Envelope Protection in Autonomous Unmanned Aerial Vehicles

Ilkay Yavrucuk, Suraj Unnikrishnan, J.V.R. Prasad
School of Aerospace Engineering, Georgia Institute of Technology
Atlanta,GA 30332

This paper details the application of an adaptive neural network based limit detection and avoidance algorithm for envelope protection on the autonomous Yamaha R-Max unmanned helicopter test bed. Software-in-the-loop and flight test results are presented. The envelope protection system is implemented as a mid-level controller component into the unmanned helicopter software infrastructure, called the Open Control Platform (OCP). The method utilizes an observer type adaptive neural network loop for the estimation of limit parameter dynamics. The constructed model is then used to predict dynamic trim response and corresponding command margins. Standard sensor measurements are used in the adaptation process and no off-line training of the networks is necessary. The command margin information is used to avoid prescribed limits.

Introduction

An aircraft's maneuvering capability is restricted by a set of envelope limits. Advanced flight control systems for autonomous Unmanned Aerial Vehicles (UAV) enable autonomous maneuvering that can challenge a UAV's flight envelope.[1,2] As a result, the development of automatic flight envelope protection systems emerge as an important element in the development of autonomous UAV's.[3,4] While envelope protection systems will improve the overall confidence and safety of UAV's, it will become essential when aggressive maneuvering close to envelope limits is desired. Therefore the capability of being aware and reacting automatically to approaching limits will be an important feature of future autonomous UAV's.

Recent studies, both for piloted and unmanned flight, have proposed various limit avoidance systems for the safe operation of a vehicle close to the edges of its flight envelope.[3–8] Some of these techniques rely on model based predictions and off-line training of neural nets with dynamic trim data.[3,5] In dynamic trim, the fast states, such as angular rates, are assumed to change fast and reach their steady state values quickly, while the slow states, such as forward speed, etc., are assumed to change slowly with time. The condition when fast states have reached a steady state corresponds to dynamic trim. A map representing the dynamic trim state is stored using an off-line trained neural network. Control margins are then calculated by perturbing the neural network. This approach gives the advantage of directly estimating the maximum response of limit parameters to current control inputs. The application of this technique to unmanned systems is presented in Ref. 3. However, the methods proposed in Refs. 3, 5 suffer from system uncertainties and the need for the generation of large amounts of dynamic trim data for off-line training of neural nets. Though some utilize adaptation,[6] their capabilities are limited, since a-priori knowledge of some system dynamics are still assumed.

Another family of methods calculate fixed-horizon predictions, where the usefulness is limited by the prediction time-step, which is typically a fraction of a second.[7,8]

In Refs. 9–12, an adaptive limit detection method for on-line estimation of dynamic trim values of a pre-selected set of limit parameters and corresponding control margins is developed. Neural networks are used for on-line adaptation. As off-line training of network weights is not used, the method has the advantage of adapting to varying flight conditions and different vehicle configurations. The method predicts dynamic trim parameters on-line and calculates the corresponding command or control margins for limit detection and avoidance. This technique is evaluated

Presented at the American Helicopter Society 59th Annual Forum, Phoenix, Arizona, May 6-8, 2003. Copyright ©2003 by the American Helicopter Society International, Inc. All rights reserved.

in simulation by combining an artificial pilot model and an active-force-feel model in the Generalized Tilt Rotor Simulation (GTRSIM) program.[12] In Ref. 4 the method is extended to automatic limit detection and avoidance applications. Software-in-the-loop simulations results using limit avoidance via control limiting and limit avoidance via command limiting for unmanned aerial vehicles were presented.

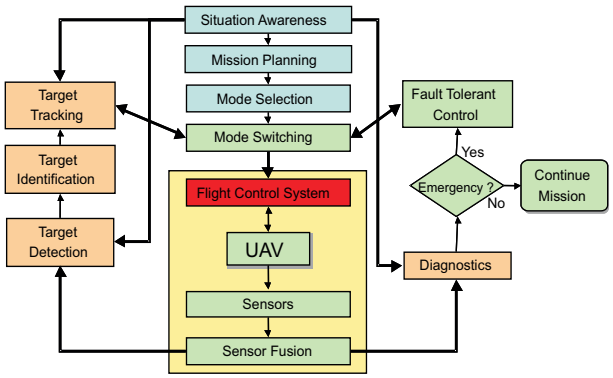

Fig. 1 Mission Intelligence Flow

An ongoing study under the DARPA sponsored Software Enabled Control (SEC) program at the Georgia Institute of Technology is considering applications of advanced control technologies to complex dynamic systems, in particular, to the control of intelligent unmanned aerial vehicles.[13] An objective of this project is to develop mid-level controllers and combine them with flight controllers in the lower level (Fig.1). In this context, flight modes are selected in the high and mid-level blocks of the overall control system and passed to the low level controller.[14] The low level controller is used to carry out those flight modes in a smooth and safe way. The low level controller needs a continuous knowledge of 'how far' the vehicle is from its structural and performance limit boundaries for safe maneuvering of the vehicle. If a 'fly safe' region violation in the flight envelope is foreseen, controller commands need to be modified automatically. This will guarantee a safe flight, regardless of the commands from the high- and mid-level controller blocks in the chain.

The applications conducted on the unmanned helicopter test-bed at Georgia Tech incorporate a new software infrastructure, called the Open Control Platform (OCP), also developed under the SEC program. The OCP aims to provide a software environment for complex systems that coordinate and support distributed interaction among diverse components and support dynamic reconfiguration and customization of the components in real time.[15] The envelope protection system is integrated through the OCP as a mid-level controller component.

This paper presents the implementation of the adaptive limit detection and avoidance algorithms as an envelope protection system to the Yamaha R-Max Unmanned helicopter (GTMAX) test bed at Georgia Tech. The algorithms were tested using software-in-the-loop and hardware-in-the-loop simulations. The implementation involved the integration of the envelope protection algorithms with the existing algorithms through the OCP. Flight tests were conducted to avoid rotor stall limits during fast accelerations and decelerations in forward flight. The Expected Retreating Indicated Tip Speed (ERITS) factor was used as a measure for rotor stall.

Methodology

The adaptive limit detection and avoidance methods of Refs. 4,9–12 is based on establishing a functional relationship between the limit parameter response and a set of measurable states and control variables. The idea is to utilize these relationships in predicting dynamic trim response of the limit parameter. Furthermore, the model developed for the limit parameter dynamics is used in computing the control or command limit values, that would cause an envelope violation. The method is applied both for manned and unmanned aircraft.

In an unmanned vehicle application, where envelope protection is to be automated, the system can be set-up in two different ways. One way to achieve automatic limit avoidance is to calculate the allowable control travel of each control channel and limit the controller outputs (u) artificially. Here, the modeling is done between the controller output and the limit parameter (y_p) (Fig. 2).

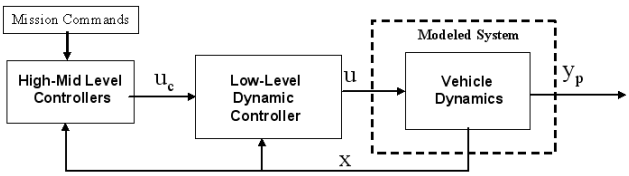

Fig. 2 System modeling for control limiting.

Problems associated with the use of control limiting are similar to those of actuator saturation. For instance, issues related to closed loop stability need to be re-assessed.

An alternative to control limiting is the use of command limiting. In developing a relationship between the limit parameter or the fast states and an input, it is possible to view the system to be modeled as the combination of the low level controller and the vehicle, instead of the vehicle only. A block diagram representation of this approach is shown in Fig. 3. Then the controller inputs (u_c) can be limited and closed loop stability can still be assured.

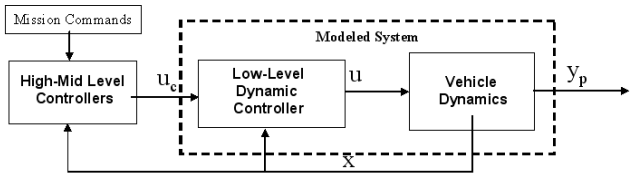

Fig. 3 System modeling for command limiting.

Following is a detailed development of the method for command limiting for automatic limit detection and avoidance. In this paper, the method is applied to limit parameters, whose sensor measurements are available and reach their maximum values in steady state. In cases, where direct sensor measurements of limit parameters is not available the method can still be applied by constructing approximate adaptive models for the fast states and estimating their dynamic trim values.[11] The fast states required for the calculation of the dynamic trim can be measured through standard sensors.

Automatic Limit Detection and Avoidance via Command Limiting

Let us represent the closed loop dynamics of an aircraft with the following nonlinear state equation:

$$\dot{x} = f(x, u_c); \qquad x(t_0) = x_0, \qquad x \in \Re^n, u_c \in \Re^i \tag{1}$$

where x is the state vector with unknown initial condition x_0, and $u_c \in U$ is the controller command vector. The function f is assumed to be continuous and satisfying the global Lipschitz condition. Moreover, for any finite initial condition the solution of eqn. 1 does not escape in finite time.

The state x can be divided into fast and slow variables with their representative dynamic equations,

$$\dot{x}_f = f_1(x_f, x_s, u_c) \tag{2}$$

$$\dot{x}_s = f_2(x_f, x_s, u_c) \tag{3}$$

Here x_f denotes the fast states and x_s the slow states of the aircraft.

Let a dynamic controller be represented in the following form:

$$\dot{x}_c = f_3(x_c, x_f, x_s, u_c) \tag{4}$$

$$u = g_1(x_c, x_f, x_s, u_c) \tag{5}$$

where x_c is the controller state vector and u the controller output (Fig.4). f_1, f_2, f_3 satisfy the same assumptions as f.

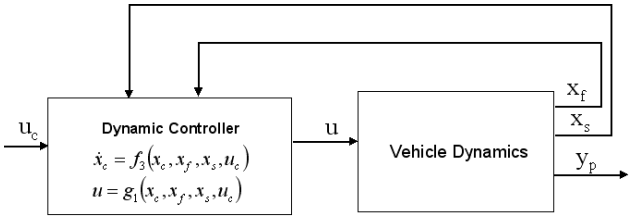

Fig. 4 Block diagram representation of controller and vehicle dynamics.

Flight parameters such as forward speed and Euler angles can be considered as slow states, whereas the angular rates and angle of attack can be taken to be fast states. If a dynamic controller is used, the controller states are also assumed to be fast and therefore can be included in the fast state vector x_f. In general, the dynamic trim condition then corresponds to

$$\dot{x}_f = 0 \tag{6}$$

Let us also assume a vector y_p consisting of limit parameters as

$$y_p = g(x_f, x_s, u_c) \quad , y_p \in \Re^m \tag{7}$$

When the limit parameters in y_p can be mapped to fast states as in eqn. 7, in dynamic trim the limit parameters y_p will also reach a quasi-steady condition. Therefore, the dynamic trim values of the limit parameters may be obtained using

$$y_{p_{DT}} = g(x_{f_{DT}}, x_s, u) \tag{8}$$

The subscript 'DT' denotes dynamic trim. Thus the dynamic trim response of the limit parameter ($y_{p_{DT}}$), can be calculated using the dynamic trim estimates of the fast states.

First-order Approximation for the Limit Parameter

A large class of limits reach their maximum value in steady state. Their dynamic behavior can be approx-

imated using first-order differential equations. Consider the following form for the dynamics of y_p :

$$\dot{y}_p = \tilde{h}(y_p, u_c) \tag{9}$$

In eqn. 9 the function $\tilde{h}$ represents the local dynamic behavior of the limit parameter y_p to the command input u_c. In an effort to construct a global model, slow states might be added as inputs. Slow states could also include changes in parameters like weight, altitude, C.G. location. Indeed, slow states will appear as an input to the system. Finally, a first order differential equation of the following form is considered:

$$\dot{y}_p = h(y_p, x_s, u_c) \tag{10}$$

The function h satisfies the same conditions as function f. The fast states do not appear in the above equations as their contribution is inherent in the first-order behavior of y_p. Here, only limits that reach their maximum values in steady state and show no coupled dynamics with the fast states are considered. Implicitly, a *unique mapping* between the fast states and the limit parameter is assumed. Moreover, a steady state condition of the limit parameter also corresponds to a steady state of the fast variables, and therefore to the dynamic trim condition. Now, a linear approximation for the function h of eqn. 10 can be written as

$$\dot{y}_p = Ay_p + Bu_c + \xi(y_p, x_s, u_c) \tag{11}$$

where ξ is the modeling error and the remaining terms represent a linear approximation for the function h. An adaptive neural network with appropriate weight update laws can be developed to approximate this modeling error. If ν_{ad} represents the output from such a network, then eqn. 11 can be approximated as:

$$\dot{\hat{y}}_p = A\hat{y}_p + Bu_c + \nu_{ad}(y_p, x_s, u_c) \tag{12}$$

where $\hat{y}_p$ is the approximation of y_p.

The dynamic trim response of the limit parameter ($\hat{y}_{p_{DT}}$) could then be computed by setting $\dot{\hat{y}}_p = 0$ and solving the nonlinear equation:

$$A\hat{y}_{p_{DT}} + Bu_c + \nu_{ad}(\hat{y}_{p_{DT}}, x_s, u_c) = 0 \tag{13}$$

Also using eqn. 12 the command limit $\hat{u}_{c_{lim}}$ could be obtained by solving the following nonlinear equation:

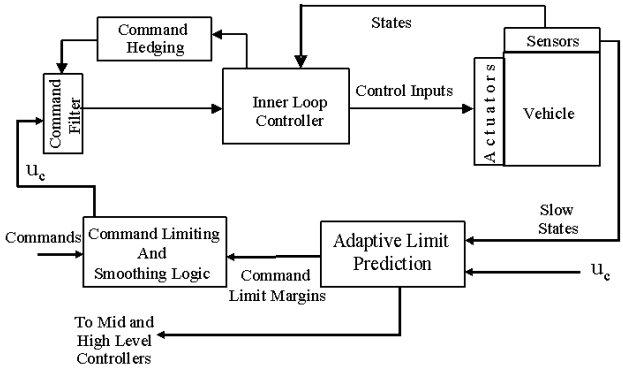

Fig. 5 Block diagram of limit avoidance via command limiting.

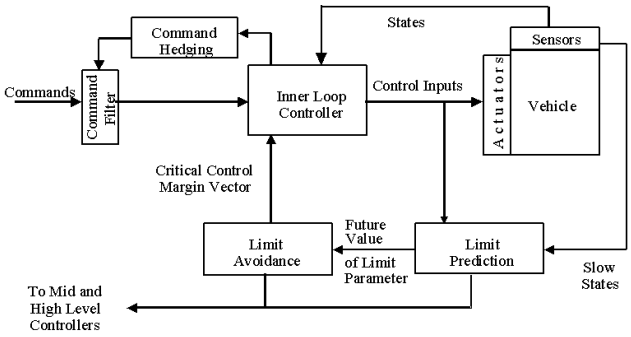

Fig. 6 Block diagram of limit avoidance via control limiting architecture.

$$Ay_{p_{lim}} + B\hat{u}_{c_{lim}} + \nu_{ad}(y_{p_{lim}}, x_s, \hat{u}_{c_{lim}}) = 0 \tag{14}$$

The command margin than can be defined as:

$$\hat{u}_{c_{marg.}} = \hat{u}_{c_{lim}} - u_c \tag{15}$$

If an appropriate weight update law can be developed for an adaptive neural network so as to obtain a good functional approximation of the limit parameter dynamics, than one could estimate and predict the dynamic trim response behavior and the command margins on-line.

Figure 5 is a block diagram representation of the limit avoidance via command limiting architecture. The limit detection block receives the current values of the slow states and the current command inputs. The dynamic trim predictions leading the actual response, are used to calculate the command margin vector and then limit the command inputs to the controller accordingly.

As an alternative, Fig. 6 is a block diagram representation of the limit avoidance via control limiting architecture. The limit detection block receives the

current values of the slow states and the current control inputs. The dynamic trim predictions are used to calculate the control margin vector and they are passed to the low level controller as artificial control saturation limits.[4]

Adaptive Architecture

The proposed architecture consists of one adaptive loop to construct approximate limit parameter dynamics (see Fig. 7) and two algebraic loops (see eqns. 17 and 18) to solve for dynamic trim and command limit values. Figure 7 is a block diagram representation of eqn. 12 with an additional error feedback term. As will be seen in the Lyapunov analysis, the proportional feedback of error is for the faster convergence of the error dynamics. Note that, the approximate model dynamics for $\hat{y}_p$ now becomes:

$$\dot{\hat{y}}_p = A\hat{y}_p + Bu_c + \nu_{ad}(y_p, x_s, u_c) + Ke \tag{16}$$

The adaptive neural network utilized in this observer loop is a nonlinearly parameterized single hidden layer network with sigmoid basis functions. The input to the network are the slow states, controller command and the limit parameter measurement. The network weights adapt based on the model error. The neural network weight adaptation laws are derived in the next section.

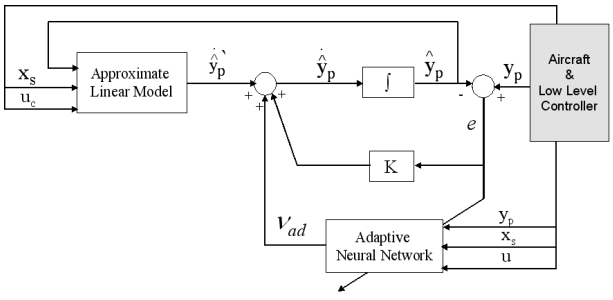

Fig. 7 Observer loop with adaptive neural network for model of limit parameters.

In eqns. 13 and 14 the variables to be solved, $\hat{y}_{p_{DT}}$ and $\hat{u}_{c_{lim}}$ appear inside the nonlinear function ν_{ad} which complicates an analytic solution. However, fixed-point solutions can be guaranteed to exist to these equations, when the neural network is guaranteed to have bounded outputs. A fixed-point solution for these can be obtained by using the following equations:

$$\begin{aligned}\hat{u}_{c_{lim_{i+1}}} =& -B^{-1}[Ay_{p_{lim}} + \nu_{ad}(y_{p_{lim}}, x_s, \hat{u}_{c_{lim_i}}) \\ &+ Ke] \end{aligned} \tag{17}$$

$$\hat{y}_{p_{DT_{i+1}}} = -A^{-1}[Bu_c + \nu_{ad}(\hat{y}_{p_{DT_i}}, x_s, u_c) + Ke] \tag{18}$$

In practice, however, separate iterative loops for eqns. 17 and 18 are not needed if used along with the neural network weight update.[10] Since the number of equations for $\hat{y}_p$ will be the same as the number of elements in $\hat{y}_p$, A is a square matrix. Moreover, for a unique solution to exist, it has to be invertible. Depending on the number of input parameters u_c and the number of limit parameters, the inversion of B might not be always possible. The above is possible, if the number of limit parameters are associated with the same number of controls and B is invertible. A least squares approach could be used for cases where the number of control inputs is larger than the number of limit parameters.[11]

Single Hidden Layer Artificial Neural Network Augmentation

A neural network is a parallel structure of interconnected neurons or neural processors. Given $x \in \Re^N$, a three layer NN has an output given by:

$$y_i = \sum_{j=1}^{N_2}[m_{ij}\sigma(\sum_{k=1}^{N_1} n_{jk}x_k\theta_{nj}) + \theta_{mi}], i = 1, \ldots, N_3 \tag{19}$$

where σ is the activation function, n_{jk} are the first to second layer interconnection weights, m_{ij} are the second to third layer interconnection weights, N_2 is associated with the number of neurons in the hidden layer, θ_{nj} and θ_{mi} are bias terms. Such an architecture can approximate any continuous nonlinearities with "squashing" activation functions.[16,17] Hence any general function $f(x) \in C, x \in D \subset \Re^n$ can be written as:

$$f(x) = M^T\sigma(N^Tx) + \epsilon(x) \tag{20}$$

where $\epsilon(x)$ is known as the function reconstruction error.

The nonlinear approximation ν_{ad} in eqn. 16 can be replaced by using a so-called single hidden layer neural

network of the form of eqn. 19. In matrix form, the approximation can be written as,

$$\nu_{ad} = \hat{M}^T \sigma(\hat{N}^T \bar{\mu}) \tag{21}$$

where $\hat{M}$ and $\hat{N}$ are the adaptive NN weights and $\bar{\mu}$ is the normalized network input vector. Following is the development of the tuning law of the weights $\hat{M}$ and $\hat{N}$ of eqn. 21.

Weight update law

Lyapunov analysis technique is used to derive the update laws for the neural network weights.

The error equation, obtained using eqns. 11 and 16, can be written as:

$$\dot{e} = Ae - [M^T \sigma(N^T \bar{\mu}) - {M_o}^T \sigma({N_o}^T \bar{\mu}) + \epsilon] - Ke \tag{22}$$

where $e = y_p - \hat{y}_p$ and M_o, N_o are ideal network weights which best approximates the modeling error ξ.

Using the Taylor series expansion for $M^T \sigma(N^T \bar{\mu})$ eqn. 22 can be expanded as:

$$\dot{e} = Ae - [\tilde{M}^T(\sigma - \sigma' N^T \bar{\mu}) + M^T \sigma' \tilde{N}^T \bar{\mu} + w] - Ke$$

where $\tilde{M} = M - M_o$ and $\tilde{N} = N - N_o$. Also,

$$w = \epsilon + {M_o}^T[\sigma_o - \sigma + \sigma' \tilde{N}^T \bar{\mu}] + \tilde{M}^T \sigma' {N_o}^T \bar{\mu}$$

Consider the following Lyapunov candidate function,

$$L = \frac{1}{2}[e^T P e + tr(\tilde{M}^T F^{-1} \tilde{M}) + tr(\tilde{N}^T G^{-1} \tilde{N})] \tag{23}$$

and the weight update laws:

$$\dot{M} = -F[(\sigma - \sigma' N^T \bar{\mu})(e^T P) + \kappa ||e|| M] \tag{24}$$

$$\dot{N} = -G[\bar{\mu}((e^T P) M^T \sigma') + \kappa ||e|| N] \tag{25}$$

In eqn. 23 the matrix P is the solution of the Lyapunov equation,

$$A^T P + PA = -Q \tag{26}$$

and is known to be unique for any given positive definite matrix Q when all eigenvalues of A have negative real part. Taking the derivative of Lyapunov candidate function with respect to time and substituting the update laws in eqns. 24 and 25 the following is obtained:

$$\dot{L} = -\frac{e^T Q e}{2} - e^T P w - e^T K e - \kappa ||e|| tr(\tilde{Z}^T Z) \tag{27}$$

where $Z = \begin{bmatrix} M & 0 \\ 0 & N \end{bmatrix}$ and $\tilde{Z} = \begin{bmatrix} \tilde{M} & 0 \\ 0 & \tilde{N} \end{bmatrix}$. Using the relation $-tr(\tilde{Z}^T Z) \leq ||\tilde{Z}||||Z_o|| - ||\tilde{Z}||^2$ and the fact that norm of w is upper bounded as:

$$||w|| \leq c_o + c_1 ||\tilde{Z}|| + c_2 ||e|| ||\tilde{Z}|| + c_3 ||\tilde{Z}||^2 \tag{28}$$

where c_o, c_1, c_2, c_3 are constants. The derivative of Lyapunov function can be written as:

$$\dot{L} \leq -\frac{\lambda_{min}(Q)}{2} ||e||^2 + ||e|| ||P|| ||w|| - K||e||^2 - \kappa ||e|| ||\tilde{Z}||^2 + \kappa ||e|| ||\tilde{Z}|| \bar{Z} \tag{29}$$

$$\implies \dot{L} \leq -\frac{\lambda_{min}(Q)}{2} ||e||^2 - (\kappa - ||P|| c_3) ||e|| ||\tilde{Z}||^2 + a_o ||e|| + a_1 ||e|| ||\tilde{Z}|| - (K - c_2 ||P|| ||\tilde{Z}||) ||e||^2 \tag{30}$$

where $a_o = c_o ||P||$ and $a_1 = c_1 ||P|| + \kappa \bar{Z}$. $\bar{Z}$ is the upper bound on the norm of the ideal weight matrix Z_o, i.e $||Z_o|| \leq \bar{Z}$

If $K \geq c_2 ||P|| ||\tilde{Z}||$ and $\kappa \geq ||P|| c_3$ then $\dot{L} \leq 0$ when,

$$||\tilde{Z}|| \geq Z_m = \frac{a_1 + \sqrt{a_1^2 + 4 a_o (\kappa - ||P|| c_3)}}{(\kappa - ||P|| c_3)} \tag{31}$$

or, when

$$||e|| \geq \frac{a_o + a_1 Z_m}{\frac{1}{2} \lambda_{min}(Q)} \tag{32}$$

Hence by appropriate choice of $\lambda_{min}(Q)$ and K along with network parameters F, G, κ the derivative of Lyapunov function L can be made negative everywhere outside a compact set $\bar{D}$. Thus for initial conditions within this compact set, the model error e and weights $\tilde{Z}$ are uniformly ultimately bounded, when the weight update laws of eqns. 24 and 25 are used.

Application to the Yamaha R-Max Unmanned Helicopter Test-bed

As part of the ongoing DARPA sponsored Software Enabled Control (SEC) program a Yamaha R-MAX helicopter is modified to serve as a test-bed (GTMAX) for enabling technologies for unmanned autonomous helicopters (Fig. 8). Recent testing included mid-level controller algorithms. The design and testing of the automatic envelope protection system was one among them.

The integration of the envelope protection system to the GTMAX test bed involved its testing and verification with the software-in-the-loop simulation and

Fig. 8 GTMAX, autonomous helicopter test bed

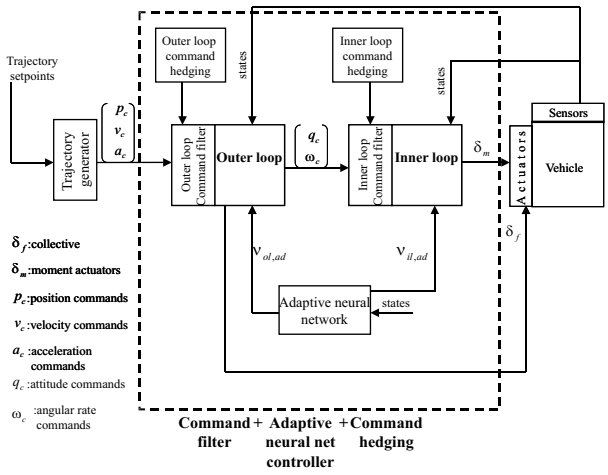

Fig. 9 GTMAX adaptive neural net based controller architecture.

the onboard computer software and hardware. As the envelope protection system would modify trajectory commands that enter the low-level controller the integration of the system with the existing low level controller was essential.

In Ref. 1 the baseline flight controller for the GTMAX was presented. A simplified block diagram of this low level controller architecture along with a mid-level trajectory generator is shown in Fig. 9. The input to the trajectory generator are trajectory set points, which is a set of position and heading commands. Then continuous acceleration commands (a_c) are generated and integrated to obtain velocity (v_c) and position commands (p_c) to be used in the outerloop. Hence, in the present architecture, acceleration commands from the trajectory generator can be viewed as the independent inputs. Based on the method explained in previous sections for adaptive limit detection and avoidance, a functional representation between the limit parameter and the trajectory acceleration commands is assumed. Then the acceleration command limits (a_{lim}) can be computed. The acceleration commands are then modified to avoid limit violations. The acceleration command ultimately input to the outer loop (a_{inp}) is calculated as follows:

$$a_{inp} = a_c + K_A(a_{lim} - a_c) \tag{33}$$

where a_c is the command that would have gone through without limit avoidance, a_{lim} is the predicted acceleration command limit and a_{inp} is the modified acceleration command input. K_A is a function of $a_{lim} - a_c$. (Ref.4)

A block diagram representation of this limit avoidance architecture is shown in Fig. 10.

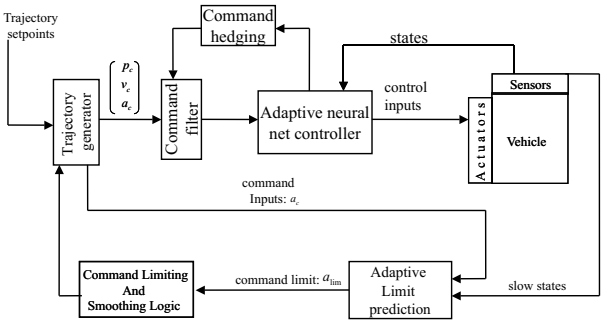

Fig. 10 Proposed limit detection and avoidance architecture for the GTMAX unmanned helicopter.

For the testing of the system, the Expected Retreating Indicated Tip Speed (ERITS) factor is considered as a measure of rotor stall limit and used as the limit parameter. The ERITS factor is estimated using the following equation:[7]

$$ERITS = \frac{V_{tip}\frac{\rho}{\rho_s} - v_{cas}}{N_z} \tag{34}$$

where V_{tip} is the rotor tip speed, v_{cas} is the calibrated airspeed, N_z is the load factor and $\frac{\rho}{\rho_s}$ is the relative air density. All the data required to calculate the ERITS number was available from sensor measurements onboard the unmanned helicopter.

In the software-in-the-loop simulations and flight tests presented in this paper, the limit parameter (ERITS) is limited using one command input, namely a_{c_t} the acceleration command component laying in the x-y plane of the vehicle carried frame. Once the command limit for a_{c_t} is calculated, it is resolved into its appropriate components and used as command limits in the trajectory generator.

System Integration Using the Open Control Platform

One of the objectives of the DARPA sponsored SEC program is the design and development of an

Open Control Platform (OCP) and to demonstrate its benefits.[15] The OCP would provide a base for control systems with features such as adaptability whereby, for instance, individual components can dynamically reconfigure themselves to provide optimal system performance during extreme maneuvers. The OCP based control systems would also allow interoperability, which means control algorithms running on different software hardware platforms, different processors etc. would still be able to interact and communicate with each other. Other main features provided by the OCP are Plug-and-Play extensibility and openness.[15, 18]

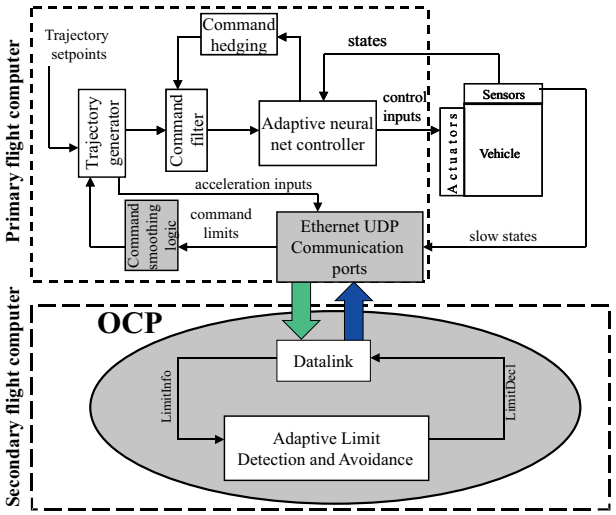

Fig. 11 Block diagram representation of the setup used for flight evaluation of the limit detection and avoidance algorithms on the OCP.

The SEC program aims to develop and integrate mid-level and high-level control algorithms and demonstrate benefits when used through the OCP. As part of achieving this goal the envelope protection component is integrated through the OCP as a mid-level component into the GTMAX software design. The GTMAX avionics design incorporates two onboard computers. The primary computer runs the low level flight controller and the secondary computer runs other software components, such as the the mid-level controller components. In the implementation, the envelope protection system is run on the secondary computer. The OCP-datalink component was used to exchange data between the ethernet UDP ports of the primary and secondary flight computers. The primary flight computer supports the architecture shown in Fig. 9 with the low-level controller and trajectory generator. First, the OCP was compiled on a computer running the QNX operating system, with a limit avoidance module and a datalink component. The executable code was then made to run on the secondary flight computer of the GTMAX also running the QNX operating system. The final setup with the primary and secondary flight computers is shown in Fig. 11. In Fig. 11 the datalink component of the OCP receives an encoded signal from the ethernet UDP port of the primary flight computer. This signal is decoded within the datalink and contains the necessary information required by the components within the OCP, in this case the limit avoidance module.

The envelope protection module is set up to prevent limit violations of the ERITS factor. Based on sensor information provided through the OCP the ERITS factor is computed and an adaptive nonlinear model is developed. The acceleration command limits are computed from the model and then sent through the datalink. The datalink encodes this signal and sends it to the primary flight computer as shown in Fig. 11.

Software-in-the-loop Simulation and Flight Test Results

In moving towards a successful flight test all software is first tested in a software-in-the-loop (SITL) simulation.[19] In this section, the results of the SITL evaluations of the adaptive limit detection and avoidance algorithm on the GTMAX unmanned helicopter test bed is presented. Rotor stall limits are considered as the limit parameter.

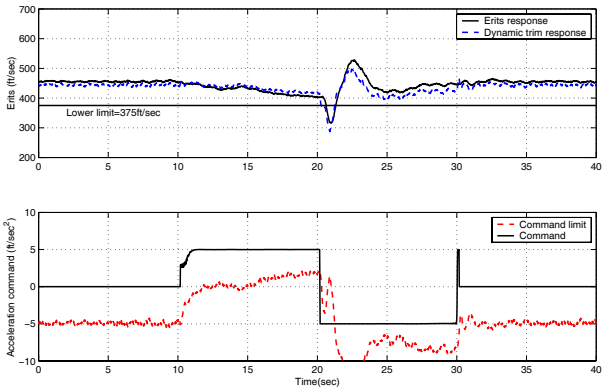

Fig. 12 SITL simulation results. Limit avoidance OFF.

In this simulation, a lower limit of 375 ft/sec is assumed for the ERITS factor. The load factor is estimated using the filtered sensor measurements of

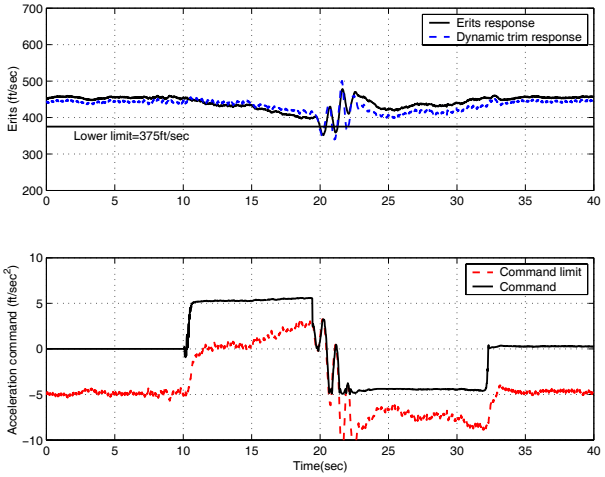

Fig. 13 SITL simulation results. Limit avoidance ON.

the body accelerations. The filter used is a first order low pass filter with a cut off frequency of 2Hz. The calibrated airspeed is computed using the measured velocity components along the three body axes.

The simulation starts from an initial hovering condition. After around 10 seconds into the simulation the trajectory module generates the position, velocity and acceleration commands to fly the vehicle to a point 500ft north of the current position and hover. A functional representation of the limit parameter dynamics with respect to the acceleration command (a_{c_t}) is generated. The maneuver is first run with limit avoidance OFF, but limit detection ON. The results for this case are presented in Fig. 12. The upper figure of Fig. 12 shows the actual ERITS factor response along with the dynamic trim response violating the assumed lower limit of 375 ft/sec. The lower figure of Fig. 12 shows the command limit as estimated by the limit detection algorithm along with the acceleration command. Note that, the command limit violation precedes the limit parameter violations. Also, the dynamic trim estimation shown in Fig. 12 leads the actual response.

Next, the same maneuver is run with both the limit detection and limit avoidance ON. Plots in Fig. 13 present those results. When the limit avoidance routines are turned ON, the ERITS factor response is forced to stay above the lower boundary of 375 ft/sec. Figure 13 also shows the modified acceleration command staying within the predicted command margin. Slight system induced oscillations could be avoided through further adjustment of the limit avoidance routines.

The software-in-the-loop simulation results presented above illustrated the effectiveness of the proposed adaptive limit detection and avoidance scheme and brought confidence to flight tests.

Flight Tests

Figure 14 is a concise representation of the setup used for flight test evaluations. The rate of communication between the primary flight computer and the secondary flight computer through the OCP datalink component is 50Hz. Also, the datalink and limit avoidance components within the OCP exchange information at the same rate. The signal LimitInfo from the datalink to the limit avoidance module contains the following sensor measurements of the vehicle: The main rotor RPM, position, velocity, acceleration and time from the onboard primary flight computer. The limit avoidance component sends back the signal LimitDecl which contains the computed acceleration command limit values.

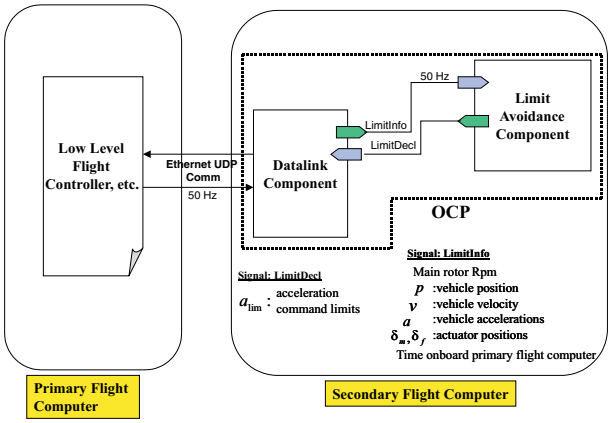

Fig. 14 Reference architecture for flight testing of the limit detection and avoidance algorithms on the GTMAX.

The maneuver considered was a forward dash to a point 600ft from the initial hover point and a backward flight to the starting point. The maneuver ended in hover. This maneuver required accelerations and decelerations, that resulted in exceedance of the prescribed rotor stall limits.

Two flight tests were conducted. First the limit avoidance routine was OFF and the limit detection was ON. During this maneuver the vehicle violated its limit boundary. Fig. 15 shows results of the dynamic trim estimation and the command limit prediction. The limit detection algorithm did predict a command limit violation, when the dynamic trim response exceeded

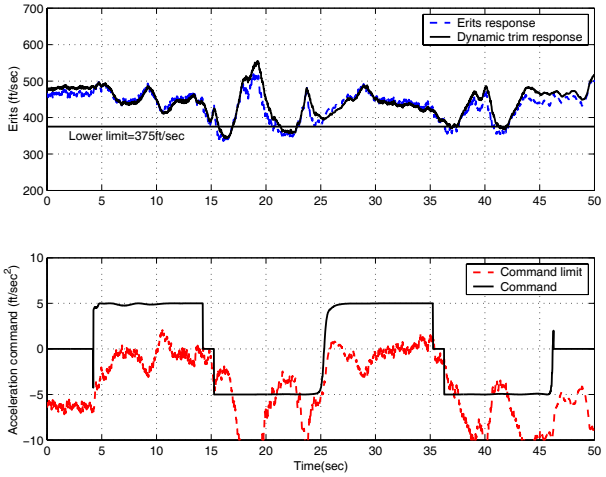

Fig. 15 Flight test results. Limit Avoidance OFF.

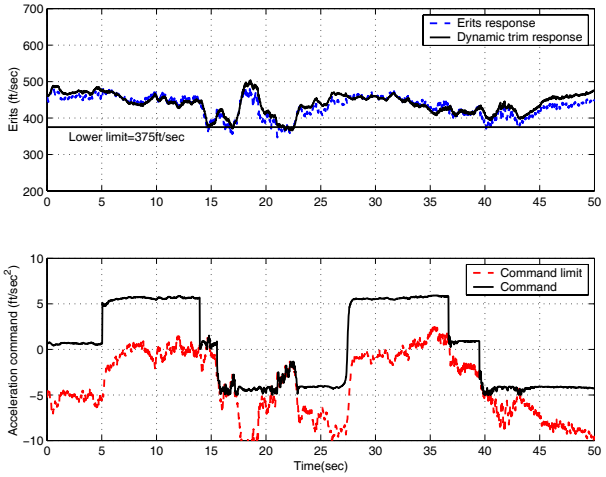

Fig. 16 Flight test results. Limit avoidance ON.

the limit.

Next, both routines were switched ON and the same maneuver was executed (see Fig. 16). The vehicle was successful in staying within the prescribed limits. Comparing Figs. 15 and 16 it can be observed that, when the limit avoidance routine is switched ON the acceleration command follows the command limit when a limit violation is foreseen. By modifying the acceleration command the limit parameter corresponding to rotor blade stall (ERITS) is kept within the limit boundary.

Conclusions

A previously developed neural network based adaptive limit detection and avoidance algorithm is integrated into the unmanned helicopter test bed at Georgia Tech. Successful flight test results demonstrated the effectiveness of the system when used as an envelope protection system on a UAV as a mid-level controller component. The limit detection algorithms have successfully predicted dynamic trim conditions, limit and command margins. The envelope protection system interacted with low level controller commands and adjusted them such that the aircraft stays in its prescribed limit boundaries. Procedures followed prior to flight tests increased system confidence and cut the development time. The capabilities of the open control platform were effective in the ease of integration and the flexibility in changing, developing and interconnecting software components.

The ERITS number was used as a measure for rotor stall. Therefore one limit parameter was controlled using one effective input channel. Currently, the system is being extended to be able to limit multiple limit parameters using multiple command control channels.

Acknowledgments

This study is carried out under the NRTC Center of Excellence in Rotorcraft Technology (CERT) program and the DARPA sponsored Software Enabling Control program at Georgia Tech.

References

[1]Johnson, E. and Kannan, S., "Adaptive Flight Control for an Autonomous Unmanned Helicopter," Aug. 2002, In Proceedings of the AIAA Guidance, Control and Navigation Conference, Monterey,CA.

[2]Gravilets, V., Martinos, I., Mettler, B., and E.Feron, "Control logic for Automated Aerobatic Flight of a Miniature Helicopter," Aug. 2002, Proceedings of the AIAA Guidance, Navigation and Control Conference,Monterey, CA.

[3]Yavrucuk, I. and Prasad, J. V. R., "Automatic Limit Detection and Avoidance for Unmanned Helicopters," May 2001, In Proceedings of the American Helicopter Society 57th Annual Forum, Washington D.C.

[4]Prasad, J., Yavrucuk, I., and Unnikrishnan, S., "Adaptive Limit Prediction and Avoidance for Rotorcraft," Sept. 2002, In Proceedings of the 28th European Rotorcraft Forum, Bristol, UK.

[5]Horn, J., Calise, A. J., and Prasad, J. V. R., "Flight Enveleope Cueing on a Tilt-Rotor Aircraft Using Neural Network Limit Prediction," *Journal of the American Helicopter Society*, Vol. 46, No. 1, Jan. 2001.

[6]Horn, J., Calise, A. J., and Prasad, J. V. R., "Flight Envelope Limit Detection and Avoidance for Rotorcraft," Sept. 1999, In Proceedings of the European Rotorcraft Forum, Rome, Italy.

[7]Menon, P. K., Iragavarapu, V. R., and Whalley, M. S., "Estimation of Rotorcraft Limit Envelopes Using Neural Networks," June 1996, In Proceedings of the American Helicopter Society 52th Annual Forum, Washington D.C.

[8]Bateman, A. J., Ward, D. G., Barron, R. L., and Whalley, M. S., "Piloted Simulation Evaluation of a Neural Network Limit Avoidance System for Rotorcraft," May 1998, AIAA.

[9]Yavrucuk, I., Prasad, J. V. R., and Calise, A. J., "Adaptive Limit Detection and Avoidance for Carefree Maneuvering," Aug. 2001, In Proceedings of the AIAA Atmospheric Flight Mechanics Conference, Montreal, Canada.

[10]Yavrucuk, I., Prasad, J. V. R., Calise, A. J., and Unnikrishnan, S., "Adaptive Limit Margin Prediction and Limit Avoidance," June 2002, In Proceedings of the American Helicopter Society 58th Annual Forum, Montreal, Canada.

[11]Yavrucuk, I., Prasad, J. V. R., and Calise, A. J., "Carefree Maneuvering Using Adaptive Neural Networks," Aug. 2002, In Proceedings of the AIAA Atmospheric Flight Mechanics Conference, Monterey,CA.

[12]Yavrucuk, I. and Prasad, J. V. R., "Adaptive Limit Margin Prediction and Control Cueing for Carefree Maneuvering of VTOL Aircraft," Oct. 2002, In Proceedings of the AHS Flight Control and Crew Design Technical Specialists' Meeting, Philadelphia, PA.

[13]Schrage, D. P. and Vachtsevanos, G., "Software Enabled Control for Intelligent UAV's," 1999, In Proceedings of the IEEE International Conference on Control Applications.

[14]Yavrucuk, I., Kannan, S., and Prasad, J. V. R., "Simulation Evaluation of a Reconfigurable Flight Controller of a Heli-UAV for Extreme Maneuvers," Aug. 2000, In Proceedings of the AIAA Atmospheric Flight Mechanics Conference, Denver,CO.

[15].Wills, L., Kannan, S., M.Guler, S. S., Heck, B., Prasad, J. V. R., and D. Schrage, G. V., "An Open Control Platform for Reconfigurable Control," June 2001, IEEE Control Systems Magazine.

[16]Funahashi, K., "On the Approximate Realization of Continous mAppings by Neural Networks," 1989, Neural Networks.

[17]Hornik, K., Stinchcombe, M., and White, H., "Multilayer Feedforward Networks are Universal Approximators," 1989, Neural Networks.

[18]Wills, L., Sander, L., Kannan, S., Kahn, A., Prasad, J. V. R., and Schrage, D., "An Open Control Platform for Reconfigurable Distributed Hierarchical Control Systems," Oct. 2000, In Proceedings of the Digital Avionics Systems Conference, Philadelphia, PA.

[19]Johnson, E. N. and Mishra, S., "Flight Simulation for the Development of an Experimental UAV," Aug. 2002, In Proceedings of the AIAA Modeling and Simulation Technology Conference, Monterey, CA.