2005 American Control Conference
June 8-10, 2005. Portland, OR, USA

Implementing and Testing a Nonlinear Model Predictive Tracking Controller for Aerial Pursuit/Evasion Games on a Fixed Wing Aircraft

J. Mikael Eklund, Jonathan Sprinkle and Shankar Sastry

Abstract— **Flight test and simulation results are presented for a Nonlinear Model Predictive Tracking Controller (NMPC) used in pursuit and evasion maneuvers in three dimensions on a fixed wing Unmanned Aerial Vehicle (UAV) for the purposes of pursuit/evasion games (PEGs) against a piloted F-15 aircraft. These controllers are shown to be effective for both asymmetric and symmetric PEGs. While the capability of UAVs to perform autonomously has not yet been demonstrated, this is an important step to enable at least limited autonomy in such aircraft to operate with temporary loss of remote control, or when confronted with an adversary or obstacles for which remote control is insufficient. Such capabilities have been under development in the Software Enabled Control (SEC) program and were recently tested in the Capstone Demonstration of that program.**

I. Introduction

Unmanned Aerial Vehicle (UAV) have recently been used with great success in gathering military intelligence [1] by providing a viable alternative to manned aircraft through their smaller size, reduced risk to life and limb, and reduced cost. These successes and challenges have stimulated research into UAV autonomy.

UAVs have, however, exhibited very little autonomy to-date, and in the face of adversaries this lack of autonomy is a liability. While some defense against ground based adversaries can be achieved through either low-level or very high-level flight, success against an intelligent (i.e. manned) airborne adversary must rely on one of four possible dimensions in which to obtain an advantage: speed, maneuverability, munitions, and intelligence of control. This experiment focuses on the last of those by improving the intelligence of the aircraft, which allows for current aircraft designs to be reused with software changes.

Nonlinear model predictive control (NMPC) is a control technique that explicitly addresses nonlinear systems with constraints on operation and performance. Previously the use of NMPC has been shown to be effective for rotary-wing UAVs [2]. Aerial vehicles, with their nonlinear dynamics and input/state constraints to guarantee adherence to safe flight, are a proving ground for this technology. Although, the use of these control methods that run in real-time on fixed-wing UAVs has been in development [3], this has not been previously demonstrated in flight test. This is also the first application of NMPC to a symmetric PEG in which the evader aircraft is able to switch roles and become the pursuer.

This research was supported under DARPA's IXO SEC program, under contract number DARPA SEC F33615-98-C-3614.

Department of Electrical Engineering and Computer Sciences, University of California, Berkeley. Berkeley, CA 94720, USA {eklund,sprinkle,sastry}@eecs.berkeley.edu

Fig. 1. The Global Hawk Medium Altitude Long Endurance UAV (photo courtesy of US Department of Defense).

In this paper, the results of the final integration and testing for the Capstone Demonstration of the Software Enabled Control (SEC) program are presented for the fixed wing pursuit/evasion games (PEGs). A numerically efficient nonlinear model predictive tracking control (NMPTC) algorithm is used to encode the PEG between two fixed-wing adversaries. This follows on earlier work [3] and the approach of [4]. The control problem is formulated as a cost minimization problem in the presence of input and state constraints. The minimization problem is solved with a gradient-descent method, which is computationally light and fast [4]. The NMPTC controller uses an interface to an existing autopilot in order to influence the system behavior. By formulating the cost function to include the state information of the other aircraft, input saturation, state constraints and flight test boundary constraints, we show the performance of the NMPTC as a one-step solution for trajectory planning and control of UAVs competing in a PEG.

This paper describes the experimental results of flight tests using an NMPTC controller that was designed to perform evasive maneuvers on a fixed-wing UAV when confronted by an airborne adversary of *a priori* type. Section II gives a brief description of the details of the fixed-wing aircraft used for flight and testing, and the expression of the UAV's dynamic and kinematic description. Section III describes the rules of the PEG. Section IV gives a description of the encoding of the PEG requirements into

the controller for both pursuer and evader roles. Section V gives the results of some games using the controller, and Section VI presents our conclusions and continuing work.

II. Development and Test Platforms

A. Aircraft details

A Boeing Aircraft Company owned T-33 (originally manufactured by the Lockheed Martin Company) two-seater jet trainer was modified by Boeing for use in the live flight testing in June 2004, and functioned as a UAV surrogate aircraft. The T-33 included a third party autopilot system which did not include airspeed control of the aircraft. This aircraft will hereafter be referred to as the UAV. The UAV had on board a safety pilot who could take control of the aircraft in the event of controller malfunction or poor decision making, and well as for controlling the airspeed of the aircraft based on indicator alerts and a displayed target airspeed given to the pilot to increase/decrease thrust. The route and trajectory of the UAV was controlled by CORBA-based experimental Technology Developer (TD) applications running on a laptop PC with a Linux operating system and Boeing's Open Control Platform (OCP) that was interfaced to the avionics of the aircraft. The TD applications sent the control commands to the avionics pallet that transformed them into autopilot maneuver commands. The state of the UAV, as well as the state of the other aircraft (an F-15 which exchanged state data with the UAV on a wireless link), was available via this avionics interface. The details of the avionics interface, the available state information, and the input controls are given in the rest of this section.

The NMPTC for PEGs described in this paper was one of the TD applications developed and tested for this UAV as part of this SEC Capstone Demonstration program.

B. Software in-the-loop simulation testbed

In order to facilitate development, reliable testing, rapid integration, and a uniform interface independent of operating system, a software in-the-loop simulation (SILS) platform was provided by Boeing. This interface used Boeing's Open Control Platform (COP), a generic (black-box) aircraft simulator called DemoSim and Java based UAV Experiment Controller GUI. The final versions of OCP and the Experiment Controller used were identical to the ones that would be used in the final flight test experiments. This interface provides state information based on the DemoSim model of the UAV, as well as the F-15, to the NPMTC controller and the various other experimental applications that uses it. In addition, a high-level interface to the simulated UAV autopilot is provided that allows the interfacing application to control the rate of change of heading, altitude, and velocity. This included a model of the UAV test pilot's implementation of the airspeed commands from the NMPTC controller, the functionality missing from the third-party autopilot described above. This is just one example of model-mismatch for which the NMPTC controller would have to demonstrate sufficient robustness.

In order to test dynamic PEGs, Boeing modified the SILS to allow for two simulated UAVs to fly against each other, one as evader and the other pretending to be an F-15 pursuer. This allowed for extensive SILS testing of the NMPTC in PEGs with two simulated players.

C. Hardware in-the-loop simulation testbed

All software developed for the test flight was provided to Boeing for integration and testing in their Hardware In-the-Loop Simulator (HILS). The HILS system included an interface to the same avionics system used on the UAV and a proprietary aircraft simulation system. Along with software, the experiment test plans were provided with which Boeing developers tested and verified all the software for the various experiments, including the NMPTC controller for pursuit/evasion.

One of the major limitations on the HILS testbed was again, the lack of a human piloted pursuit aircraft. Additionally, the modification made to the DemoSim aircraft simulation that allowed for the pursuer to use a similar NMPTC controller as the evader aircraft could not be integrated into the HILS system and the pursuer aircraft could only execute a preprogrammed, non-responsive search pattern. This meant that only the basic functionality of the NMPTC controller could be verified. The actual performance of the controller in a real PEG could only be evaluated during the actual flight test.

D. Vehicle modeling

As described above, the UAV was simulated with the DemoSim executable that was provided. This simulator included a high level interface to the control the aircraft through the autopilot, and no model of the aircraft nor the simulator was provided. So, a simplified model was constructed based on the states and inputs available and testing of the DemoSim to determine appropriate parameters for this model.

1) State Vector: The overall system state vector, $\mathbf{x}$, is defined using the following equations.

$$\mathbf{x} = \left[\mathbf{x}^K, \mathbf{x}^D\right] \in \mathbb{R}^{n_x} \tag{1}$$

The vector $\mathbf{x}$, which is the overall system dynamics, is partitioned in (1) into the kinematics (denoted by the superscript K) and system-specific dynamics (denoted by the superscript D) matrices. The kinematics of the system is given as the current state of the system in 3 dimensional space, and with respect to the 3-axis posture of the body.

$$\mathbf{x}^K = [x, y, z, \phi, \theta, \psi] \tag{2}$$

The kinematics is shown in (2), where (x, y, z) is the position of the center of mass in 3 dimensions, ϕ is the roll, θ is the pitch, and ψ is the yaw. The dynamics of the system is given as the time rate of change of the kinematic

state variables, along with incidental changes, which are represented in classical notation as

$$\mathbf{x}^D = [u, v, w, p, q, r]\,, \tag{3}$$

where $u = \dot{x},\ v = \dot{y},\ w = \dot{z},\ p = \dot{\phi},\ q = \dot{\theta},\ r = \dot{\psi}$. Two state variables, the angle of attack and the angle of sideslip, are absent due to the lack of sensors available on the aircraft, and the autopilot's ability to guarantee heading and attitude of the aircraft.

2) Input vector: The input state vector, $\mathbf{u}$, which is the space of possible inputs to the controller to modify the system state, is determined by the autopilot interface through which we have control of the system (as previously described). We define the input state vector as,

$$\mathbf{u} = \left[u_{\dot{v}}, u_{\dot{\psi}}, u_{\dot{z}}\right] \in [-1, 1]^3 \in \mathbb{R}^{n_u} \tag{4}$$

where $u_{\dot{v}}$ is the desired rate of change of airspeed velocity, $u_{\dot{\psi}}$ is the desired rate of change of turn, and $u_{\dot{z}}$ is the desired rate of change altitude. The input space is constrained by the $[-1, 1]^3$ matrix. However, the actual values sent to the input controller are linearly scaled from the range $[-1, 1]$, such that the maximum outputs to the autopilot were as follows: 50ft/s for airspeed, π/50 rad/s for turn rate and 10 ft/s for rate of climb to match the approximate limits of the DemoSim.

In addition, boundaries for the values of the state vector, $\mathbf{x}$, are integrated into the optimization cost function to prevent flight out of the test range and autopilot flight envelope, and to prevent violation of the minimum or maximum safe values for speed and altitude.

III. The Pursuit/Evasion Game

The PEG is an interesting application of NMPTC, as discussed in greater detail in [3], [5]. In this application, a asymmetric PEG is played in the UAV plays the part of the *evader*, and the F-15 plays the part of the *pursuer*, and there are asymmetric objectives for the pursuer and evader. The objective of the evader is to either,

- fly for a predetermined period of time, *T*, since the start of the game; or
- exit the test range at an opposite corner without being targeted by the pursuer.

The objective of the pursuer is to,

- target the evader before the end of the game.

In the symmetric game, the evader win conditions include the targeting of the pursuer.

In these games, targeting is defined by aligning ones heading with that of the other aircraft and locating oneself within a spherical cone (of predefined height, angle, and diameter) aligned with the tail of the other aircraft. This targeting condition is shown in Fig. 2, and in these games a 10° cone of length 3 nautical miles is used. A time limit *T* = 20 minutes is also imposed to prevent a trivial solutions in which the pursuer blocks the exit point, and to reflect the constraints of the flight test experiments.

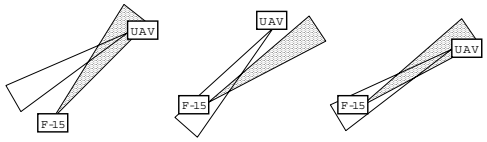


Fig. 2. The targeting rules for these PEGs. In the condition on the left the F15 is not behind UAV, in the middle the F15 not pointed at the UAV, and on the right a successful targeting conditions is shown with the F15 behind AND pointed at the UAV

IV. Controller Design

A. Controller Design for the Evader

NMPC problems, in general, consist of the following steps: 1) solve for the optimal control law starting from the state $\mathbf{x}(k)$ at time k; 2) implement the optimal input $\mathbf{u}(k), \cdots, \mathbf{u}(k+\tau-1)$ for $1 \leq \tau \leq N$; and 3) repeat these two steps at time $k+\tau$. The solution for the optimal control law can be found by formulating a cost function and minimizing it when performing the optimization. The cost function can be composed using specific details of the application, and the designers best knowledge of optimal performance of the object being tracked. This and the computational speed, and method, of the technique are discussed in detail in [2].

Considering the rules of the PEG we were able to design the evader controller by incorporating our desired outcome of the game and encoding some basic aerial tactics as described in [6]. Additionally, state and game constraints necessary for the flight test were added to the basic controller described in [3]. For our final design, we chose the timestep $\tau = 1[s]$, and a lookahead length of $N = 30$ steps. The 30[s] lookahead proved sufficient in simulation to produce good evasion and pursuit tactics in the aircraft. Note, that the 1[s] timestep is used for trajectory planning using NMPTC, while the avionics system performs the low level control at a 0.01[s] timestep.

The desired trajectory of the evader, the location and orientation of the pursuer, the input constraints, and the state constraints, are each a part of the cost function. We set this cost function, J, to be

$$J = \phi(\tilde{\mathbf{y}}_N) + \sum_{k=0}^{N-1} L(\mathbf{x}, \tilde{\mathbf{y}}, \mathbf{u}, \mathbf{d}), \tag{5}$$

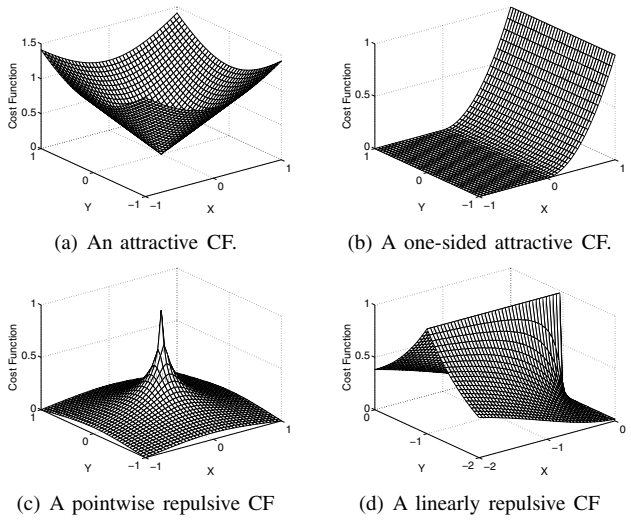
where,

$$\phi(\tilde{\mathbf{y}}_N) \triangleq \frac{1}{2}\left(\tilde{\mathbf{y}}_N^{\mathrm{T}} P_0 \tilde{\mathbf{y}}_N\right), \tag{6}$$

and,

$$\begin{aligned} L(\mathbf{x}_k, \tilde{\mathbf{y}}_k, \mathbf{u}_k, \mathbf{d}_k) \triangleq \\ \frac{1}{2}\tilde{\mathbf{y}}_k^{\mathrm{T}} Q \tilde{\mathbf{y}}_k + \frac{1}{2}\mathbf{x}_k^{\mathrm{T}} S \mathbf{x}_k + \frac{1}{2}\mathbf{u}_k^{\mathrm{T}} R \mathbf{u}_k \\ \frac{1}{2}\mathbf{p_1}_k^{\mathrm{T}} B_1 \mathbf{p_1}_k + \frac{1}{2}\mathbf{p_2}_k^{\mathrm{T}} B_2 \mathbf{p_2}_k \\ + \frac{1}{\left(\mathbf{d}_k^{\mathrm{T}} G \mathbf{d}_k\right)^{\frac{1}{2n_1}}} + \frac{1}{\left(\mathbf{a}_k^{\mathrm{T}} H \mathbf{s}_k\right)^{\frac{1}{2n_2}}} \end{aligned} \tag{7}$$

In these equations, $\mathbf{x}$ is the state vector, and $\mathbf{u}$ is the input vector. The vector $\tilde{\mathbf{y}}$ is the encoding of the error on the current trajectory, and is defined as $\tilde{\mathbf{y}} \triangleq \mathbf{y}_d - \mathbf{y}$, where



(a) An attractive CF. (b) A one-sided attractive CF.

(c) A pointwise repulsive CF (d) A linearly repulsive CF

Fig. 3. Basic forms of the cost functions used by the NMPTC

$\mathbf{y} = C\mathbf{x} \in \mathbb{R}^{n_\mathbf{y}}$. The vector $\mathbf{y}_d$ is the desired trajectory of the aircraft at the given timestep and C acts as a filter to remove elements in $\mathbf{x}$ that are unimportant to the rules of the game. The vectors $\mathbf{p}_1$ and $\mathbf{p}_2$ are the horizontal proximity of the aircraft to the 2 nearest boundaries of the playing area/experimental test range. The vector $\mathbf{d}$ is the proximity danger vector between the evader and its adversary, and $\mathbf{a}$ is the AOT of the pursuer with respect to the evader.

The Q, S, R, B_1, B_2, G, and H square matrices each serve as weighting factors in the cost function. By modifying their relative values, it is possible to give more "weight" to certain portions of the cost function. The details of the basic parameters and the process of determining the values for these weights is described in [3]. Some additions were made to the cost function leading up to the final flight test, which are described below.

The choice of which boundaries to use for the computation of vectors $\mathbf{p}_1$ and $\mathbf{p}_2$ is made by computing the distance from all boundaries (five in these experiments, as illustrated in Fig. 4) and choosing the two nearest. This method is clearly effective only inside a non-convex area. Furthermore, the B_1 and B_2 matrices are set to zero when the aircraft is further than a preset distance from the respective closest range boundary.

Similarly, when the pursuer is more that a preset distance from the evader, the matrix H is set to zero and has no affect on the NMPTC controller.

The first three terms in (7) are quadratic and zero-mean attractive functions which penalize deviation from the zero valued inputs, as shown in Fig. 3(a). The fourth and fifth functions (of $\mathbf{p}_1$ and $\mathbf{p}_2$) are one sided versions of the previous type such that they penalize deviation in one direction only, as shown in Fig. 3(b). The last two functions are quadratic and zero-mean repulsive functions, meaning that they penalize inputs that approach zero values. The first of those two is a point repulsive function, the second is repulsive along the axis extending behind the aircraft as a function of the angle from zero, as shown in Fig. 3(c) and Fig. 3(d) respectively. The forms of these functions are chosen to allow for the easy computation of their derivatives necessitated by the optimization procedure. Note also that the last two (repulsive functions) contain the additional parameters n_1 and n_2 in the order of the denominator polynomial functions which affects the shape of the repulsive functions.

The cost-function is optimized using the iterative technique described in [4], [7], which requires the calculation of the derivatives of the vehicle dynamics with respect to both the state and input vectors. Since only the DemoSim interface was available and no mathematical description of the vehicle dynamics, a simplified model was used with the Eulerian equations of motion, input latency gains and limits on the states to capture the "projected" values for the state of the evader (and pursuer) in the predictive component of the controller.

In order to reduce the computational burden of the nonlinear gradient-descent optimization, the result from the previous time step is used to initialize the optimizer [7], [3], and a limit on the number iterations is imposed. The second measure may result in a sub-optimal solution being found, however this is generally during a periods rapid change during which (particularly in a PEG) rapidly determined, sub-optimal decisions are preferable to waiting for optimal solutions based on estimates of future states. Such estimates of the future are also made inaccurate by the changing and unpredictable actions of the other aircraft.

Other methods for optimizing the cost function can be considered, including off-line multiparametric quadratic programming [8]. However, off-line methods restrict how the cost function can be modified as the situation changes, and that in the case of [8] they require linear constraints and a quadratic cost function, which would preclude several of the functions used here. Also the dimensionality of state, input and constraint vectors would lead to a significant storage issue for the solution computed off-line.

B. Controller Design for the Pursuer

An NMPTC controller design for the pursuer aircraft was partially necessitated by the need for an opponent for the evader in the SILS testbed, and also to enable the UAV to assume the role of pursuer.

When in pursuer mode, the UAV used the same controller as in evader mode, however with different matrix values for Q, S and R, the other 4 terms in (7) omitted and the computation of $\tilde{\mathbf{y}}$ was modified for pursuit. Additionally, the pursuers $\mathbf{y_d}$ used to compute $\tilde{\mathbf{y}}$ depended on the pursuers position and relative direction compared to the evader, specifically to offset its approach prior to turning into a position on the tail of the evader, as discussed in [6]

In the symmetric version of the PEG, in which the evader is able to target the pursuer to win the game, the evader checks its angle of tail (AOT) and angle of nose (AON) conditions (see V-B below), and if they are within a certain range simultaneously it will switch to the pursuer NMPTC

parameters until such time as the favorable conditions are lost. There is sufficient deadband in these switch conditions to prevent chattering behavior.

V. Experimental Results

A. Implementation

The NMPTC algorithm and the PEG were implemented in C++ and run in a Windows environment. The evader aircraft's NMPTC controller was tuned by building it up, component by component of (7) to provide for a successful game outcome in terms of exiting the playing area and avoiding the pursuer aircraft. The trajectory, state and input matrices Q, S, and R were first tuned to ensure that aircraft would follow a set trajectory as both evader and pursuer. Then the evader matrices G and H were tuned along with exponential terms n_1 and n_2 for the evader to prevent the pursuer from taking up a targeting position as per the rules of the PEG. Finally the boundary matrices B_1 and B_2 were tuned to keep the aircraft within the boundaries of the game. This did require several iterations to fine-tune the behaviors.

The pursuer algorithm was tuned to close with the evader using its Q matrix and the choice of $\mathbf{y}_d$ as a path toward the predicted position of the evader. Both aircraft controllers used the S and R matrices to constrain the states and inputs.

In the SILS experiments, two instances of the aircraft were simulated, one as the pursuer and one as the evader. For these examples, the performance characteristics of the two aircraft were identical, as was the case in all the simulations leading up to the final test flight due to the constraints of the DemoSim interface.

In the HILS experiments and in the flight test experiments only one aircraft used the NMPTC algorithm. A dummy aircraft was used in the HILS and the initial flight tests as the other other aircraft to verify the software for flight testing. In the remaining flight tests the other aircraft was an F-15 flown by a USAF pilot.

B. Simulation Results

The experiment controller used in the simulation and flight test experiments is shown in Fig. 4 at the start of a SILS experiment. In this case the evader has entered from the west and it trying to reach the 'End Zone' to win the game. The pursuer was in the southern half of the area, turned to engage the evader and then turned to get behind it.

Fig. 5 shows the outcome of this experiment after 30 minutes of simulation time. The tracks of the two aircraft can be seen as the evader carries out a series of maneuvers to prevent the F-15 from adopting a targeting position behind it. It can also be seen that the UAV tries to break contact and head for the 'End Zone' when it feels safe to do so, although it only gradually makes its way in that direction. The effect of the game boundary constraints can also be seen as the UAV's evasive maneuvers bring it close to the southeast boundary. The UAV is forced back into the game area at these times.

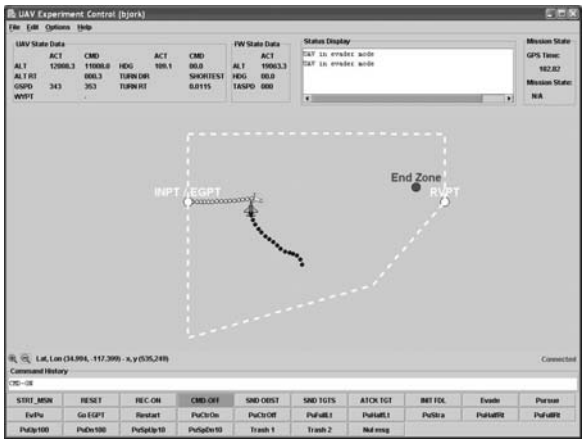


Fig. 4. The UAV Experiment Controller: UAV is yellow, F-15 is blue

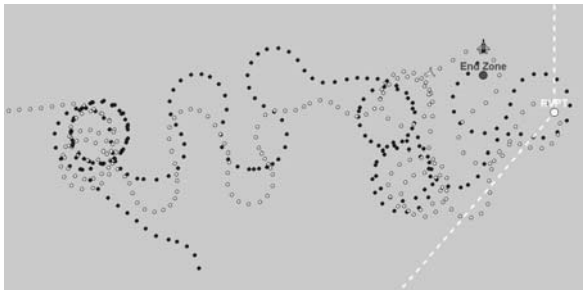


Fig. 5. A 30 minute simulation game: the pursuer (blue trace) started near the bottom, the evader (yellow trace) started on the left

In this game, the pursuer would have won by being in the 10° AOT cone of the evader (i.e. within 10° of directly behind the evader) while *at the same time* having the evader within the 10° angle off nose (AON) cone of itself (i.e. having the evader within 10° of directly in front).

C. Flight test validation

The NMPTC controller was ported to Linux and provided to Boeing for final integration, hardware testing in the HILS and experimental flight tests. These flight tests were carried out as part of the SEC Capstone Demonstration at Edward's AFB during the weeks of June 14-25, 2004. This experiment was run 4 times as part of three different sorties, in which

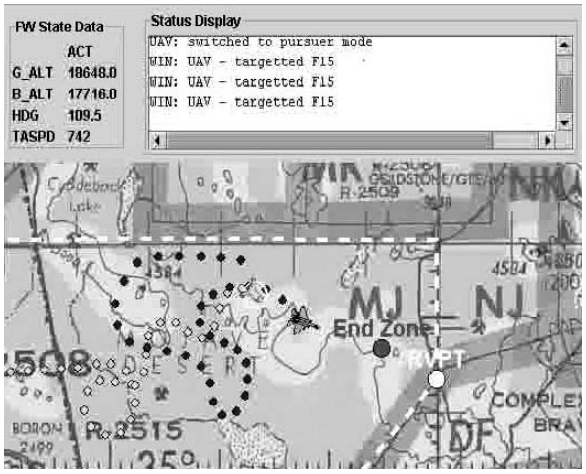


Fig. 6. Flight test experimental results: A symmetric PEG in which the UAV (yellow trace) has just targeted the F-15 (blue trace)

the NPMTC controller and the Boeing platform performed as expected from the simulations, despite significant wind and other conditions.

The first two test flights were flown without the F-15 in order to verify the system. In this case, the same method was used as on the HILS in which the F-15 was simulated, and as in the HILS it could only follow a preprogrammed flight path. In these cases the simulated F-15 flew a figure-8 pattern through which the UAV successfully flew to its desired destination virtually unimpeded.

In the third experiment, the UAV was set to only evade. In this case the F-15 was able to get behind the UAV, although it did not succeed in targeting the UAV for several minutes.

In the fourth and final experiment, the UAV was able to switch to pursuer mode if it detected favorable conditions to do so. In this experiment, shown in Fig. 6, the paths of the aircraft can be seen. The F-15 was carrying out a search pattern as the UAV approached. The UAV then detected a advantageous condition and switched to pursuer mode and successfully targeted the F-15. The experiment was allowed to continued after this (not shown) and the F-15 did shake the UAV off its tail and managed to get behind it as the UAV switched back to evader mode and tried to reach the 'End Zone'.

The interesting variable in these experiments was the behavior of the F-15 pilot, which was highly non-deterministic. The pilot also provided valuable feedback and commented very favorably on the behavior of the UAV and the NMPTC controller. Paraphrasing the F-15 pilot after one experiment: the UAV did precisely what one is taught to do in flight school for that situation (in which the F-15 got behind the UAV and tried to adopt a targeting position).

VI. Conclusions and Future Works

A. Conclusions

In this paper we have presented experimental results of the effectiveness of the NMPTC approach to the PEG for fixed-wing aircraft in a time-critical application.

By using the NMPTC approach, rapid computations can be performed, and (given accurate dynamics) the true advantages of autonomy can be encoded using the concepts of competitive games. By providing this autonomous mode (e.g., evader) to a UAV operator, it is possible for a remote operator to relinquish control of the vehicle in time-critical situations, to allow the intelligent controller to serve as a surrogate that incorporates the same theories and behaviors of the pilot. Because the safety and functionality constraints of the aircraft are encoded into the cost function, the UAV is not endangering itself or its environment.

The simulation results show that the encoding of the game into the cost function was successful and these results were validated in actual flight tests on a T-33 UAV surrogate in PEGs with a piloted F-15. NMPTC had not yet been demonstrated on fixed-wing aircraft for the pursuit/evasion problem, and this work shows that this method is appropriate when providing input to an autopilot interface.

B. Future Works

With this flight test validation, we will be able continue developing and testing PEGs and the use of NMPTC in the simulation environment with high confidence in their extension to real flight situations.

The application of NMPTC will continue as we encode the notion of a symmetric PEG into the cost function. This will enable an aircraft to switch roles in the game, thus switching its goals (and associated costs) at runtime. The encoding of this decision to switch roles as part of the cost-function is an exciting possibility.

Currently the game can only be played with the high-overhead avionics interface to the T-33 jet, provided by Boeing. The execution of the code, however, is extremely fast, and future work will involve the hardware implementation in avionics interface to low-cost fixed-wing UAVs.

Proven tactics for evasion and pursuit of fighter aircraft, as discussed in [6], would be useful if encoded into the cost function to encourage these behaviors. Future work into iteratively deriving the weights and values of the cost function matrices could be employed to provide this emergent behavior. Both works are slated for the future, along with simulations that show such behavior emerges from the controller definition.

Finally, the overhead associated with creating a new NMPTC controller is substantial and it would be useful to have a high-level understanding of the controller, as well as a way to generate the controller from this higher level. Providing an abstraction for this level of detail is a useful future research area.

VII. Acknowledgments

This research funded with the support of the DARPA Software Enabled Control (SEC) program, under contract number DARPA SEC F33615-98-C-3614.

References

[1] AFMC Public Affairs, "Global Hawk UAV supports OEF recon," AFMC News Service Release 1217, December 2002.

[2] H. J. Kim, D. H. Shim, and S. Sastry, "Nonlinear model predictive tracking control for rotorcraft-based unmanned aerial vehicles," in *American Control Conference*, May 2002.

[3] J. Sprinkle, J. M. Eklund, H. J. Kim, and S. Sastry, "Encoding aerial pursuit/evasion games with fixed wing aircraft into a nonlinear model predictive tracking controller," in *Conference on Decision and Control*, December 2004.

[4] G. J. Sutton and R. R. Bitmead, *Nonlinear Model Predictive Control*, ser. Progress in Systems and Control Theory. Basel-Boston-Berlin: Birkhäuser Verlag, 2000, vol. 26, ch. Computational Implementation of Nonlinear Predictive Control on a Submarine, pp. 461–471.

[5] I. Mitchell, "Application of level set methods to control and reachability problems in continuous and hybrid systems," Ph.D. dissertation, Stanford University, Aug. 2002.

[6] R. L. Shaw, *Fighter Combat: Tactics and Maneuvering*. Annapolis, Maryland: United States Naval Inst., 1985.

[7] H. J. Kim, D. H. Shim, and S. Sastry, "Decentralized nonlinear model predictive control of multiple flying robots in dynamic environments," in *Conference on Decision and Control*, Dec 2003.

[8] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit solution of model predictive control via multiparametric quadratic programming," in *American Control Conference*, June 2000.