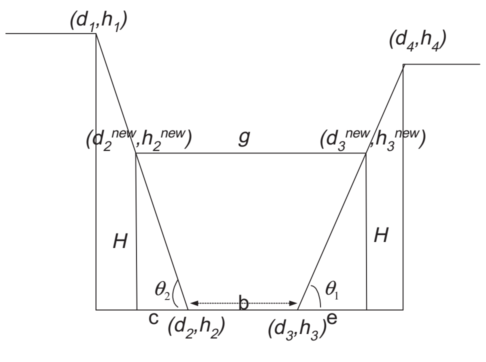

Fig. 5. A descent followed by an ascent with gap less than the specified value.

The reference path generated using these altitude change (or node) points has the vehicle's rate of climb and descent constraints built into it; other constraints can be implemented indirectly by using low-pass filters. As we have seen, a minimum distance (the user-defined gap g) can be kept between a descent and a subsequent climb for a feasible flight profile. After the final node list is generated, it can be screened so that minor height changes in adjacent flight segments can be neglected if these fall within a certain tolerance. In other words two adjacent sections can be merged into one if their heights are close to each other; this will avoid unnecessary and small climb/descend commands to the control system of the vehicle.

Terrain-following trajectories computed using the Stair algorithm for different section lengths and a clearance of 100 m are shown in Figs. 6–8. Fig. 6 shows the trajectory computed with a section length of 2.5 km, Fig. 7 uses a length of 5 km whereas Fig. 8 employs a section length of 10 km. The terrain profile is also shown in the figures in light color. Climb start/end points and descent start/end points are shown as black dots. Rate of climb and descent constraints are met in each case. A smaller section length allows a closer following of the terrain profile, but with more climb/descent maneuvers for the control system to execute. This could mean greater fuel consumption and more actuator usage. There is thus a tradeoff between how closely we want the terrain to be followed, and the climb/descent maneuvering the vehicle will be required to perform.

The algorithm presented above can be executed quickly in an onboard computer and can generate a list of altitude change points (node points) and corresponding segment heights. Next an optimal cubic spline based algorithm will be briefly discussed that minimizes the cost function (1) and implements all constraints using sequential quadratic programming. Although not being proposed for online application, the purpose here is to compare the performance to the Stair algorithm discussed above.

3.3. Optimal Spline algorithm

Cubic splines [10] are degree three polynomials that smoothly connect to adjoining spline polynomials. They provide smoothness and continuity by ensuring that the value of the function and its

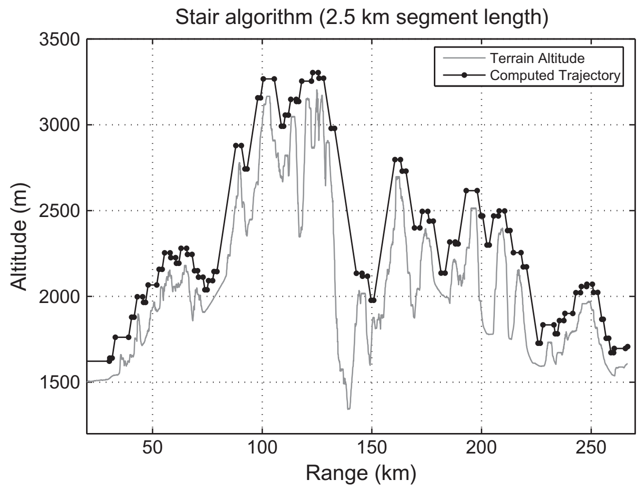

Fig. 6. Trajectory computed using the Stair algorithm with segment length of 2.5 km and clearance of 100 m.

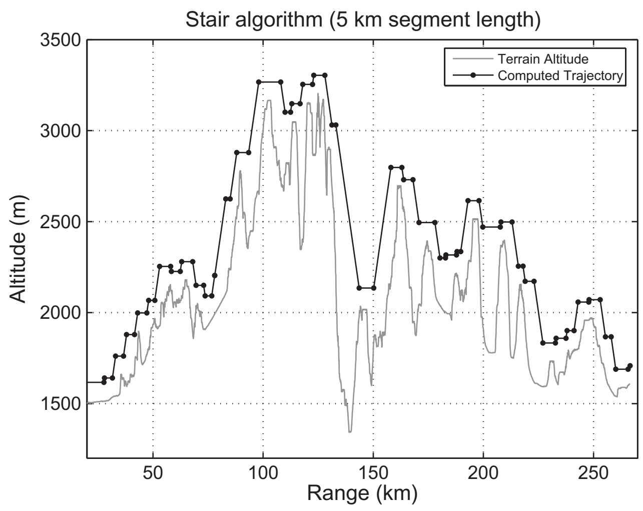

Fig. 7. Trajectory computed using the Stair algorithm with segment length of 5 km and clearance of 100 m.

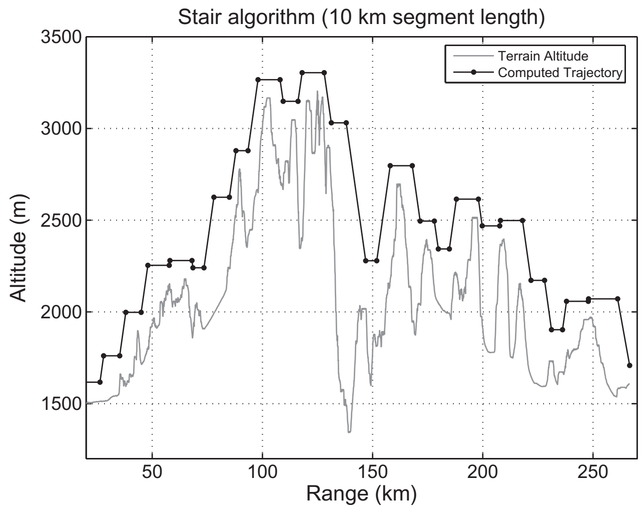

Fig. 8. Trajectory computed using the Stair algorithm with segment length of 10 km and clearance of 100 m.

first and second derivatives match with those of neighboring splines. For optimal terrain-following using splines, we divide the total range into N subintervals each of length ΔR as shown in Fig. 9. We now have $N+1$ data points and $N-1$ interior points. Let a cubic spline on the ith interval $R_i \leqslant R \leqslant R_{i+1}$ be defined as: $H_i(R) = h = a_iR^3 + b_iR^2 + c_iR + d_i$, which expresses height as a function of range in that interval. The altitudes $h_1, h_2, \ldots, h_{N-1}$ at the knot points are the optimizing parameters. The heights h_0 and h_N, and the gradients at R_1 and R_{N-1} are fixed during the optimization process; these correspond to the take-off and landing points, and the slopes of the first and last splines evaluated at R_1 and R_{N-1}. Vehicle constraints are enforced not only at the knot points but at intermediate points along the spline also. Mathematically the spline continuity and smoothness constraints are written as:

$$H_{i-1}(R_i) = H_i(R_i), \tag{12}$$

$$\frac{d}{dR}H_{i-1}(R)\bigg|_{R_i} = \frac{d}{dR}H_i(R)\bigg|_{R_i}, \tag{13}$$

$$\frac{d^2}{dR^2}H_{i-1}(R)\bigg|_{R_i} = \frac{d^2}{dR^2}H_i(R)\bigg|_{R_i}, \tag{14}$$

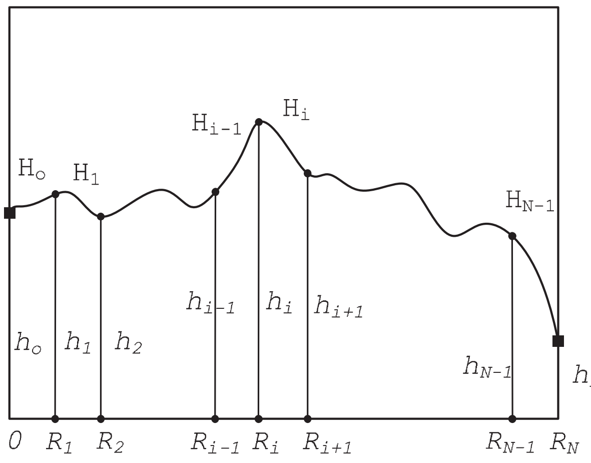

Fig. 9. Range partitioning for cubic splines.

Boundary conditions on the first and last splines are:

$$H_0(0) = h_0, \tag{15}$$

$$H_{N-1}(R_N) = h_N, \tag{16}$$

$$\frac{d}{dR}H_0(R)\bigg|_{R_1} = s_1, \tag{17}$$

$$\frac{d}{dR}H_{N-1}(R)\bigg|_{R_{N-1}} = s_{N-1}, \tag{18}$$

We thus have a total of $4N$ constraints (conditions) to compute the N splines; the splines have four coefficients each. The optimization problem is now to minimize (1) subject to constraints (2)–(5), (12)–(18). This is a nonlinear problem and different algorithms can be employed; we use sequential quadratic programming (SQP) for a solution to this problem. At each major iteration, an approximation of the Hessian of the Lagrangian function is made which is then used to generate a quadratic programming (QP) subproblem. The solution to this subproblem is then used to form a search direction for a line search procedure. Each QP subproblem minimizes a quadratic approximation of the Lagrangian function subject to a linear approximation of the constraints. The approximations are carried out using Taylor series expansion. The QP subproblem is to minimize the quadratic objective function:

$$F(\bar{h}) = \frac{1}{2}\bar{h}^T\mathcal{H}\bar{h} + y^T\bar{h}, \tag{19}$$

subject to linear equality and inequality constraints:

$$A\bar{h} = a, \tag{20}$$

$$B\bar{h} \geqslant b, \tag{21}$$

where $\bar{h} = [h_1, h_2, \ldots, h_{N-1}]^T$ denotes the optimizing parameter vector, y is the gradient vector, $\mathcal{H}$ is the positive definite Hessian matrix, and A, B, a, b are matrices of appropriate dimensions defining the constraints.[2] For details regarding the sequential quadratic programming algorithm, the reader is referred to [11,12]. The Optimal Spline algorithm proceeds as follows:

Step 1: Start with an initial guess of altitudes at the knot points.
Step 2: Invoke the SQP algorithm which will find the optimum parameters (heights) by repeatedly solving QP subproblems. As the algorithm iterates to converge to a solution, the optimizing parameters h_i are used to compute splines satisfying constraints (12)–(18) in each iteration cycle. These splines are evaluated at the knot points plus all intermediate points, the constraints (2)–(5) are then

[2] The superscript 'T' denotes the transpose of a vector or matrix.

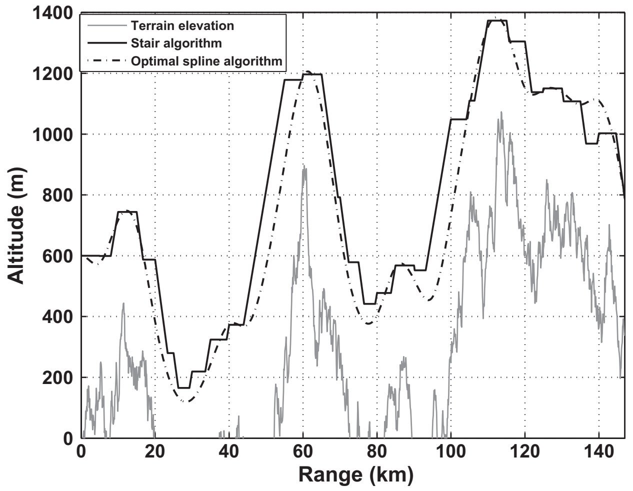

Fig. 10. Terrain-following trajectories for Stair and Optimal Spline algorithms (4 km section length).

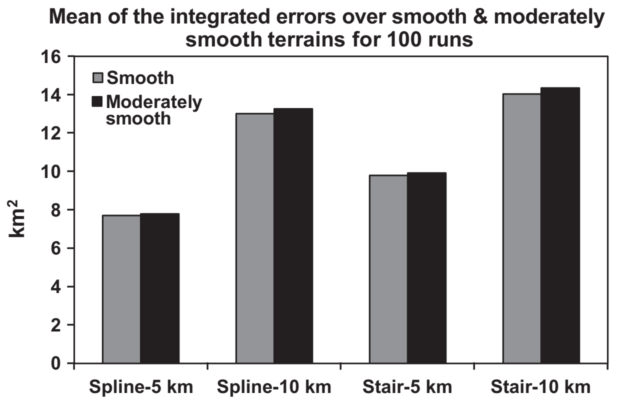

Fig. 11. Mean of the integrated errors for 100 runs of Stair and Optimal Spline algorithms.

evaluated and supplied back to the algorithm. Thus constraints (2)–(5) are satisfied not only at knot points, but at all points for which terrain elevation data is available.

3.4. Comparison of the algorithms

Here we present a comparison of the Stair and Optimal Spline algorithms. The Stair algorithm is simple and inexpensive, well-suited for online autonomous application. The Spline algorithm is based on setting up a constrained optimization problem and solving it through sequential quadratic programming. In order to compare the algorithms we generate sample terrain elevation data using terrain models. Gauss–Markov terrain models are used, as discussed in [13]. Different types of terrain can be modeled, namely smooth, moderately smooth, moderate, moderately steep and steep; the standard deviation of the terrain elevation increases as the terrain categories change from smooth to steep.

Both algorithms are tested for different terrain types, results are discussed below. Fig. 10 shows a sample terrain profile and the trajectories computed by the two algorithms. The spline trajectory is smooth as expected and is optimal in the sense of minimizing the objective function. The output of the Stair algorithm is also seen for a section length of 4 km; it follows the spline solution. Smaller section lengths will reduce the difference between the two solutions. Both algorithms satisfy their respective constraints. A number of tests run with different terrain profiles indicate that the Stair algorithm can be used for all types of terrains, whereas the Optimal Spline algorithm sometimes shows convergence problems for steep terrains. This is not uncommon for nonlinear programming based algorithms. For online autonomous use however, one cannot afford running into convergence issues.

We now present performance comparison of the two algorithms for smooth and moderately smooth terrains. One hundred terrain profiles are randomly generated for both types of terrain, and a performance index computed for the two algorithms. The index is defined as the integral over range of the difference between the actual terrain and the trajectory generated by the algorithm, i.e., $\int e dR$ (note that e is constrained to be positive semidefinite). Section (patch) lengths of 5 km and 10 km are used and the results shown in Fig. 11. The figure shows the mean of the integrated errors for 100 terrain profiles using the two algorithms. The results of the spline algorithm are shown in the left part of the figure for section lengths (ΔR) of 5 and 10 km. The right part of the figure shows the results of the Stair algorithm for same section lengths and terrain types. The error reduces for smaller section lengths as the planned path can follow the terrain underneath with greater precision. Smaller section lengths may therefore be used for steep terrains. It is seen that the two algorithms show comparable performance for same section (patch) lengths.

To summarize, the Stair algorithm is seen to offer reasonably good performance, it is simple and efficient and has no convergence issues. It can therefore be employed in an autonomous mode. The output it produces consists of altitude change points and heights as a function of range. The rate of climb and descent constraints are not violated. However the trajectory so generated has sharp corners and is not smooth. We will now propose a simple method to use the Stair algorithm's output and generate a smooth trajectory which is well-suited for tracking controllers.

3.5. Trajectory smoothing

The output of the Stair algorithm consists of straight and slanted lines, indicating level flight, a climb with the specified rate of climb, or a descent with the specified rate of descent. The lines intersect at node points creating sharp corners where a sudden change in the altitude profile takes place. If this profile is directly given as an input to the tracking controller, the derivative part of the controller will produce large control signals corresponding to these corner points, possibly saturating the actuators. Large spiky commands for actuators are not desirable. Furthermore at the descent stop nodes, the sudden change in slope (say from a descent rate of -0.1 to zero), will not be tracked exactly by the vehicle. A vehicle in flight with a certain rate of descent can only level off in finite time, and hence there will always be an undershoot (the actual altitude of the vehicle falling below the desired level) at descent stop nodes. Similarly ascent stop nodes will have overshoots, which is also undesirable. The undershoots are however more dangerous since the actual clearance of the vehicle from the underlying terrain will reduce, increasing chances of ground collision.

This problem of overshoot and undershoot can be largely eliminated by replacing the sharp corners of the reference trajectory at the ascent/descent stop nodes by exponentials of the form $1 - e^{-\tau r}$, where τ is the time constant of the exponential and r is the range. The range r is initialized to zero at the instant the switching takes place from the straight line slope to the exponential part, and increases thereafter. The straight line is continued with the prescribed rate of climb (or descent, whichever is applicable) until a small height difference Δh remains to the next altitude segment. Thereafter the exponential part starts and continues till the next segment's altitude is attained.[3] The constant τ and the height

[3] Although the exponential never attains the next segment's altitude exactly, but we consider it attained when it comes to within a given tolerance.

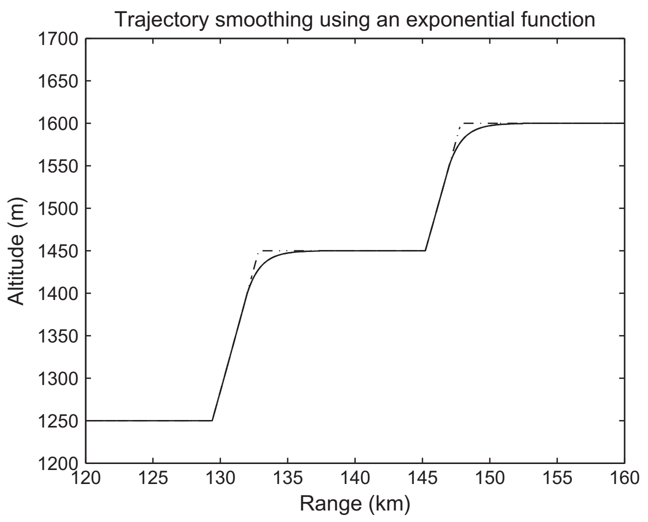

Fig. 12. Trajectory smoothing by using an exponential function.

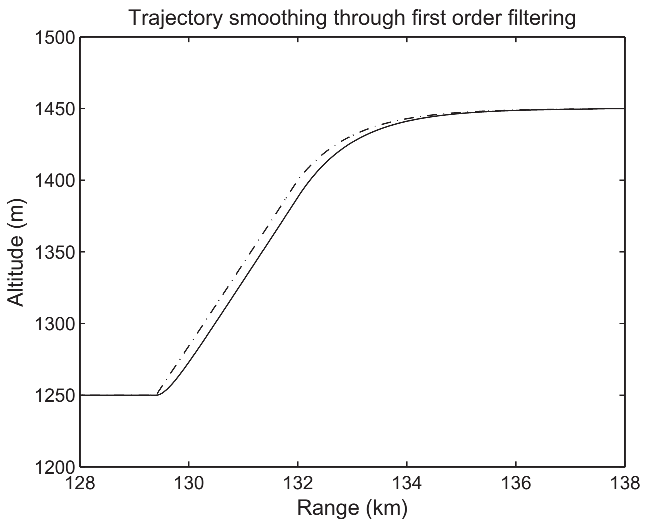

Fig. 13. Trajectory smoothing using a first order filter.

margin Δh depend on the dynamics of the particular vehicle and can be easily tuned through simulation. The exponential part is given by $\Delta h(1 - e^{-\tau r})$. Fig. 12 shows a section of the trajectory where the sharp corners at the ascent end points are replaced with the exponential function. The same technique can be used at the descent end points also. The final step in smoothing the reference trajectory is to pass it through a low-pass filter, e.g., of the form $\frac{a}{s+a}$, 's' being the Laplace variable. Fig. 13 shows the effect of passing the reference trajectory through a first order low-pass filter. The corners at the ascent/descent start points are also smoothed out. The order of the filter and the cutoff frequency can be selected based on the quality of smoothness required. In the authors' experience a first order filter suffices for most applications.

4. Tracking controller design

A simplified block diagram of the overall terrain-following guidance and control system is shown in Fig. 14. Terrain elevation data for the given mission track is generated first using the digital elevation map. The uncertainty in the vehicle's position in following the given mission is also considered and catered for. The terrain-following algorithm uses the elevation data and knowledge of the vehicle's constraints to compute a reference trajectory for the vehicle in the vertical plane. h_{ref} in the figure indicates the reference altitude for the vehicle to fly on. The actual height h of the vehicle is fed back and compared to h_{ref}, the altitude controller is designed to track the reference altitude and drive the error to zero. Control of altitude is achieved through control of the vehicle's pitch angle θ. The inner pitch control loop follows the pitch reference commands generated by the altitude controller. Body pitch-rate q is also fed back through a gain for stability augmentation.

4.1. The plant model

The roll and pitch angles of the vehicle are measured by a vertical gyro. This gyro has very fast dynamics as compared to the vehicle, and so this sensor is modeled as a simple gain in the feedback loop. The actuator dynamics are modeled by a fourth order transfer function as discussed in Section 5.1 below. The model of the UAV for controller design is taken as a linear approximation obtained at a cruise altitude of 2000 m and a speed of 50 m/s. The flight envelope however consists of an altitude range from 10 to 5000 m and a speed range from 35 to 60 m/s. A number of linear models are available across the flight envelope to test the robustness of the control system at different operating conditions. Note that the vehicle model has one input and three outputs (measured variables). The control input is the elevator deflection command to the actuators. The outputs are the pitch angle θ, body-axes pitch-rate q and the altitude h. The computation of the pitch-rate and its filtering is discussed below. Feedback of the pitch-rate is considered useful for providing damping to the short-period mode of the vehicle. The vertical gyro however does not provide measurement of angular rates, it only measures roll and pitch angles. A magnetic sensor on the vehicle senses the heading angle ψ, the dynamics of this sensor are also assumed fast and neglected. The height is sensed by the air data system (which measures the pressure altitude) onboard the vehicle. Body angular rates are computed from attitude angles as follows.

We define a navigation axis system $(X_nY_nZ_n)$ in which the Z_n axis is upwards along the local vertical, and the X_n and Y_n axes are in the local horizontal plane directed eastward and northward, respectively. The (true) heading, pitch and roll angles are denoted by ψ, θ and ϕ respectively, and these rotations occur in this specific order to align the navigation axes with the body axes. The heading rotation occurs first about the Z_n axis, followed by the pitch rotation about the X'_n axis, finally followed by the roll rotation about the Y''_n axis.[4] Resolving the Euler angle rates $\dot{\psi}$, $\dot{\theta}$ and $\dot{\phi}$ into body axes, we can solve for the body axes rates. Denoting the roll, pitch and yaw rates in the body axes by P, Q and R respectively, we have:

$$P = \dot{\psi}\sin\theta + \dot{\phi}, \tag{22}$$

$$Q = \dot{\theta}\cos\phi - \dot{\psi}\cos\theta\sin\phi, \tag{23}$$

$$R = \dot{\theta}\sin\phi + \dot{\psi}\cos\theta\cos\phi, \tag{24}$$

It may be noted that the attitude sensors measure the body angles and not their rates. The derivatives of the attitude angles are computed by fitting a least-squares line to n consecutive attitude measurements. We have chosen $n = 4$ here as we have seen it to yield good results; the derivative thus computed has enough noise smoothing and an acceptably small time delay. However other choices for n can be made depending on the application and on the sampling time of the attitude sensor measurements. If for example the pitch angle measurements from the vertical gyro are denoted by θ_0, θ_1, θ_2 and θ_3, where θ_0 corresponds to the current

[4] Note that the primed axes are the intermediate axes obtained while going from the navigation to the body frame through the ψ, θ, ϕ rotation angles.

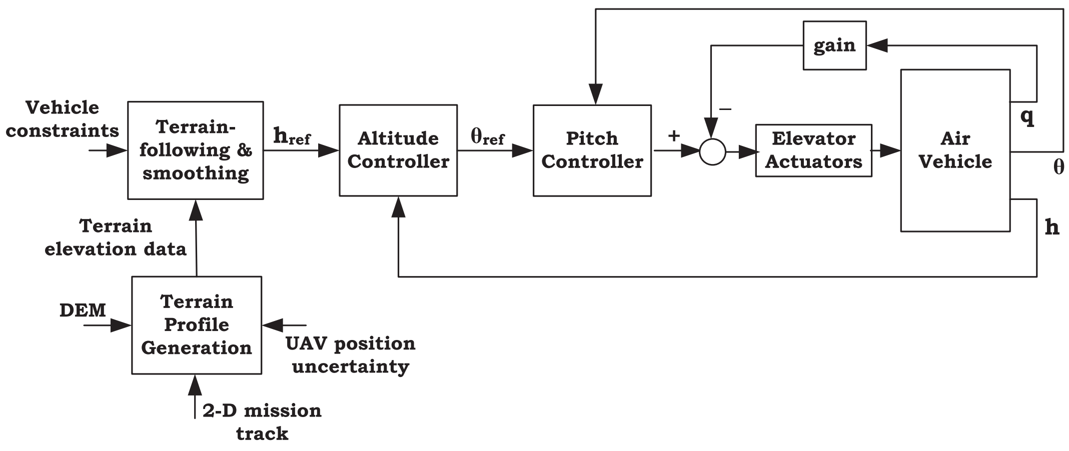

Fig. 14. Block diagram of the overall terrain-following system.

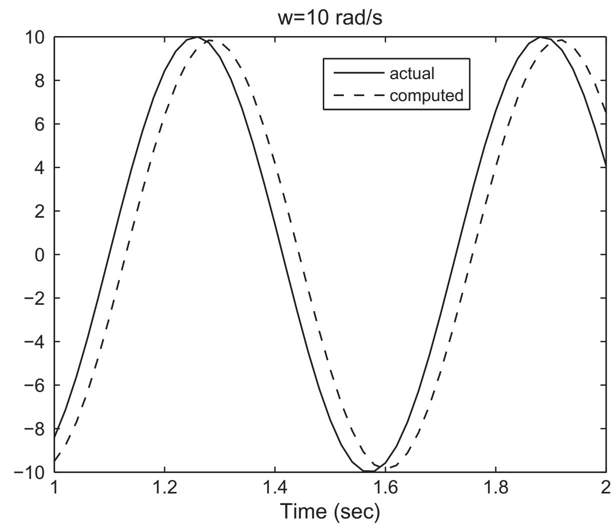

Fig. 15. Delay introduced because of the four-point derivative approximation.

measurement and θ_3 corresponds to the measurement taken three samples previously, then the slope of the least-squares line for these points approximates the derivative [14]:

$$\dot{\theta} \approx \frac{3\theta_0 + \theta_1 - \theta_2 - 3\theta_3}{10\Delta t}, \tag{25}$$

Here Δt is the time interval at which the angle θ is sampled (20 ms for our case). This approximation of the derivative by slope of a four-point least-squares line gives a good compromise between sensitivity to measurement noise and the delay introduced into the estimation of $\dot{\theta}$. Fig. 15 shows the delay between the actual and computed derivatives for a pure sinusoid of frequency 10 rad/s, which is about 25 ms. A test flight was carried out in which a rate sensor was specially installed on the vehicle for validation purposes only; this sensor is not available for feedback control otherwise. The comparison between the measured and computed pitch-rates shown in Fig. 16 indicates the validity of the approximation. It is thus concluded that the approximation (25) can be used for Euler angle rate estimation from attitude measurements, and thereafter (22)–(24) can be employed for transforming the Euler angle rates into body-axes angular rates which are used for feedback control. This method of estimating body rates and using them for feedback is seen to work well for normal flight maneuvers.

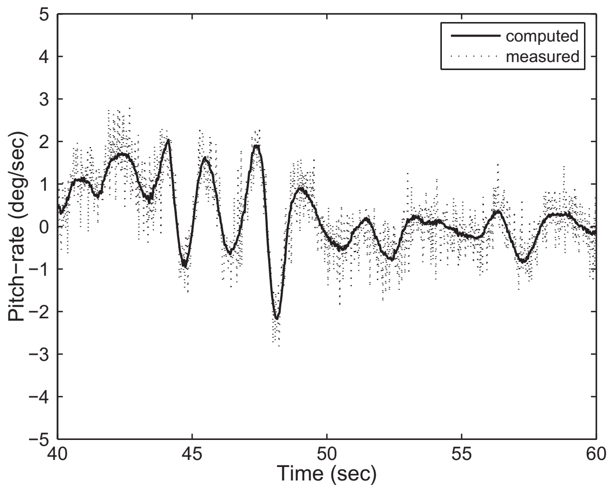

Fig. 16. Measured and computed pitch-rates during test flight.

4.2. H_∞ loop-shaping design procedure

For controller design, we shall use the H_∞ loop-shaping design procedure proposed in [15]. The procedure is intuitive as it is based on the generalization of classical loop-shaping ideas. The open-loop plant, once given the desired loop-shape, is robustly stabilized against coprime factor uncertainty. The resulting controller has been shown to enjoy some favourable properties, such as no pole-zero cancellation occurs in the closed-loop system (except for a certain special class of plants), see [16]. In addition, the controllers thus designed have been successful in various applications; examples are those described in [17–20].

In practical design applications, the performance specifications are first translated into the frequency domain, and the open-loop plant's frequency response is given the desired shape. This is achieved by augmentation of the nominal plant model G by a suitable compensator (or weighting function) W. The shaped plant $G_s = GW$ is then robustly stabilized against coprime factor uncertainty, and the controller K_∞ thus obtained is cascaded with the weight W to obtain the final controller $K = WK_\infty$. It can be shown that the controller does not significantly alter the specified loop-shape provided a sufficiently small value of the cost γ is achieved, for details refer to [20].

We now outline a design procedure for designing controllers based on open-loop shaping and robust stabilization of the normalized coprime factors of the plant. The procedure consists of the following main steps:

1. Plot the frequency response of the open-loop plant $G(s)$. Based on the desired bandwidth of the feedback loop, decide upon a crossover frequency for the open-loop transfer function.
2. Select a suitable weighting function $W(s)$ to give the plant a desired open-loop shape. The crossover frequency should be adjusted close to the desired bandwidth. Also $W(s)$ should be designed to give a roll-off of -1 (or -20 dB/decade) at the crossover frequency. Build the shaped plant $G_s(s) = G(s)W(s)$ and calculate the optimal cost γ_{opt} [15]. A high value (typically >10) of γ_{opt} indicates that the specified loop-shape is inconsistent with robust stability; in such a case the weighting function W should be modified.
3. Use a slightly sub-optimal value of γ and compute the corresponding controller K_∞.
4. Cascade the controller with the weight W to compute the final controller $K = WK_\infty$. Controller order reduction may be performed if desired.
5. Form the closed-loop and check the appropriate performance and robustness measures against the given specification.

4.3. Controller design

A simplified block diagram of the stabilization and tracking control system is shown in Fig. 17. The sensed variables of the air vehicle are the pitch (θ) and roll (ϕ) angles measured by the vertical gyro, the heading angle (ψ) measured by the magnetometer, and the height (h) sensed by a barometric air data system. The pitch-rate estimation block uses the three attitude angles and generates an estimate of the body pitch-rate (q) which is fed back through the gain k_d to the elevators for improving damping of the short-period mode of the vehicle. The pitch controller $K_p = W_pK_{p\infty}$ provides pitch angle tracking control so that the actual pitch angle follows the reference angle θ_{ref} generated by the outer loop. W_p is the weighting function for design of the pitch control loop, and $K_{p\infty}$ is the H_∞ controller as discussed above (Section 4.2). The outer loop is the altitude tracking loop; the altitude controller $K_h = W_hK_{h\infty}$ generates the pitch reference command in order to make the actual altitude h track the reference altitude h_{ref} computed by the terrain-following algorithm.

4.3.1. Pitch controller design

The model of the UAV for controller design is taken as a linear approximation at a cruise altitude of 2000 m and a speed of 50 m/s. The actuator dynamics are modeled by a fourth order transfer function (Section 5.1). The delay τ_d due to the pitch-rate estimation algorithm is modeled by a second order Padé approximant:

$$e^{-\tau_d s} \cong \frac{1 - \tau_d s/2 + (\tau_d s)^2/12}{1 + \tau_d s/2 + (\tau_d s)^2/12}.$$

Now the pitch-rate feedback gain k_d is adjusted to get a well-damped short-period mode for the plant. A damping ratio of 0.55 for the short-period mode is considered satisfactory. We now proceed with the design of the controller following the procedure given in Section 4.2. We denote the plant transfer function with the pitch-rate feedback in place by $G_p(s)$; the actuator transfer function is also included.

1. It is desired to have a bandwidth of approximately 8–10 rad/s for the pitch control loop. The crossover frequency of the open-loop transfer function $G_p(s)W_p(s)$ should therefore be adjusted in this range.
2. The weighting function $W_p(s)$ is chosen as follows:

$$W_p(s) = k\frac{(s+1)(s+3)}{(s+70)(s+0.0225)}E(s),$$

where k is a gain to adjust the crossover frequency at the desired value, and $E(s)$ denotes an elliptic filter transfer function. Elliptic filters are low-pass filters that are well-suited for filtering of resonant lightly damped modes of a flexible structure. Flight structures have to be light weight and therefore cannot be very rigid, also flutter modes have to be guarded against. Furthermore sensor noise is always present, and therefore low-pass filters in the loop are useful. The phase lag effect of these filters at the crossover frequency however, has to be considered carefully. Elliptic filters have been found to be particularly useful for such applications [21]. The filter chosen here is a third order filter with a stopband attenuation of 40 dB and a cutoff frequency of 60 rad/s. The term $\frac{s+1}{s+70}$ is a lead compensator to give adequate phase lead at the crossover frequency, whereas the other term $\frac{s+3}{s+0.0225}$ is a lag filter to boost the low-frequency gain for good disturbance rejection and command following at low frequencies. The frequency response of the original plant $G_p(s)$ and the shaped plant $G_p(s)W_p(s)$ is shown in Fig. 18, the crossover frequency is adjusted to 7 rad/s. The optimal cost γ_{opt} is computed to be 1.88 which is deemed acceptable.
3. A slightly suboptimal γ of 1.98 is selected and the controller $K_{p\infty}$ computed.
4. The complete controller $K_p = W_pK_{p\infty}$ is now formed. The controller has 18 states, it is reduced to 11 states using optimal Hankel-norm approximation [22]. The reduced controller is discretized using the bilinear (Tustin's) method for onboard implementation.
5. The magnitude plot of the sensitivity function is shown in Fig. 19. The figure shows adequate disturbance rejection on the pitch angle for frequencies less than 3 rad/s. The flight envelope is shown in Fig. 20, which ranges from ground level to 5000 m in altitude, and from 34 m/s to 61 m/s in forward speed. A number of linearized models are taken along the periphery of the envelope as shown in the figure. Step responses for these models using the designed controller are given in Fig. 21. The controller performs acceptably well throughout the envelope, the robustness of the design is thus illustrated.

4.3.2. Altitude controller design

The closed-loop transfer function of the pitch control system discussed above forms the plant for design of the altitude controller.

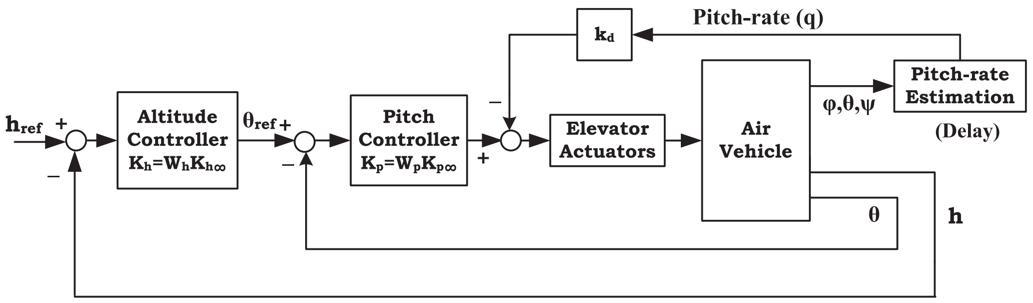

Fig. 17. Block diagram of the tracking control system.

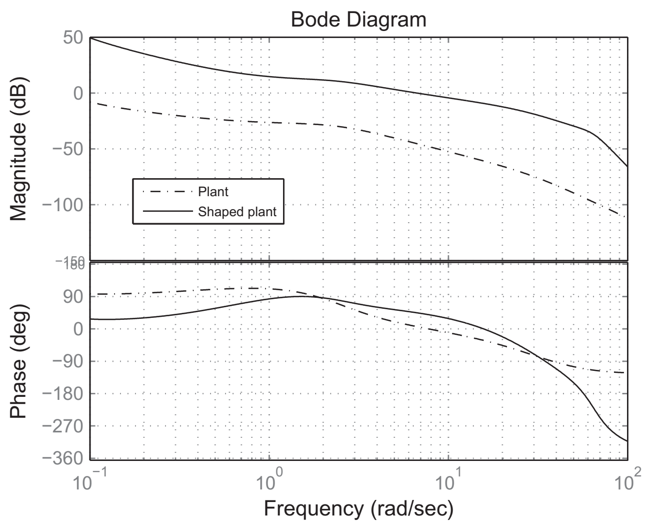

Fig. 18. Frequency response of the plant $G_p(s)$ and the shaped plant $G_p(s)W_p(s)$.

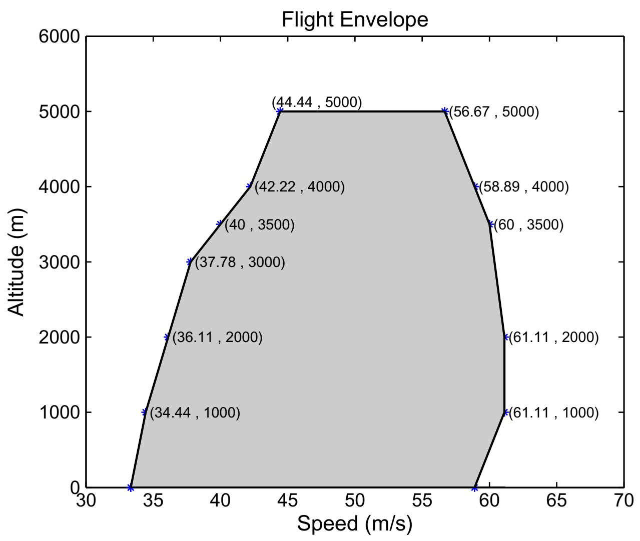

Fig. 20. The flight envelope of the vehicle.

The input to this transfer function is the commanded pitch angle θ_{ref} and the output is the height of the vehicle. The height is sensed by the barometric air data system, the lag in the sensor and tubing is modeled as a first order transfer function of the form $\frac{1}{\alpha s+1}$. The altitude controller will control the altitude of the vehicle by commanding the inner pitch control loop. The bandwidth of the outer altitude loop is kept a factor of about 10 lower than the pitch loop, this spectral separation ensures no unstable dynamic interaction between the two loops. The design procedure of Section 4.2 is again followed, here we summarize the main points. The weighting function $W_h(s)$ selected is $k\frac{s+0.25}{s+3}\frac{s+0.1}{s+0.0001}$, where the zeros are designed to give phase lead at the crossover frequency, and the pole at -0.0001 boosts the low-frequency gain for accurate altitude tracking in the steady state. The gain k adjusts the crossover frequency to 0.6 rad/s.

Fig. 22 shows the sensitivity function for the altitude controller and Fig. 23 shows the response of the altitude controller for a step demand of 100 m in altitude for various plants in the envelope. The figure shows good robustness properties of the designed altitude control autopilot.

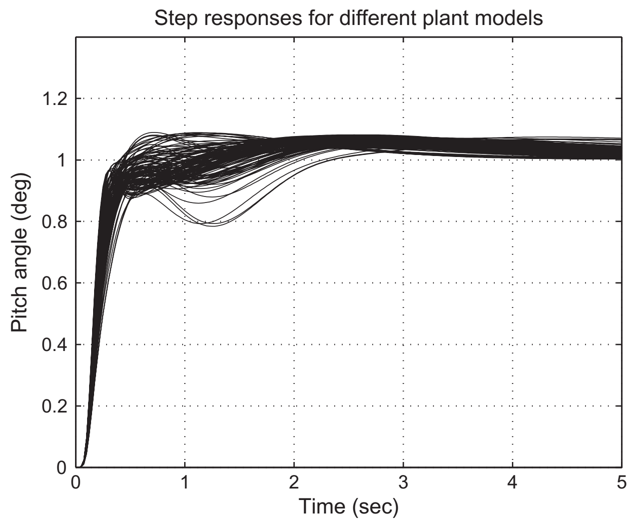

Fig. 21. Step responses for different plant models across the flight envelope.

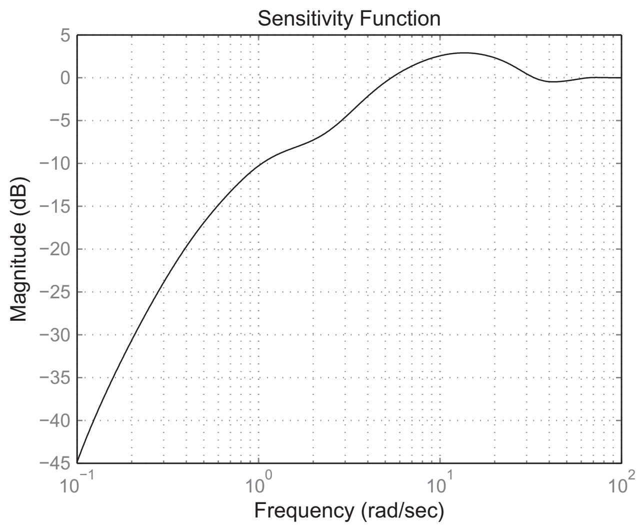

Fig. 19. Magnitude plot of the sensitivity function $1/(1+G_pK_p)$ versus frequency.

5. Flight control hardware

5.1. Actuator design

The actuators employed for deflection of control surfaces of the UAV (the elevators) have to be light weight, compact, reliable, low cost and capable of long continuous operation to cater for the flight endurance of the vehicle. These are mounted on the surface (skin) of the vehicle and coupled directly to the respective control surface. Here we present the design of the actuator and its main characteristics. The actuator is a miniaturized L-shaped 5 N m rotary electromechanical device. It employs a brushless DC servo-motor, a rotary reduction harmonic drive and a bevel gear mechanism.

5.1.1. General construction

The actuator converts the high speed, low torque rotary energy of the drive motor to a high torque, lower speed rotary position output. This output is sensed and fed back through a potentiometer. The major components of the actuator include a permanent magnet brushless DC (BLDC) servo-motor, a bevel gear train, a harmonic drive, a rotary potentiometer assembly, an output shaft and the main housing (Fig. 24).

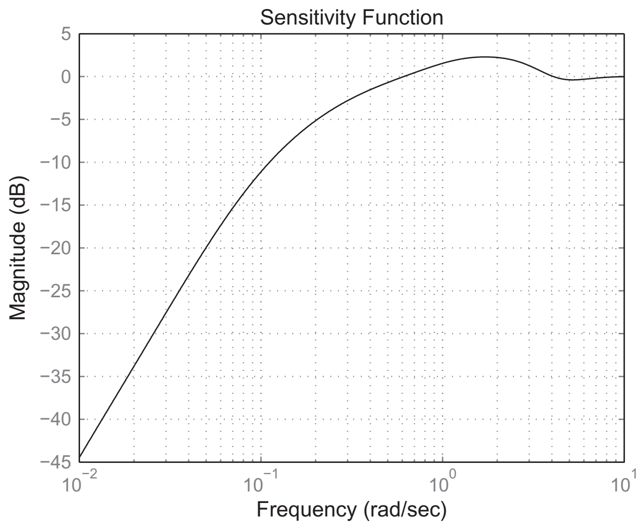

Fig. 22. Magnitude plot of the sensitivity function $1/(1 + G_hK_h)$ versus frequency.

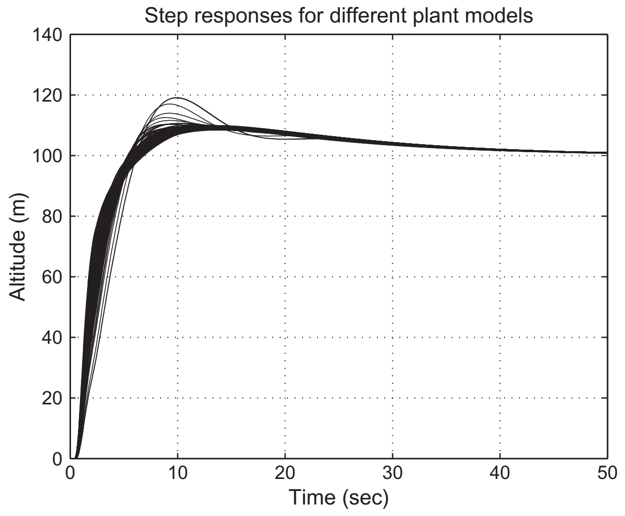

Fig. 23. Response of the altitude controller for a step of 100 m for different plant models in the flight envelope.

The working principle of the actuator system is illustrated by the block diagram shown in Fig. 25. The electronic servo-controller receives the reference position command from the central flight computer of the vehicle which runs all control and guidance algorithms in real time. The actual position of the actuator is sensed by the potentiometer, and compared with the commanded reference. The position error thus formed is filtered and compensated by the signal synthesizing circuit, which employs a proportional–integral–derivative (PID) control logic. It is subsequently converted to a suitable pulse-width-modulated (PMW) signal, amplified, and applied to the motor. The output shaft of the motor is coupled to the stage-1 decelerator, which is a bevel gear mechanism providing a reduction of about three. The bevel stage drives the wave generator of stage-2 decelerator, which is a harmonic drive. The output of the harmonic drive is coupled to the output shaft which has the feedback potentiometer mounted on it, thus closing the servo loop. The output deflection of the actuator is ±25°. The speed at rated load is greater than 70 deg/s and the bandwidth is around 5 Hz.

The motor chosen for the actuator is a three-phase, permanent magnet brushless DC motor. Such motors have high torque-to-weight ratios, good dynamic response, are compact and low-noise. They can operate at high speeds with moderately linear output (torque versus speed) curves. The selected motor has a rated speed of 1000 rpm and a rated torque of 0.023 N m.

5.1.2. The gear train

The gears used for speed reduction and torque transmission are required to be precise to avoid oscillations and achieve the desired position accuracy with low backlash. Here we use a two-stage decelerator mechanism. Stage-1 is a bevel gear set with a reduction ratio of 2.8, and stage-2 is a harmonic drive system. The flex spline of the harmonic drive has 202 teeth and the circular spline 204 teeth, giving a reduction ratio of 101. The overall gear ratio n achieved is 282; both gear trains are made from alloy steel. The harmonic drive has the advantage of no backlash, high gear ratio and high torque transfer capability within a small volume, thus yielding a flat, compact surface mount actuator.

5.1.3. Actuator dynamic model

The electrical equation of the actuator motor is given by:

$$e_i = i_aR_a + L_a\frac{d}{dt}(i_a) + k_a\dot{\vartheta}_m,$$

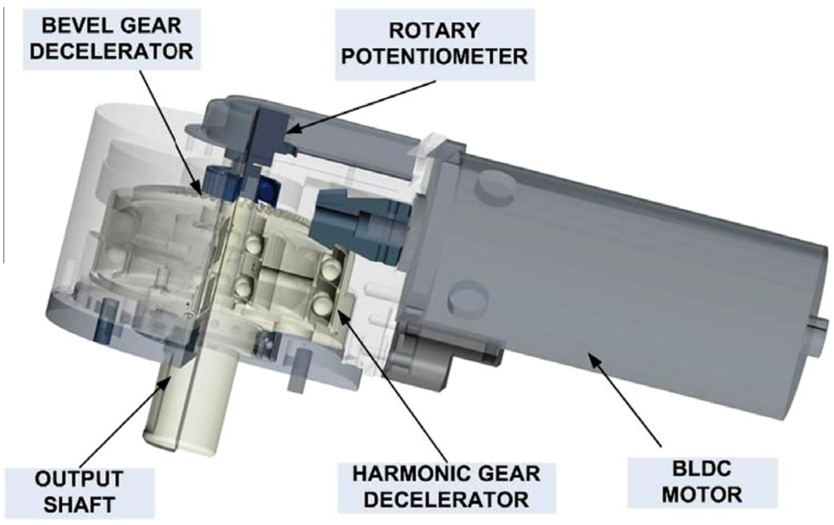

Fig. 24. General composition of the actuator.

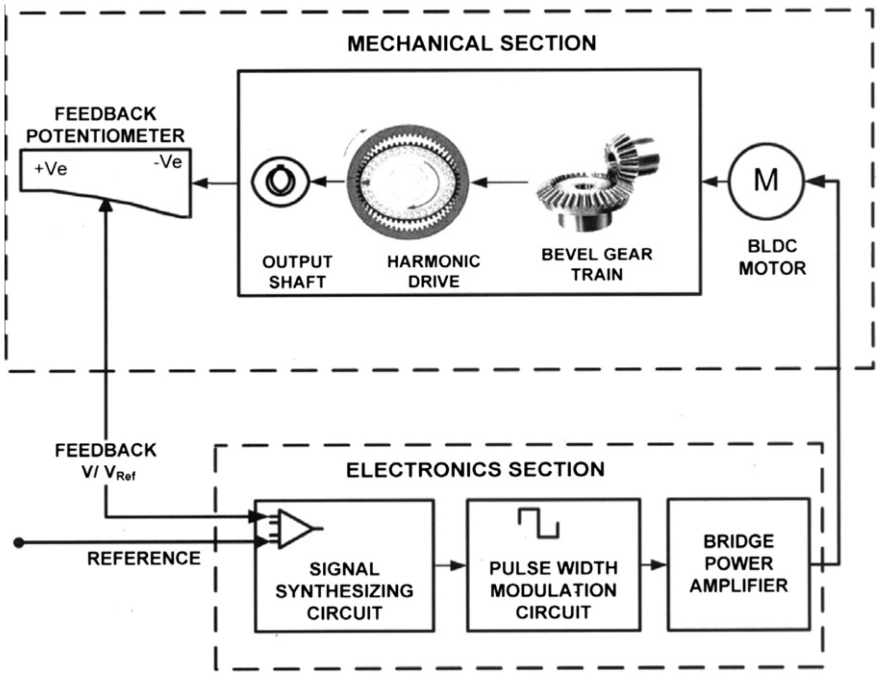

Fig. 25. Working principle of the actuator system.

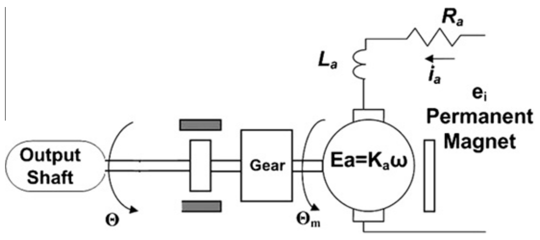

Fig. 26. A simplified model of the actuator system.

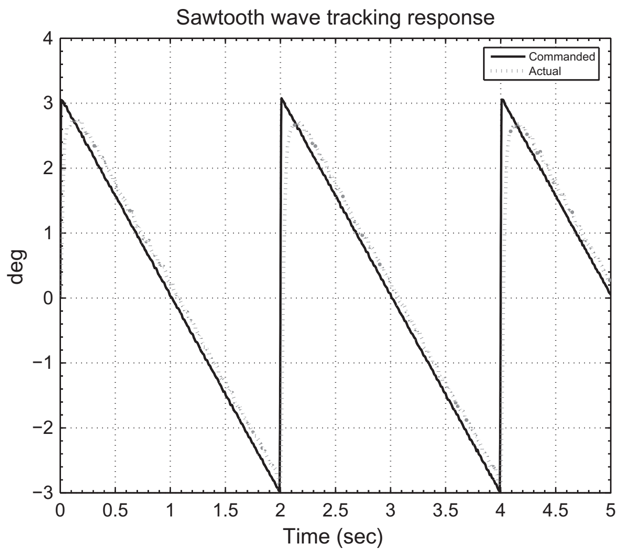

Fig. 27. Experimental tracking response of the actuator for a sawtooth wave reference command.

where e_i is the control voltage applied to the motor, i_a is the motor armature current, and R_a and L_a are the motor armature resistance and winding inductance, respectively (see Fig. 26). k_a is the motor

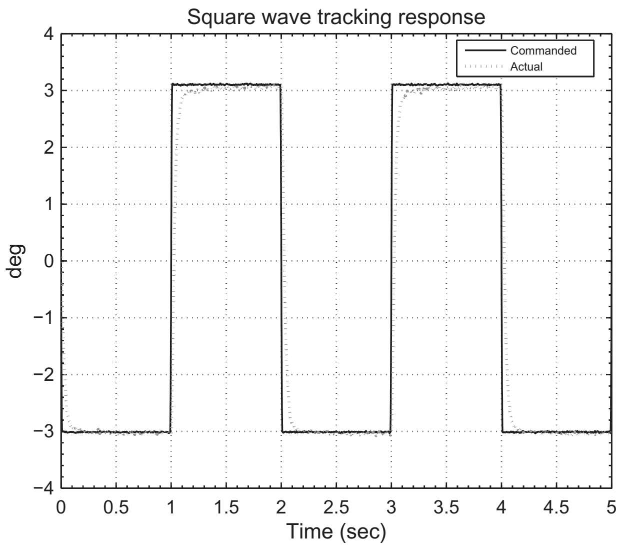

Fig. 28. Experimental tracking response of the actuator for a square wave reference command.

torque constant and ϑ_m represents the motor shaft rotation angle. The rotational equation of motion is:

$$T_m = J_m\ddot{\vartheta}_m + B_m\dot{\vartheta}_m,$$

where T_m is the motor torque, and J_m and B_m are the moment of inertia of the motor armature plus load, and the viscous damping coefficient of the rotating assembly. The motor torque T_m is directly proportional to the armature current: $T_m = k_a i_a$. The rotation of the actuator output shaft ϑ is related to the motor shaft rotation by the overall gear ratio or deceleration constant n:$\vartheta = \vartheta_m/n$.

The nonlinearities in the actuator arise primarily from the voltage and current limitation in the control and drive circuits, and the friction of the moving parts. The friction torque can be modeled as $T_f = k_a i_0 \mathrm{sgn}(\dot{\vartheta}_m)$, where $\mathrm{sgn}(\cdot)$ denotes the *'signum'* or *'sign'* function

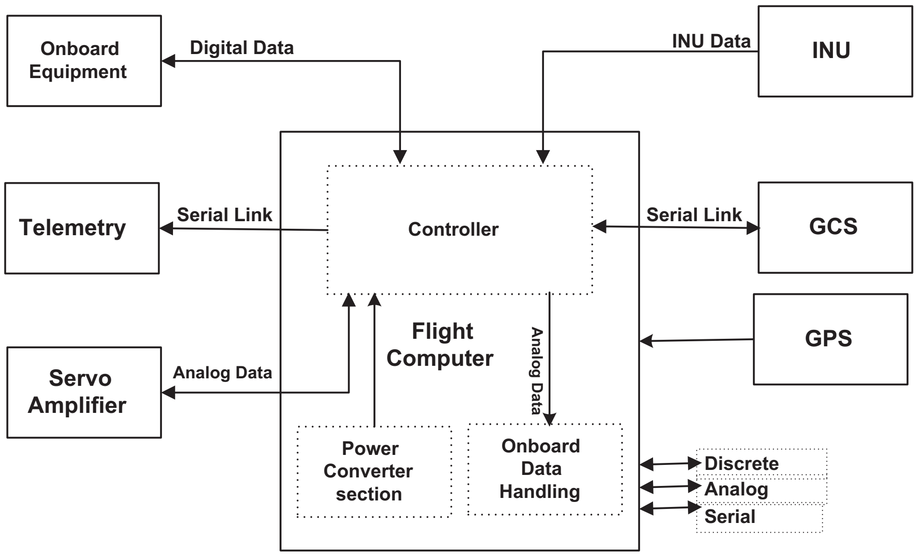

Fig. 29. Block diagram of the flight control computer.

and i_0 is the motor no-load current. Neglecting the nonlinearities we can, using the above relations, derive an open-loop transfer function of the actuator dynamics relating the output shaft angle ϑ to the control voltage e_i as:

$$\frac{\vartheta}{e_i} = \frac{k_a/n}{(L_a s + R_a)(J_m s^2 + B_m s) + k_a s^2}.$$

The actuator has a local PID controller (implemented in the signal synthesizing/control circuit) with parameters k_p, k_I, k_D, so that the closed-loop actuator transfer function relating the actual actuator angular displacement ϑ to the commanded displacement ϑ_{ref} is:

$$\frac{\vartheta}{\vartheta_{ref}} = \frac{k_a k_D s^2 + k_a k_p s + k_a k_I}{L_a J_m n s^4 + n(J_m R_a + L_a B_m)s^3 + (R_a B_m n + k_a^2 + k_a k_D)s^2 + k_a k_p s + k_a k_I}. \quad (26)$$

where k_p, k_I and k_D are the proportional, integral and derivative gains of the PID controller respectively, used for the actuator servo loop compensation.

5.1.4. Servo control performance

We now present the dynamic performance of the actuator by carrying out experiments of the prototype unit manufactured. Tests are conducted to check performance parameters such as dead-zone, linearity, maximum angular deflection, and bandwidth. The tracking performance of the prototype actuator is shown in Fig. 27 for a sawtooth wave reference command. Fig. 28 shows the tracking performance for a square wave reference input. The actuator is seen to exhibit good tracking performance with a fast response time, low overshoot, and small error. The said actuator exhibits good closed-loop dynamic performance and is found to be suitable for flight control of the vehicle.

5.2. Flight computer hardware

The flight control computer is an intelligent processing and control unit. The architectural block diagram is shown in Fig. 29. The flight computer interfaces with the Ground Control Station (GCS) via the wireless data-link, the inertial navigation unit (INU), the GPS receiver, the actuator servo-amplifier unit, and other onboard equipment such as the air data system, the magnetometer, onboard cameras, etc. It also houses a test/telemetry port, analog and discrete inputs/outputs and serial communication ports. The central processing unit is an Am5$_{\times}$86 double-precision floating point processor with 1 MB of static RAM[5] and 1 MB of flash memory. The printed circuit boards have a 12-layered design. Special features include onboard current and voltage sensing for fault diagnosis, EMI[6] filters on power lines, and design considerations for low-noise high integrity signal transfer.

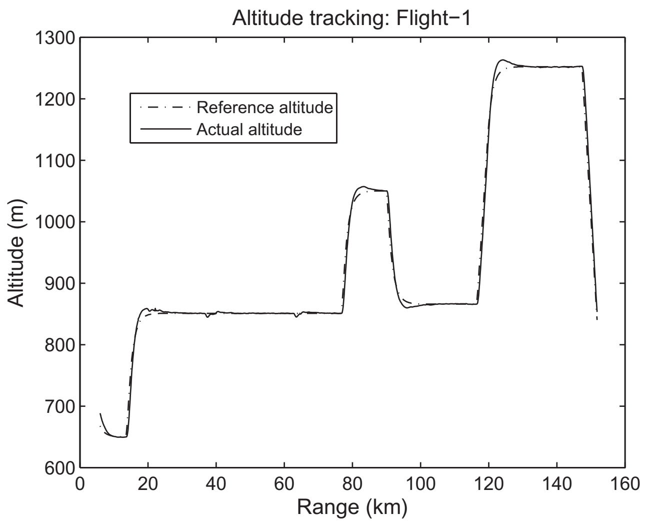

Fig. 30. Flight-1: commanded and actual altitudes.

6. Flight test results

The vehicle used as a test-bed for demonstrating the efficacy of the proposed algorithms is shown in Fig. 1. It has a swept-back trapezoidal wing, a set of (forward) canards for pitch control, and ailerons on the main wing for roll control. Rudders are mounted on the vertical tails. The vehicle falls in the low-to-medium cate-

[5] Random Access Memory.
[6] Electromagnetic interference.

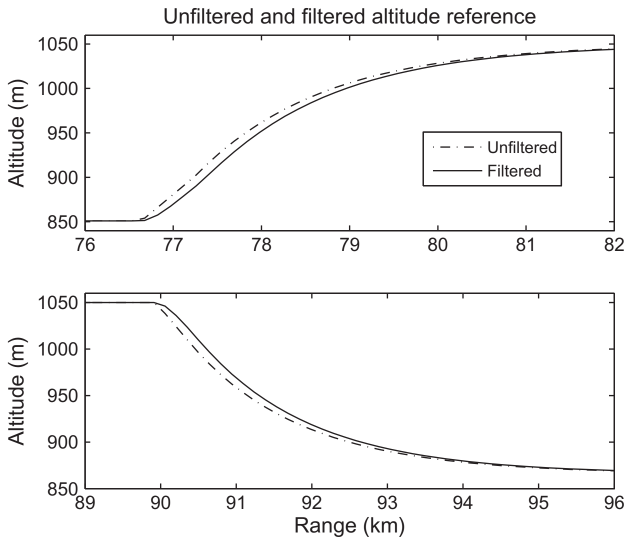

Fig. 31. Filtered and unfiltered altitude command signals.

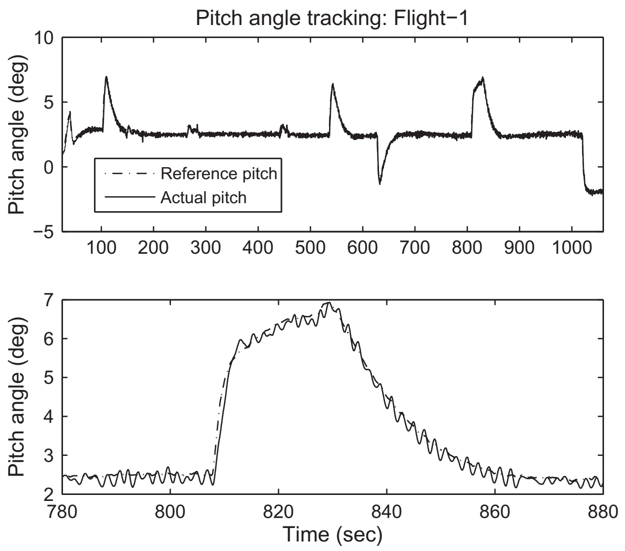

Fig. 32. Flight-1: commanded and actual pitch angles.

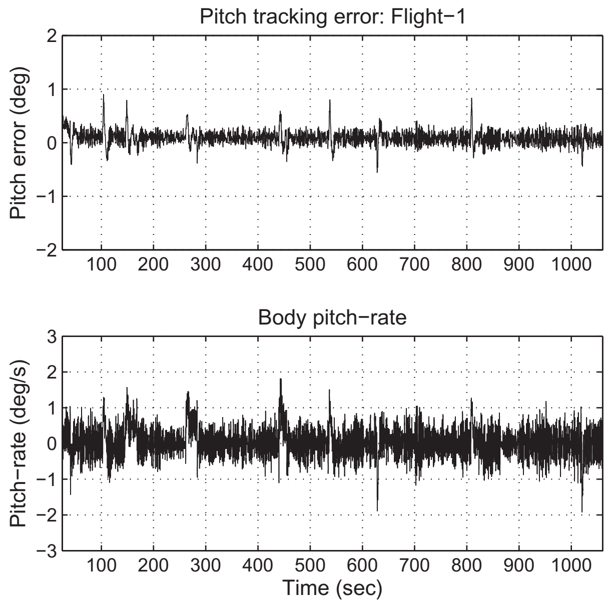

Fig. 33. Flight-1: pitch angle error and body pitch-rate.

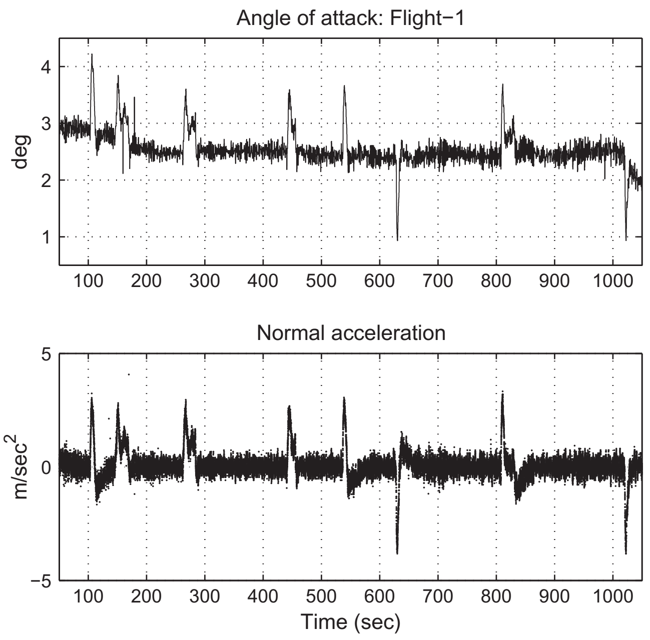

Fig. 34. Flight-1: angle of attack and normal acceleration.

gory of UAVs in terms of its ceiling and endurance parameters. The flight envelope covers an altitude range from 10 to 5000 m and a speed range from 35 to 60 m/s. A piston engine drives the propeller in a push-configuration and provides a maximum power of 100 hp. The wing span is 6.5 m, the aspect ratio is around 7, and the aerodynamic design provides a maximum lift-to-drag ratio in excess of 10. The maximum take-off weight is 450 kg; the vehicle can stay airborne up to 12 h carrying high performance surveillance equipment on board.

Several test flights of the air vehicle were conducted over a period of one year. These were designed to test different aspects of the aircraft. Results of two test flights (referred to as flight-1 and flight-2 henceforth) are presented below. Figs. 30–34 show the results of flight-1 and Figs. 35–38 show the results of flight-2. Fig. 30 shows the reference altitude h_{ref} and the actual vehicle altitude as a function of range. The slight drop in the vehicle altitude at 40 km and 65 km range is due to turn maneuvers of the vehicle at these instances. As the vehicle banks to turn, the lift vector tilts sideways, reducing the upward component, thus causing the altitude to drop. The altitude controller senses this drop and raises the pitch angle demand θ_{ref}. The increase in pitch angle causes the nose of the vehicle to be raised, increasing the angle of attack, and thus generating the extra lift required to control the altitude at the desired level. The altitude tracking is seen to be good with the actual altitude closely tracking the reference altitude.

Fig. 31 shows the filtered and unfiltered altitude command signals. The filtered altitude command is shown in solid line, which is obtained by passing the unfiltered version through a first order filter, as discussed in Section 3.5. The filtered signal becomes the reference altitude that is tracked by the altitude controller. The tracking performance of the pitch controller is shown in Fig. 32. The figure shows the reference and actual pitch angles, with the

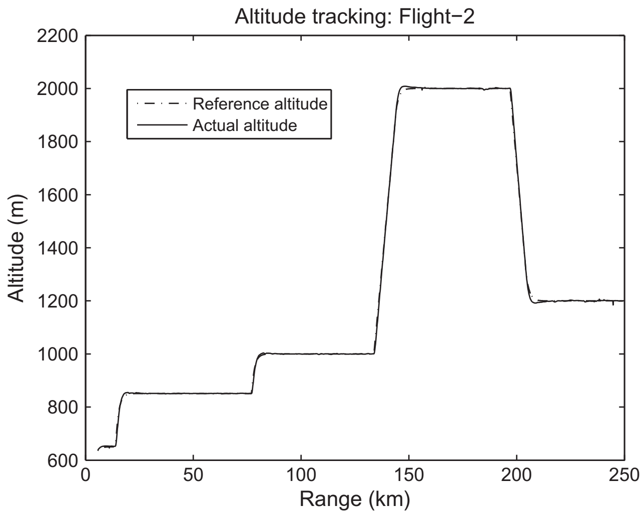

Fig. 35. Flight-2: commanded and actual altitudes.

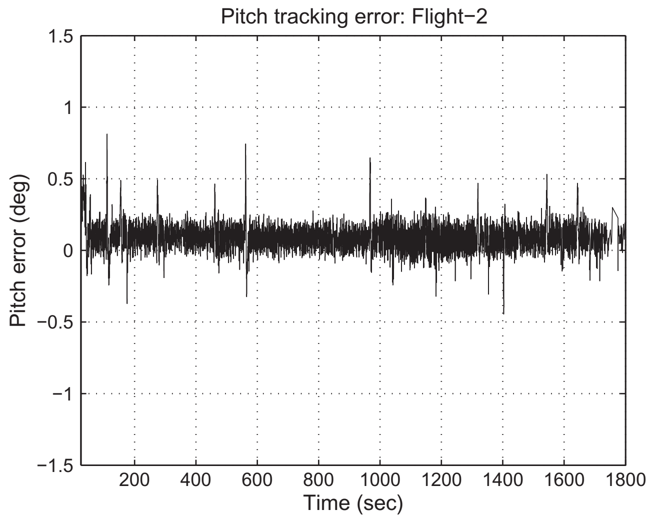

Fig. 37. Flight-2: pitch angle error.

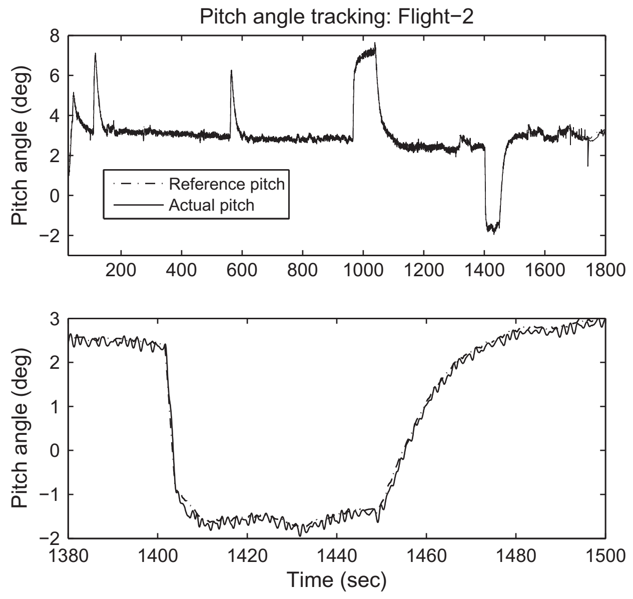

Fig. 36. Flight-2: commanded and actual pitch angles.

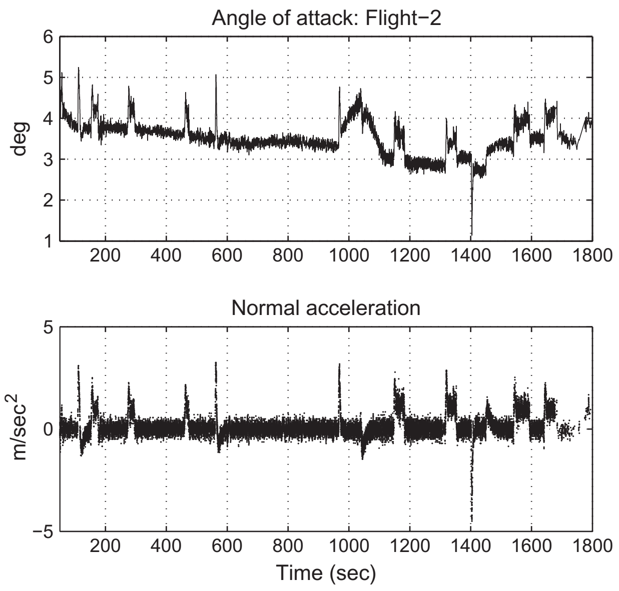

Fig. 38. Flight-2: angle of attack and normal acceleration.

lower half of the figure showing a zoomed view of the data. Excellent command following of the pitch angle is observed. It is seen that an increase in h_{ref} causes the altitude controller to generate a larger θ_{ref}, causing the vehicle to climb. Control of altitude is achieved through control of the pitch angle, and good pitch angle tracking is seen to give tight altitude control. Fig. 33 shows the error in the pitch angle (i.e., the difference between the reference and actual pitch angles). The error stays within ±1° demonstrating the pitch tracking performance. The body pitch-rate q (Section 4.1) is also seen in the lower half of the figure. The angle of attack and the normal acceleration of the vehicle are displayed in Fig. 34. The angle of attack sensor is installed for test purposes only, it is not available as a standard sensor for feedback control. The correlation between the climb/descent maneuvers, the pitch angle, the angle of attack and normal acceleration can be clearly seen from the figures. During the climb phase the pitch angle is increased which results in an increase in the angle of attack and the normal acceleration of the vehicle. The larger angle of attack generates the higher lift required to climb. It may be noted that a separate speed control system is also in operation throughout, which throttles the engine up and down to maintain the speed of the vehicle at a given level.

Figs. 35 and 36 show the altitude and pitch angle tracking respectively, for the second flight. The tracking is seen to be tight, the errors being small. The lower half of Fig. 36 gives a zoomed view of the pitch response. There is a continuous climb from 1000 m to 2000 m with a slope of 0.092 with almost perfect tracking. The vehicle climbs with a (nose-up) pitch angle of about 7.5°. Fig. 37 plots the pitch error which stays within ±1°. Fig. 38 shows the angle of attack and normal acceleration. It is seen that the vehicle climbs with an angle of attack of 4–5°.

7. Conclusion

An integrated methodology for the design of an autonomous terrain-following system is developed. This is done keeping in view the practical aspects of the problem, and implementation on a real-time platform. A method for the generation of terrain elevation data is given which takes into consideration the growing position uncertainty of the vehicle with range. An algorithm[7] for terrain-following is next developed which is well-suited for online computation in autonomous flight. The entire route is broken down into sections of given length, and the height of each section is computed. The climb/descend control effort can be adjusted by varying the section lengths. Altitude change points and heights are computed that do not violate practical vehicle constraints, while providing the desired terrain clearance. The performance of this algorithm is found to compare favorably to an Optimal Spline algorithm. The design of the tracking control system is next presented. A practical method for pitch-rate estimation is given. Robust pitch angle and altitude tracking controllers are designed, which maximize robustness against system parametric uncertainty. The controllers are seen to perform well across the flight envelope of the vehicle. The design of an electromechanical servo actuator is discussed which is used to actuate the control surfaces (elevators) of the vehicle during flight. Finally flight test results are presented and discussed, these demonstrate the overall performance of the terrain-following guidance and control system. Plans are underway to carry out more tests on a smaller vehicle with shorter and possibly more flexible mission requirements. Although simulation results indicate no limitation of the proposed algorithms, however further testing on a range of mission scenarios is planned to generate more experimental data.

[7] The basic algorithm is called the Stair algorithm; this is then augmented with a trajectory smoothing algorithm employing an exponential function and a low-pass filter.

References

[1] Funk J. Optimal-path precision terrain-following system. J Aircraft 1977;14(2):128–34.
[2] Lu P, Pierson B. Optimal aircraft terrain-following analysis and trajectory generation. J Guid Control Dyn 1995;18(3):555–60.
[3] Lu P, Pierson B. Aircraft terrain following based on a nonlinear continuous predictive control approach. J Guid Control Dyn 1995;18(4):817–23.
[4] Rippel E, Bar-Gill A, Shimkin N. Fast graph-search algorithms for general-aviation flight trajectory generation. J Guid Control Dyn 2005;28(4):801–11.
[5] Asseo S. Terrain following/terrain avoidance path optimization using the method of steepest descent. In: Proceedings of the IEEE 1988 national aerospace and electronics conference (NAECON). Dayton (OH); 1988.
[6] Waldock M, Roberts G, Sutton R. Terrain following control of an unmanned underwater vehicle using artificial neural networks. In: Proceedings of the IEE colloquium on control and guidance of remotely operated vehicles. London; 1995.
[7] Twigg S, Calise A, Johnson, E. On-line trajectory optimization for autonomous air vehicles. In: Proceedings of the AIAA guidance navigation and control conference. Austin, Texas; 2003
[8] Pettit R, Homer M. An autonomous threat evasion response algorithm for unmanned air vehicles during low altitude flight. In: Proceedings of the AIAA 1st intelligent systems technical conference. Chicago; 2004.
[9] Rehman A, Shahzad A, Kamal W, Samar R. Techniques for terrain following of autonomous vehicle. In: Proceedings of the 2nd European conference for aerospace sciences (EUCASS). Brussels; 2007.
[10] Hoffman J. Numerical methods for engineers and scientists. 2nd ed. New York: Marcel Dekker, Inc.; 2001.
[11] Betts J. Practical methods for optimal control using nonlinear programming. Society for Industrial and Applied Mathematics (SIAM); 2001.
[12] Fletcher R. Practical methods of optimization. John Wiley and Sons; 1987.
[13] Kuchar J. Markov model of terrain for evaluation of ground proximity warning system thresholds. J Guid Control Dyn 2001;24(3):428–35.
[14] Mathews J. Numerical methods for mathematics, science, and engineering. 2nd ed. Englewood Cliffs (NJ, USA): Prentice-Hall; 1992.
[15] McFarlane D, Glover K. A loop shaping design procedure using H_∞ synthesis. IEEE Trans Autom Control 1992;37(6):759–69.
[16] Tsai M, Geddes E, Postlethwaite I. Pole-zero cancellations and closed-loop properties of an H_∞ mixed sensitivity design problem. Automatica 1992;28(3):519–30.
[17] Samar R, Murad G, Postlethwaite I, Gu D-W. A discrete time H_∞ observer-based controller and its application to a glass tube production process. Eur J Control 1996;2:112–25.
[18] Smerlas A, Walker D, Postlethwaite I, Strange M, Howitt J, Gubbells A. Evaluating H_∞ controllers on the NRC Bell 205 fly-by-wire helicopter. Control Eng Pract 2001;9(1):1–10.
[19] Postlethwaite I, Prempain E, Turkoglu E, Turner M, Ellis K, Gubbells A. Design and flight testing of various H_∞ controllers for the Bell 205 helicopter. Control Eng Pract 2005;13:383–98.
[20] Skogestad S, Postlethwaite I. Multivariable feedback control analysis and design. 2nd ed. West Sussex, England: John Wiley & Sons; 2005.
[21] Samar R. Digital filters for gain stabilization of flexible vehicle dynamics. In: Proceedings of the 17th international federation of automatic control world congress. Seoul, South Korea; 2008.
[22] Glover K. All optimal Hankel-norm approximations of linear multivariable systems and their L^∞-error bounds. Int J Control 1984;39(6):1115–93.