JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 25, No. 5, September–October 2002

Nonlinear Receding Horizon Control of an F-16 Aircraft

Raktim Bhattacharya* and Gary J. Balas†
University of Minnesota, Minneapolis, Minnesota 55455
and
M. Alpay Kaya‡ and Andy Packard§
University of California, Berkeley, Berkeley, California 94720

The application of receding horizon control (RHC) with the linear, parameter varying (LPV) design methodology to a high-fidelity, nonlinear F-16 aircraft model is demonstrated. The highlights are 1) use of RHC to improve upon the performance of a LPV regulator; 2) discussion on details of implementation such as control space formulation, tuning of RHC parameters, computation time and numerical properties of the algorithms; and 3) simulated response of nonlinear RHC and LPV regulator.

Nomenclature

$epos$	=	east position, ft
h	=	altitude, ft
$npos$	=	north position, ft
p	=	roll rate, deg/s
q	=	pitch rate, deg/s
r	=	yaw rate, deg/s
T	=	thrust, lb
V_t	=	velocity, ft/s
α	=	angle of attack, deg
β	=	sideslip angle, deg
δ_a	=	aileron, deg
δ_e	=	elevator, deg
δ_r	=	rudder, deg
θ	=	pitch angle, deg
ϕ	=	bank angle, deg
ψ	=	yaw angle, deg

I. Introduction

RECEDING horizon control, also known as model predictive control, has been popular in the process control industry for several years.[1,2] It is based on the simple idea of repetitive solution of an optimal control problem and updating states with the first input of the optimal command sequence. The repetitive nature of the algorithm results in a state-dependent feedback control law.

The attractive aspect of this method is the ability to incorporate state and control limits as hard or soft constraints in the optimization formulation. When the model is linear, the optimization problem is quadratic if the performance index is expressed via a $\mathcal{L}_2$-norm, or linear if expressed via a $\mathcal{L}_1/\mathcal{L}_\infty$-norm. Issues regarding feasibility of on-line computation, stability, and performance are largely understood for linear systems and can be found in Refs. 3–6.

The focus of this paper is on the application of receding horizon control (RHC) techniques to nonlinear systems. Specifically, the use of affine, linear, parameter varying control[7] to derive a control Lyapunov function (CLF), which is used as a terminal penalty function in the RHC framework. Similar work has been done by Primbs[8] and Jadbabaie et al.[9] on the Caltech Ducted Fan,[10] where combination of off-line linear, parameter varying (LPV) design and on-line RHC optimization led to better regulation of the system states. The approach used by Primbs imposes path and terminal constraints to guarantee stability. Satisfying path constraints on state and control trajectories during optimization is however computationally demanding. This is not so lucrative if we are to explore any possibility of real-time implementation of the RHC algorithm. In the work done by Jadbabaie et al.,[9] stability is guaranteed by choosing a suitable terminal cost. No constraints are necessary to guarantee stability. We chose this approach to formulate our RHC problem for the nonlinear F-16 model because of the simplicity of formulation and also because unconstrained optimization is faster than constraint optimization.

The receding horizon control problem investigated in this paper is a regulation-based problem. We are interested in regulating state perturbations of F-16 to a target set, namely the origin. The performance of the controller is measured in terms of the value of a objective function, which depends on the state and control trajectories. A LPV controller is designed for the F-16 aircraft, and a CLF is obtained from it. The CLF is used as a terminal cost in the formulation of the optimization problem for receding horizon control.

The purpose of this paper is to demonstrate an aerospace application of nonlinear RHC algorithm and also discuss the implementation issues in detail. The latter is missing in most publications in this field. This paper goes through an overview of the nonlinear RHC theory, F-16 model details, LPV regulator design, optimization problem setup for RHC, formulation of control space, numerical issues in nonlinear simulation, hardware and software details. This paper is aimed to answer most questions that arise in a typical nonlinear RHC implementation.

II. Stability and Performance of Nonlinear RHC

Nonlinear trajectory optimization problem can be posed as (p. 131 in Ref. 11)

$$J = \min_{u(.)} \Phi(x,t)\Big|_{t=t_0+T} + \int_{t_0}^{t_0+T} L(x,u)\,\mathrm{d}t \qquad (1)$$

with system dynamics, initial condition, and terminal boundary condition defined as

$$\dot{x} = f(x,u), \qquad x(t_0) = x_0, \qquad \Psi(x,t)|_{t=t_0+T} = 0$$

Solving the associated partial differential equation for the optimal return function of the optimization defined by Eq. (1) can be computationally intractable, even for a moderate number of states. Hence the optimal control trajectories for RHC are usually obtained by solving the simpler Euler–Lagrange type of optimization. Because this is based on calculus of variation, its solution is locally optimal

Received 19 April 2001; revision received 15 January 2002; accepted for publication 12 March 2002.

*Graduate Research Assistant, Department of Aerospace Engineering and Mechanics; raktim@aem.umn.edu.
†Associate Professor, Department of Aerospace Engineering and Mechanics; balas@aem.umn.edu.
‡Graduate Student Researcher, Department of Mechanical Engineering; alpay@jagger.me.berkeley.edu.
§Professor, Department of Mechanical Engineering; pack@me.berkeley.edu.

and is valid only for a single initial condition. The desired feedback nature of the receding horizon control law is achieved by resolving the optimization at every encountered state.

The unconstrained finite horizon optimization does not guarantee stability. In Ref. 8, Sec. 4.2.2, p. 36, an example is available that clearly demonstrates this fact. However, if the optimization is carried out over an infinite horizon the controller will inherit stability guarantees and performance properties enjoyed by the infinite horizon solution. Unfortunately, in reality one cannot optimize over an infinite horizon for practical reasons. There are difficulties in proving stability for RHC feedback laws also. These arise because the solution to each receding horizon optimization is specific to a single point in the state space, and typical stability theories consider system behavior within a region. To get around these issues, the stability of finite horizon RHC algorithm is guaranteed by imposing constraints on the optimization.

The first method to guarantee stability[12,13] imposes a terminal boundary condition on state as

$$x(t_0 + T) = 0$$

Because a nonlinear optimization problem with equality terminal constraint is computationally demanding, in Ref. 14 closed-loop stability is ensured by imposing a terminal inequality constraint. The constraint requires $x(t_0 + T)$ to enter a suitable neighborhood of the origin. Once such a neighborhood is reached, the receding horizon controller is switched to a locally stabilizing linear controller. A different approach has been proposed in Ref. 15 where the receding horizon controller is obtained by solving a finite horizon problem with quadratic terminal state penalty

$$\Phi(x, t)|_{t_0+T} = ax^T Px|_{t_0+T}$$

for some a, $P > 0$. In the more recent work of De Nicolao et al.,[16] stability of the receding horizon controller is guaranteed by using a possible nonquadratic terminal penalty, which is the cost incurred if a locally stabilizing linear control law is applied at the end of the horizon. The linear control law ensures exponential stability of the equilibrium point at the origin, and it is assumed that the region of attraction of the linear controller is reachable within the horizon length. This is different from Ref. 14 in that the linear control law is never applied, it is used just to compute the terminal cost. In another method proposed by Primbs,[8] first a globally stabilizing control law is achieved by finding a global CLF. Stability of the receding horizon controller is ensured by including state constraints that make the derivative of the CLF negative along the receding horizon trajectory and also makes the value of the CLF at the end of the horizon less than that when the controller obtained from CLF is applied. Unfortunately, the path and terminal constraint on the optimization makes it computationally intensive.

As mentioned earlier, our implementation of receding horizon control for the F-16 model is based on the work by Jadbabaie et al.[17] This method is a hybrid of the methods proposed by Primbs[8] and De Nicolao et al.[16] As in Ref. 8, it is assumed that a suitable CLF already exists. The CLF is used to replace the state inequality constraint with a terminal cost. The terminal cost is the cost incurred if the controller from the CLF is applied at the end of the horizon. We derived the CLF from a LPV regulator. In theory, as in Ref. 16, the CLF-based controller is never implemented; it is used only to compute the terminal cost. However, because of the limitations in the optimization software we used we needed the LPV regulator to prestabilize the system. The optimization software, NPSOL,[18] has numerical problems with unstable systems. It uses shooting method to search for an optimal direction, and a candidate solution can potentially destabilize an unstable system.

III. F-16 Aircraft Model

The nonlinear F-16 model used in receding horizon control is available in Ref. 19, which we shall represent as

$$\dot{x}_{\mathrm{NL}} = f(x_{\mathrm{NL}}, \delta) \tag{2}$$

The mathematical model uses the wind-tunnel data from the NASA Langley wind-tunnel tests on a scale model of an F-16 aircraft.[20] The aerodynamic data are valid up to Mach 0.99, angle-of-attack range of -10 deg $\leq \alpha \leq 45$ deg and sideslip-angle range -30 deg $\leq \beta \leq 30$ deg. The wind-tunnel tests were conducted on sufficiently close points to capture the nonlinear behavior of the aerodynamic force and moment coefficient. Aerodynamic data for the intermediate points are linearly interpolated. The states $x_{\mathrm{NL}} \in \mathcal{R}^{12}$ and controls $\delta \in \mathcal{R}^4$ in the model are defined as

$$x_{\mathrm{NL}} = [npos, epos, h, \phi, \theta, \psi, V_t, \alpha, \beta, p, q, r]$$

$$\delta = [T, \delta_e, \delta_a, \delta_r]$$

Actuators for the control surfaces and engine are modelled as first-order systems, details of which are discussed later in the paper.

IV. LPV Modelling of F-16

Based on aerodynamic data in Ref. 20, the control variables in Eq. (2) enter affinely in δ_a, δ_r, and T, though not in δ_e. To derive an LPV model[21] of the nonlinear equation of motion for F-16, it is necessary that all of the controls be in affine form. This is achieved by transforming (δ_e, T) into synthetic inputs (u_1^S, u_2^S) with $\xi: \mathcal{R}^2 \to \mathcal{R}^2$, so that the equation of motion, after defining $\delta_{a,r} := [\delta_a \;\; \delta_r]^T$ and $u^S := [u_1^S \;\; u_2^S]^T$, can be written as

$$\dot{x}_{\mathrm{NL}} = h(x_{\mathrm{NL}}) + g(x_{\mathrm{NL}}) \begin{bmatrix} \delta_{a,r} \\ u^S \end{bmatrix} \tag{3}$$

A. Synthetic Inputs

The nonaffine terms in Eq. (2) directly affect the states (V_t, α, β, q). If we rearrange the state dynamics of these four states into affine and nonaffine terms, we can write it as

$$\begin{Bmatrix} \dot{V}_t \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{q} \end{Bmatrix} = \lambda(x_{\mathrm{NL}}) + \zeta(x_{\mathrm{NL}})\delta + \eta(V_t, \alpha, \delta_e, T) \tag{4}$$

where $x_{\mathrm{NL}} \in \mathcal{R}^{12}$ is the state vector and $\delta \in \mathcal{R}^4$ is the control vector of the system. The nonaffine terms are represented as $\eta(V_t, \alpha, \delta_e, T)$. It can be observed that the elevator and thrust inputs do not enter linearly. Hence, their effect on the state derivatives need to be approximated through the use of synthetic inputs. We seek to isolate their effect into a "gain" term $[M(V_t, \alpha)]$, as follows:

$$\underbrace{\eta(V_t, \alpha, \delta_e, T)}_{4\times 1} \approx \underbrace{M(V_t, \alpha)}_{4\times 2} \underbrace{u^S(V_t, \alpha, \delta_e, T)}_{2\times 1} \tag{5}$$

This separation results in the gains on the "synthetic inputs" $u^S(V_t, \alpha, \delta_e, T)$ being constant at a given velocity and angle of attack, which in turn means that the LPV model will not need to be scheduled on either of these inputs. For continuous functions the approximation in Eq. (5) can be achieved by

$$\min_{M, u^S} \int_{V_t, \alpha} \int_{\delta_e, T} \|\eta - Mu^S\|^2 \, \mathrm{d}(\delta_e, T) \, \mathrm{d}(V_t, \alpha) \tag{6}$$

whereas for a gridded function this is equivalent to minimizing the following Frobenius norm:

$$\min_{rk(H)=r} \|X - H\|_F^2 \tag{7}$$

where X and H represent the function η and its approximation Mu, respectively, evaluated at the grid points. The minimization problem in Eq. (7) can be solved by singular value decomposition. The rank of H should equal the number of synthetic inputs; $r = 2$ in this case. The synthetic input vector u^S is in radians.

B. Inverse Table

Using the synthetic input formulation means that the LPV regulator is designed using the (V_t, α)-dependent matrix gain $M(V_t, \alpha)$, as defined in the preceding subsection. Thus, the control commands are not in term of elevator and thrust, but in terms of synthetic inputs. This necessitates the ability to calculate (δ_e, T) from a given (V_t, α, u^S). The synthetic input inverse table, denoted by ξ^{-1}, is used to recover the elevator and thrust values from the synthetic input values.

At a given (V_t, α) the synthetic inverse table is made by specifying grid points (i, j) in u^S space and searching over (δ_e, T) to minimize

$$c^k\left(V_t, \alpha, u^S_{1\,i}, u^S_{2\,j}\right) = \left[u^S_{1\,i} - u^S_1\left(V_t, \alpha, \delta^k_e, T^k\right)\right]^2 + \left[u^S_{2\,j} - u^S_2\left(V_t, \alpha, \delta^k_e, T^k\right)\right]^2$$

where c^k is the cost at the kth iteration, resulting in $u^I(V_t, \alpha, u^S)$. This is repeated for each (V_t, α) grid point. The inverse table ξ^{-1} is also a function of altitude. Because we use this model to compute optimal command trajectories for receding horizon control, it is necessary that ξ^{-1} is continuous and sufficiently smooth in the region of interest.

C. LPV Model of F-16

Assuming first-order dynamics for each actuator, the LPV model derived for the F-16 has 12 states (eight flight dynamics and four actuator) and four inputs,

$$x = \begin{bmatrix} V_t & \alpha & \beta & p & q & r & \phi & \theta & \delta_a & \delta_r & \delta^S_1 & \delta^S_2 \end{bmatrix}^T$$

$$u_{\mathrm{LPV}} = \begin{bmatrix} u_a & u_r & u^S_1 & u^S_2 \end{bmatrix}^T$$

both of which are centered about their trim values $(\bar{x}, \bar{u}_{\mathrm{LPV}})$, and the LPV model is of the form

$$\dot{x} = A_{\mathrm{LPV}}(\rho)x + B_{\mathrm{LPV}}(\rho)u_{\mathrm{LPV}} \tag{8}$$

Gain scheduling is done on parameter $\rho \in \mathcal{P} \subset \mathcal{R}^3$, defined as

$$\rho = \begin{Bmatrix} V_t \\ \alpha \\ \beta \end{Bmatrix} \tag{9}$$

It is assumed that the set $\mathcal{P}$ is compact and $\dot{\rho}$ is bounded (that is, $\dot{\rho}_{\min} \le \dot{\rho} \le \dot{\rho}_{\max}$). The matrix functions $A_{\mathrm{LPV}}(\rho)$, $B_{\mathrm{LPV}}(\rho)$ are defined as

$$A_{\mathrm{LPV}}(\rho): \mathcal{P} \to \mathcal{R}^{12\times 12}, \qquad B_{\mathrm{LPV}}(\rho): \mathcal{P} \to \mathcal{R}^{12\times 4}$$

Because the scheduling parameters are also states of the system, the LPV model is actually a quasi-LPV model. A quasi-LPV system is a special class of LPV system where the system varies as functions of state variables and exogenous variables. The state variables and the exogenous variables are required to be continuous functions of time and measurable in real time. The LPV full-state control design algorithm cannot take advantage of the fact that the scheduled variables are also the system states. Therefore, the full-state LPV regulator may be conservative.

A cost function for Eq. (8) penalizing not only the states and inputs, but the actuator rates, is considered here. Clearly, this will result in the state cost being coupled to the input cost; define such a function as the integral of

$$c := \begin{bmatrix} x \\ u_{\mathrm{LPV}} \end{bmatrix}^T \underbrace{\begin{bmatrix} Q_{\mathrm{LPV}} & S \\ S^T & R_{\mathrm{LPV}} \end{bmatrix}}_{Z} \begin{bmatrix} x \\ u_{\mathrm{LPV}} \end{bmatrix} \tag{10}$$

By defining $G := R^{-1}_{\mathrm{LPV}} S^T$ and the transformation $u := u_{\mathrm{LPV}} + Gx$, Eq. (10) can be rewritten as

$$c = \begin{bmatrix} x \\ u \end{bmatrix}^T \begin{bmatrix} Q_{\mathrm{LPV}} - SG & 0 \\ 0 & R_{\mathrm{LPV}} \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \tag{11}$$

The weighting matrices for the uncoupled cost are $Q := Q_{\mathrm{LPV}} - SG$ and $R := R_{\mathrm{LPV}}$. Defining $A := A_{\mathrm{LPV}} - B_{\mathrm{LPV}}G$ and $B := B_{\mathrm{LPV}}$, the dynamics are represented by

$$\dot{x} = Ax + Bu \tag{12}$$

A full-state LPV feedback regulator is designed for Eq. (12) in the following section.

V. Synthesis of CLF and LPV Regulator

A. Terminal Cost Upper Bound

The infinite horizon optimal control problem is defined here: given $Q \ge 0$ and $R > 0$,

$$J_\infty(x_0) := \min_{u(\cdot)} \int_{t_0}^{\infty} (x^T Qx + u^T Ru)\,\mathrm{d}t$$

subject to

$$\dot{x} = f(x, u), \qquad x(t_0) = x_0$$

The RHC approach seeks to solve the preceding problem by recursively solving the following finite horizon optimal control problem (FHOCP): defining $\phi[x(t_0 + T)]$ as the cost from $t = t_0 + T$ onward, the FHOCP is

$$J_\infty(x_0) = \min_{u[t_0, t_0+T]} \int_{t_0}^{t_0+T} (x^T Qx + u^T Ru)\,\mathrm{d}t + \phi[x(t_0 + T)]$$

We seek to calculate an upper bound to $\phi[x(t_0 + T)]$, the "cost-to-go." Assume there exists a matrix function $P > 0$ such that $\forall \rho(t) \in \mathcal{P}$, $P(\rho) > 0$, and $\gamma > 0$,

$$\sum_k^M \bar{v}_k \frac{\partial P}{\partial \rho_k} + A^T(\rho)P(\rho) + P(\rho)A(\rho) - P(\rho)B(\rho)R^{-1}B^T(\rho)P(\rho) + Q \le -\gamma I \tag{13}$$

Note that Eq. (13) represents not one but 2^M inequalities, each one containing a different combination of upper and lower bounds on v; by Eq. (9) $M = 3$. Define

$$V(x, t, \rho) := x^T P[\rho(t)]x$$

Along trajectories of Eq. (12),

$$\dot{V} = x^T \left[A^T(\rho)P(\rho) + P(\rho)A(\rho) + \sum_k \dot{\rho}_k \frac{\partial P}{\partial \rho_k}\right] x + u^T B^T(\rho)P(\rho)x + x^T P(\rho)B(\rho)u$$

Given Eq. (13), we can bound $\dot{V}$,

$$\dot{V} \le x^T[-\gamma I + P(\rho)B(\rho)R^{-1}B^T(\rho)P(\rho) - Q]x + u^T Ru - u^T Ru + u^T B^T(\rho)P(\rho)x + x^T P(\rho)B(\rho)u$$

$$\le \left\| R^{\frac{1}{2}}u + R^{-\frac{1}{2}}B^T(\rho)P(\rho)x \right\|_2^2 - x^T Qx - u^T Ru$$

Integration of the second line results in

$$V[x(\infty)] - V[x(t_0)] \le \left\| R^{\frac{1}{2}}u + R^{-\frac{1}{2}}B^T(\rho)P(\rho)x \right\|_2^2 - \int_{t_0}^{\infty} [x^T Qx + u^T Ru]\,\mathrm{d}t$$

from which it is clear that choosing

$$u := -R^{-1}B^T(\rho)P(\rho)x \tag{14}$$

minimizes the right-hand side. This indeed results in $x(t) \to 0$ and $V(\infty) = 0$, which means that

$$\int_{t_0}^{\infty} (x^T Qx + u^T Ru)\,\mathrm{d}t \le V[x(t_0)]$$

that is, $x^T(t)P[\rho(t)]x(t)$ is an upper bound on the terminal cost of the FHOCP. The RHC cost function follows:

$$J[x(t_0)] \le \min_{u(\cdot)} \int_{t_0}^{t_0+T} (x^T Qx + u^T Ru)\,dt + x^T(t_0+T)P[\rho(t_0+T)]x(t_0+T)$$

B. LPV Regulator and CLF

The problem formulation that was used to obtain the CLF and LPV regulator is shown here. Assuming $\bar{x}_i$ represents some typical initial conditions, a specific $P(\rho)$ can be found by the following minimization:

$$\min \sum_i^N \bar{x}_i^T P \bar{x}_i$$

Define $W := [\bar{x}_1 \cdots \bar{x}_N]$, then

$$\mathrm{tr}[W^T P W]$$

is equal to the summation in the preceding minimization. Because the affine matrix inequalities (AMI) in Eq. (13) are not convex in P, define $X := P^{-1}$. Pre- and postmultiplying Eq. (13) by P^{-1}, noting that

$$\frac{\partial X}{\partial \rho_k} = -P^{-1}(\rho)\frac{\partial P}{\partial \rho_k}P^{-1}(\rho)$$

results in

$$\sum_k -\bar{v}_k \frac{\partial X}{\partial \rho_k} + X(\rho)A^T(\rho) + A(\rho)X(\rho) - B(\rho)R^{-1}B^T(\rho) + X(\rho)QX(\rho) \le 0$$

Applying Schur complement and defining $C := Q^{\frac{1}{2}}$ yields the following problem formulation:

$$\min_{Y,X\ge 0} \mathrm{tr}[W^T Y W]$$

subject to

$$\begin{bmatrix} Y(\rho) & I \\ I & X(\rho) \end{bmatrix} \ge 0$$

$$\begin{bmatrix} -\sum_{k=1}^{3} \bar{v}_k \frac{\partial X}{\partial \rho_k} + X(\rho)A^T(\rho) & X(\rho)C^T \\ +A(\rho)X(\rho) - B(\rho)R^{-1}B^T(\rho) & \\ CX(\rho) & -I \end{bmatrix} < 0$$

Solution of this problem yields the positive definite $X(\rho) \in \mathcal{R}^{12\times 12}$, which is used to calculate the control input (14) and the CLF (Ref. 9):

$$V(x) = x^T X^{-1}(\rho)x \tag{15}$$

VI. RHC Problem Formulation

For receding horizon control the optimization problem is set up as follows:

$$J = \min_{u(.)} \int_{t_0}^{t_0+T} L(x,u)\,dt + x^T X^{-1}(\rho)x|_{t_0+T} \tag{16}$$

subject to

$$\dot{x} = f(x,u), \qquad x(t_0) = x_0$$

where $L(x,u)$ is a positive definite function of $x(.)$ and $u(.)$.

The F-16 equations of motion are used in Eq. (16) to compute the optimal control trajectory and in Eq. (2) to update the systems states with that control. The accuracy of the model determines the accuracy of the state and control trajectories obtained from the simulations. The trajectories obtained from a LPV model will be different from that of the nonlinear model as LPV systems are not exactly same as their nonlinear parents. Several assumptions were made in deriving the LPV model that RHC algorithm can potentially exploit. These defects could yield unrealistic optimal control and state trajectories.

For example, in the formulation of the F-16 quasi-LPV model the direction of gravity was held constant and α was set equal to θ. These were obviously simplifying assumptions that eliminated physical constraints from the problem. Using the LPV model as the simulation "truth" model in the RHC optimization led to α and β state trajectories that instantaneously went from their nonzero perturbed state to zero with minimal control activity. Hence, the full nonlinear simulation was henceforthused in the RHC optimizations.

VII. Formulation of Control Space for Optimization

A common practice for solving optimal control problems is to convert them into parameter optimization problems. There are several methods available for conversion.[22] Numerical integration, collocation, direct transcription, and differential inclusion are examples of these conversion methods. Most of these techniques are similar in nature and differ by what is guessed (state or control), how the integration is carried out (implicit or explicit), and the order of integration. This process of conversion is called "suboptimal" control because search is restricted to a particular subspace of the finite dimensional control space. For our RHC simulation we have used B-splines[23] to formulate our control space.[24] For a rth-order B-spline the control vector is defined as

$$u(t) = \sum_{k=1}^{N+r-1} \alpha_k \phi_k(t) \tag{17}$$

where $\phi_k(t) = B_{k,r,t_N}(t)$ are normalized scalar B-spline basis functions defined on a uniform sequence of break points or knot sequence $t_N = \{k\Delta t\}_{k=-r+1}^{N+r-1}$ as

$$B_{k,r+1,t_N}(t) = \frac{t - t_{k-r-1}}{t_{k-1} - t_{k-r-1}} B_{k-1,r,t_N}(t) + \cdots \frac{t_k - t}{t_k - t_{k-r}} B_{k,r,t_N}(t), \qquad r \ge 1$$

$$B_{k,1,t_N}(t) = \begin{cases} 1 : t_{k-1} \le t \le t_k \\ 0 : \text{otherwise} \end{cases} \tag{18}$$

and N is the number of time intervals, $\alpha_k \in \mathcal{R}^4$ the parameter of optimization, and Δt the time step in discretised horizon.

As an example, for a horizon length of 0.3 s and discrete time step of 0.05 s we have $N = 6$ intervals. If we use third-order B-splines, that is, piecewise quadratic polynomials, then the number of parameters that determine the control trajectory would be $N + r - 1 = 6 + 3 - 1 = 8$ for each control, making a total of $4 \times 8 = 32$ parameters to optimize. The knot sequence for this case would be $t_N = \{0.05 \times k\}_{k=-2}^{8}$ with $N + 2r - 1 = 11$ break points. To understand visually how B-splines are constructed, consider a knot sequence $k = [0, 0.05, 0.1, 0.15, 0.2, 0.25]$ and $\alpha = [0.4001, 0.1988, 0.6252]$. The coefficient vector for this example is chosen at random. The resulting graph and basis functions are shown in Fig. 1. The resulting curve, shown in solid line, is the

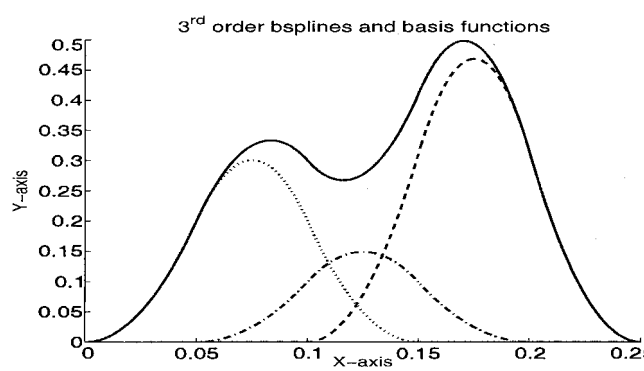

Fig. 1 Plot of B-spline functions used to construct a trajectory, basis functions (·, ·–, –), and resulting curve (——).

summation of the three nonsolid curves. The shape of the curve is determined by the knot sequence, the order of B-spline, and the coefficient multiplying each basis function. A third-order B-spline exhibits positional, first derivative, and second derivative continuity.

VIII. Simulation Setup

A. F-16 Model

The system interconnection for the F-16 aircraft is shown in Fig. 2. The control objective is to regulate the system from a perturbed state back to a trim condition using receding horizon control. The optimization to compute optimal open-loop control trajectory is carried out every 0.05 s, that is, the receding horizon controller runs at 20 Hz. The LPV controller runs every 0.01 s (100 Hz). A fourth-order Runge–Kutta integrator is used to update the system states. The step size for the integrator is 0.01 s, same as the LPV controller sampling rate. The F-16 model used for state updates and control trajectory synthesis is a full nonlinear model given in Eq. (2). The aircraft model is written in ANSI C, which includes routines needed to interpolate aerodynamic data from look-up tables. The airplane is trimmed at an altitude of 10,000 ft and angle of attack $\alpha = 7$ deg. Trimming is done by minimizing a quadratic cost function of the moments acting on the aircraft. Although the full nonlinear model has 12 aircraft states, we have only considered the state trajectories that are common with the LPV plant, that is, $(V_t, \alpha, \beta, p, q, r, \phi, \theta)$, the controls are the same as original, that is, $(T, \delta_e, \delta_a, \delta_r)$. The control signals generated by the LPV and RHC controllers are the synthetic inputs. The actuator and the trim values are for control vector $(\delta_a, \delta_r, u_1, u_2)$. They are converted to $(T, \delta_e, \delta_a, \delta_r)$ by the block labeled as ξ^{-1}. The actuators are modeled as first-order linear systems with the following properties:

$$|\delta_a| \le 20\,\text{deg}, \quad |\dot{\delta}_a| \le 80\ \text{deg/s}, \quad \tau_s = 0.0495\text{-s lag}$$

$$|\delta_r| \le 30\,\text{deg}, \quad |\dot{\delta}_r| \le 120\ \text{deg/s}, \quad \tau_s = 0.0495\text{-s lag}$$

$$|u_1| \le 63\,\text{deg}, \quad |\dot{u}_1| \le 137.51\ \text{deg/s}, \quad \tau_s = 0.0495\text{-s lag}$$

$$|u_2| \le 45.84\,\text{deg}, \quad |\dot{u}_2| \le 57.3\ \text{deg/s}, \quad \tau_s = 0.5\text{-s lag}$$

The magnitude and rate bounds of the actuators are implemented in the nonlinear simulation.

B. Numerical Details

Numerical analysis is a key part in receding horizon control simulations. Formulation of control space, integration algorithm used to update states and most importantly the optimization software all require numerical computation. Using higher-order algorithms provides better results, but it is also computationally expensive. Therefore one has to compromise between accuracy of solution and computational time. The on-line nature of the receding horizon control law has stringent real-time demands on the software used. We are still quite far from applying such control laws to aerospace problems in real life, based on the formulation in this paper. Nevertheless, an effort has been made to reduce the execution time of the nonlinear simulation software as much as possible. The following subsections briefly describe the tuning of the NPSOL software and finer numerical details that helped in reducing the computational time.

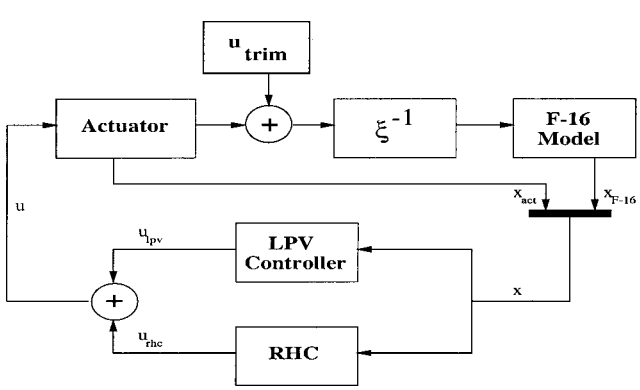

Fig. 2 System interconnection.

1. NPSOL

NPSOL is the heart of our simulation software. Details are explained in the NPSOL user manual. NPSOL is an optimization package written in FORTRAN. This was converted to ANSI C using f2c, which can be obtained from ftp://netlib.bell-labs.com/netlib/f2c/src.tar. The ANSI C version of the library is 30–50% slower than the original FORTRAN version. We used the ANSI C version because of the convenience in linking and compiling with other ANSI C modules of our simulation package.

The algorithm and software parameter selection for NPSOL is briefly described here:

1) NPSOL uses sequential quadratic programming (SQP) for nonlinear optimization. The SQP algorithm requires the gradient of the cost function with respect to the optimization parameters. The user has the option to supply these gradients as user-defined functions, or let NPSOL determine them numerically. We chose the latter for convenience at the expense of computational time.

2) The basic structure of NPSOL involves major and minor iterations. The minor iterations solves a quadratic subproblem that determines the search direction. Once the search direction has been computed, the major iteration proceeds to the next iterate, which produces a sufficient decrease in the augmented Lagrangian merit function. The sequence of iterates from the major iterations converge to a first-order Kuhn–Tucker point of the nonlinear optimization problem. Naturally, the limits on the maximum iteration number for both major and minor iterations affect the solution of the problem. A large limit will give more accurate results but will take a long time. A smaller limit will reduce execution time, but results could be far from optimal. For our simulations we set the limit on the major iterations to 10 and 5 for the minor iterations. These limits might not be suitable for all RHC simulations with various horizon lengths. One has to tune them to suit the problem. For example, for a horizon length of 0.4 s and major iteration limit set to 10, the RHC solution resulted in destabilizing the system. When the major iteration limit was increased to 20, the resulting trajectories were stable and gave cost-to-go consistent with the 0.3 and 0.5 horizon length optimizations. Hence, it is important to monitor the trajectories and tune parameters accordingly.

3) Optimization on bounded parameters is usually faster than for unbounded ones. The actuator model imposes rate and magnitude limits on the control action. Incorporating these constraints into the parameter optimization formulation improves the numerical conditioning of the problem. The bounds on the parameters of optimization $\alpha_k \in \mathcal{R}^4$ control the magnitude limits of the B-spline curves generated. We defined the bounds of the control rates to be the bounds of these parameters. To explain the reason behind this, consider the state-space formulation of a first-order actuator:

$$\dot{x}_{\text{act}} = (-x_{\text{act}} + u_{\text{in}})/\tau, \qquad u_{\text{out}} = x_{\text{act}}$$

where τ is the time constant, x_{act} is the actuator state, u_{in} is the signal at input, and u_{out} is the signal at output of the actuator. For our case u_{in} is defined as $u_{\text{in}} = u_{\text{rhc}} + u_{\text{lpv}}$ (see Fig. 2), the sum of signals from the receding horizon controller and the prestabilizing LPV controller. Thus,

$$\dot{x}_{\text{act}} = (-x_{\text{act}} + u_{\text{rhc}} + u_{\text{lpv}})/\tau$$

B-spline curves generated with α_k form the time trajectory of u_{rhc}, which influences the trajectory of $\dot{x}_{\text{act}}$. Because we are indirectly shaping the control rate trajectory with our manipulated variables $u_{\text{rhc}}(\alpha_k)$, bounds on the optimization parameters α_k are selected based on the limits associated with control rates $\dot{x}_{\text{act}}$. The stabilizing controller u_{lpv} acts in addition to u_{rhc}. Bounding u_{rhc} with limits of $\dot{x}_{\text{act}}$ does not guarantee boundedness of $\dot{x}_{\text{act}}$. The control rates and magnitude are kept within limits with the use of saturation blocks in the F-16 simulation model.

4) NPSOL needs an interval to be defined over which gradients can be estimated via finite difference. If a difference interval is not specified by the user, NPSOL computes it automatically with a

procedure that requires six calls to the cost function. Because this is computationally expensive, we defined this function discretization interval as 0.001 for our simulations. One has to be careful to select a small enough interval such that the gradients obtained from finite differences are accurate. The solution time with interval specified is 185.06 s, and without it is 334.9 s. Thus providing the interval reduces computation time by 44.74%.

5) Warm start was implemented from the second solution of optimal control. The time with warm start is 185.06 s, and time without is 175.41 s. Warm start actually increased the solution time by 5.5%.

6) Optimality tolerance was set to 0.01. The optimality tolerance determines the desired accuracy of the optimal solution. For example, with a tolerance of 10^{-6} the solution will have six correct figures.

2. Simulation Details

This section presents the numerical information required for implementation of receding horizon control law:

1) The optimization to generate the control trajectories was performed every 0.05 s. Because this optimization determines the optimal command from the current state, this is equivalent to saying that the feedback information was provided every 0.05 s.

2) Third-order B-splines were used in the control space formulation. This assumes the control trajectory behaves as quadratic polynomials in the time interval used to discretise the horizon length. A second-order B-spline would yield a piecewise linear control trajectory, and a first-order B-spline would form a piecewise constant control trajectory. Because we have four control variables to manipulate, reducing the order of B-spline r by one reduces the total number of parameters $4(N + r - 1)$ by 4, for a given number of intervals N in the horizon length. Reducing the number of parameters will reduce the computational time, but the set of allowable control trajectories is also reduced. The effect of B-spline order on cost-to-go is explained later in this paper.

3) A fourth-order Runge–Kutta algorithm is used to integrate the system with fixed step size of 0.01 s. The control for each time step in the numerical integration was computed from the B-spline curves that resulted from the optimization.

4) The entire simulation software, including the nonlinear F-16 model, aerodynamic data interpolation algorithms, basic linear algebra computation, is implemented in ANSI C.

IX. Results

This section presents the results of the receding horizon control optimization simulation, with the state-feedback LPV controller used both for prestabilizing and as the CLF in the RHC formulation. The performance of RHC and LPV controllers are compared in terms of the cost incurred to reach the origin from a given initial condition. Unfortunately, the computational time of the RHC algorithm is about 30 times slower than real-time requirements. A 5.0-s simulation took 149 and 9348 s CPU time for horizon lengths of 0.1 and 1.0 s, with third-order B-splines as the basis for control space formulation and 0.05 s horizon length discretization.

As mentioned earlier, the F-16 model was trimmed at an altitude of 10,000 ft and 7-deg angle of attack. The trim states and controls at this flight condition are

$$\begin{Bmatrix} \bar{V}_t \\ \bar{\theta} \\ \bar{\alpha} \\ \bar{u}_1 \\ \bar{u}_2 \end{Bmatrix} = \begin{Bmatrix} 387.5797 \text{ ft/s} \\ 7.000 \text{ deg} \\ 7.000 \text{ deg} \\ -7.437 \text{ deg} \\ 7.700 \text{ deg} \end{Bmatrix} \tag{19}$$

The remaining states and controls are zero at trim. The initial condition of the system was set by adding a 15- and 10-deg perturbation in angle of attack α and sideslip angle β, respectively. The control objective is to return the aircraft to its trim condition while respecting the limits on actuator magnitude and rates.

Results for 5.0 RHC simulation with horizon lengths 0.1 and 1.0 s are shown in Figs. 3–6. The plots are of the trajectories traversed by the system in course of the simulation. The control trajectories

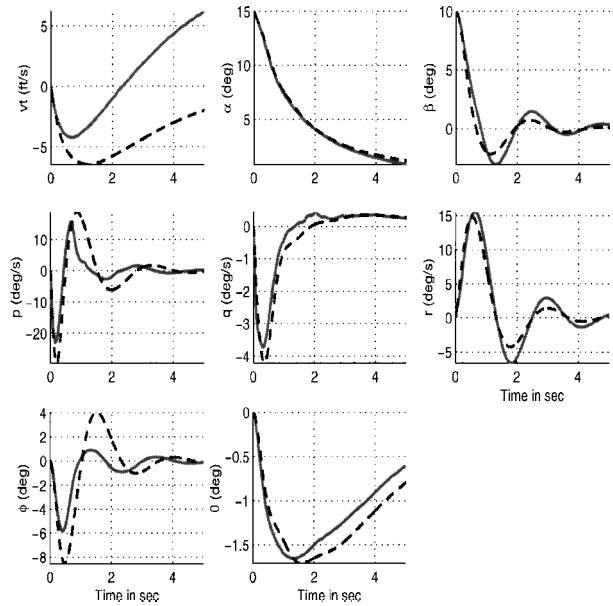

Fig. 3 State perturbation trajectories, horizon length = 0.1 s: ——, RHC and – – –, LPV.

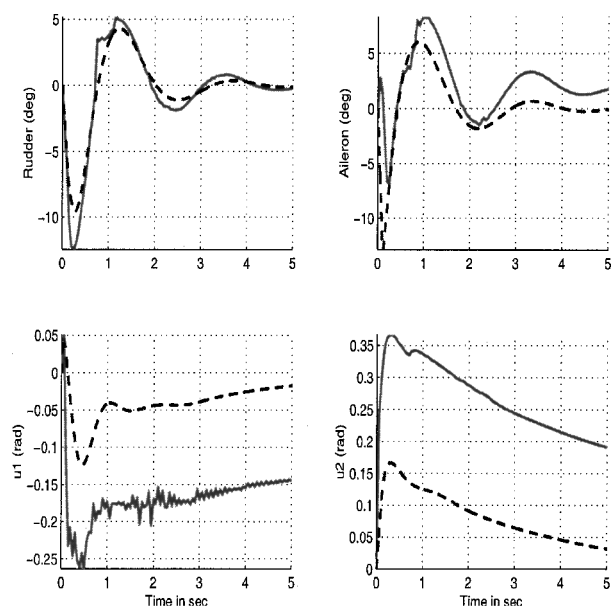

Fig. 4 Control perturbation trajectories, horizon length = 0.1 s: ——, RHC and – – –, LPV.

shown are the actuator outputs and not the controller outputs, and hence they satisfy magnitude and rate bounds. The cost, computed as

$$J = \int_0^{5.0} [x \quad u_{\text{LPV}}] P \begin{bmatrix} x \\ u_{\text{LPV}} \end{bmatrix} \mathrm{d}t \tag{20}$$

for LPV control is 128.47, with

$$P = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}_{16 \times 16}$$

$$P_{11} = \text{diag}\{6.25 \times 10^{-4}, 205, 3282, 156, 277, 100, 3282, 820, 217, 101, 72, 3\}_{12 \times 12}$$

$$P_{12} = \begin{bmatrix} 0_{8 \times 4} \\ \text{diag}\{-14.5, -9.6, -8.42, -0.5\} \end{bmatrix}_{12 \times 4}$$

$$P_{21} = [0_{4 \times 8} \text{ diag}\{-14.5, -9.6, -8.42, -0.5\}]_{4 \times 12}$$

$$P_{22} = \text{diag}\{217, 101, 72, 3\}_{4 \times 4}$$

Table 1 Cost variation with horizon length

Horizon length, s	Cost	CPU time, s
0.1	89.92	149
0.2	82.54	412
0.4	77.98	1989
1.0	77.23	9348

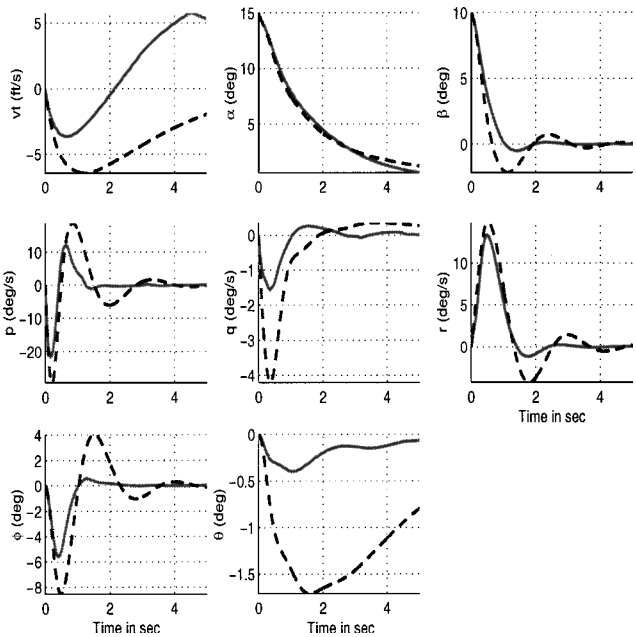

Fig. 5 State perturbation trajectories, horizon length = 1.0 s: ——, RHC and – – –, LPV.

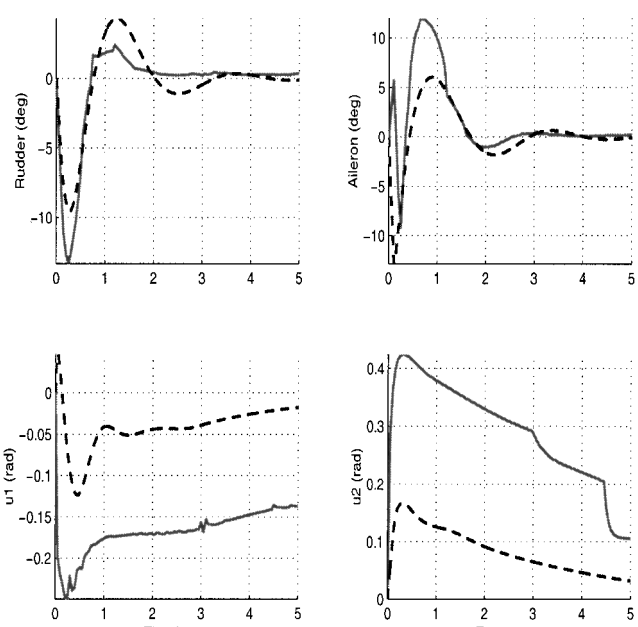

Fig. 6 Control perturbation trajectories, horizon length = 1.0 s: ——, RHC and – – –, LPV.

Nonlinear simulations reveal that the RHC cost is significantly smaller than the LPV cost. This is reflected in the fast convergence of the perturbations to origin. The variation of cost with horizon length and B-spline order (r) is given in Table 1.

The horizon length plays an important role in the performance of the receding horizon controller. It can be inferred from Figs. 3–6 that the state and control trajectories become less oscillatory as horizon length is increased. The effect of horizon length on the trajectories can be explained by drawing analogies between proportional integral differential (PID) controllers and receding horizon controllers. For horizon lengths that are small compared to the speed of response of the system, the system response to a given control input within that time span appears to be sluggish. Thus the receding horizon controller detects a slower system for short horizon lengths. It tries to minimize the objective function in this small time interval by applying large control inputs. A large control input is feasible because its contribution to the overall cost is small when integrated over a short time interval. In the linear system framework this is analogous to increasing the rise time by raising the controller gain. But increasing the controller gain increases the band width and reduces the phase margin of the closed-loop system, which might cause instability in the system. This is not possible because the RHC algorithm with a suitable terminal penalty is also stabilizing. The receding horizon controller restores stability by introducing a differential control action in the control trajectory. This increases the phase margin. However, a large differential control action is not feasible because its influence on the rise time is not significant. Thus, it cannot be large enough to fully compensate for the effects of a high loop gain. Therefore the reduced phase margin causes oscillations in the system trajectories.

In our simulations the RHC optimization with horizon length of 0.05 s led to control trajectories that caused limit-cycle-like behaviour in the system. This phenomenon can be explained based on the speed of response of the system. The first-order actuators in the F-16 aircraft model have time constant of 0.049 s. Therefore a horizon length of 0.05 was too short for the RHC algorithm to observe the effects of a given control input. The sequence of consistent large control inputs led the system into instability. For longer horizon lengths the system is able to fully respond to a given control input. The RHC algorithm is able to observe both transient and posttransient effects of the system. This rich information about the system's response yields control trajectories that are less aggressive. The resulting system trajectories are smoother and more optimal. The variation of the cost-to-go with horizon length is shown in Fig. 7.

The overall cost-to-go is lower for longer horizon lengths because the RHC algorithm has richer knowledge of the system behavior to a given input, and so better is the decision on optimality for every control trajectory generated. This also translates to the fact that optimization over longer time horizon reduces the contribution of the terminal cost on the total cost. Because the terminal cost serves as an approximation that is greater than the value of the truncated integral, reducing its contribution yields a total cost that is closer to the optimal. It is also observed that increasing the horizon length beyond a certain time does not improve the performance of the controller in terms of the cost-to-go. This is because for a long enough time horizon the system is steered close to the origin within that time, and the value of the truncated integral is almost zero. The horizon length has reached infinity relative to the system response time.

From the simulation we conclude that combination of LPV and receding horizon controller provides a better performance than the LPV controller alone. The difference is significant even for small horizon lengths and low B-spline orders. The combination of the two controllers yields control trajectories that are able to aggressively maneuver the state perturbations to the origin and at the same time satisfy limitations on the control rate and magnitude.

The choice of B-spline order is governed by the order of the integrator used. For fourth-order Runge–Kutta integrators we can only

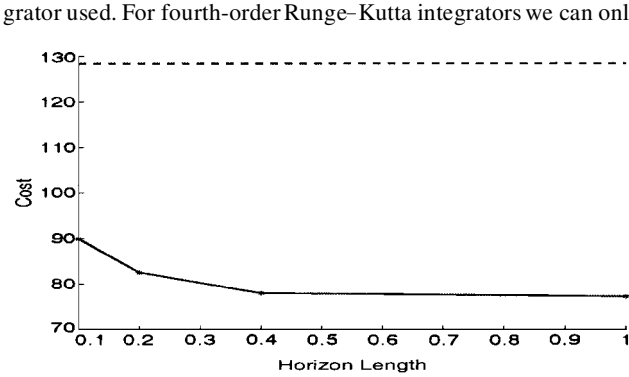

Fig. 7 Cost vs horizon length (B-spline order = 3): – – –, LPV cost and ——, RHC cost.

use B-splines of order 1, 2, and 3 (Ref. 24). Because NPSOL computes first- and second-order gradients of cost with respect to control numerically, the formulated control subspace should at least have second-order continuity. This limits us to only third-order B-splines. Using lower-order B-splines will reduce the number of parameters, and that will clearly reduce the computational time. However, its not clear what will be its effect on the overall system behavior. Owing to this numerical issue, the effect of B-spline order has not been investigated in this paper.

The RHC simulations were done on a 400 Mhz Pentium II machine running Debian Linux. The entire simulation software, including the F-16 system model, interpolation routines for aerodynamic data, basic linear algebra computation, and numerical integration, is written in ANSI C. The optimization package NPSOL came as FORTRAN routines, which were converted to C using f2c.

X. Conclusions

The combination of on-line receding horizon control and off-line linear, parameter-varying regulator design, led to improved performance on an F-16 flight control example. The LPV regulator is used to prestabilize the unstable F-16 aircraft and as a control Lyapunov function endpoint penalty in the RHC optimization. The numerics of the nonlinear RHC optimization is a significant issue. The convergence of the RHC optimizations and the overall computational time are some issues that need to be examined in order to achieve a reliable on-line control algorithm.

Acknowledgments

This work was funded by the Defense Advanced Research Projects Agency under the Software Enabled Control program with Helen Gill as the Program Manager. The contract number is USAF/AFMC F33615-99-C-1497, and Dale Van Cleave is the Technical Contract Monitor. The authors would like to thank Ali Jadbabaie and Mark Milam for their help with the receding horizon problem insight.

References

[1]Qin, S., and Badgwell, T., "An Overview of Industrial Model Predictive Control Technology," *AIChE Symposium Series*, Vol. 93, American Inst. of Chemical Engineers, New York, 1996, pp. 232–256.

[2]Bemporad, A., and Morari, M., "Robust Model Predictive Control: A Survey," *Robustness in Identification and Control*, edited by A. Garulli, A. Tesi, and A. Vicino, Lecture Notes in Control and Information Sciences, Springer-Verlag, Berlin, 1999, pp. 207–226.

[3]Kwon, W., "Advances in Predictive Control: Theory and Application," 1st Asian Control Conf., 1994.

[4]Bitmead, R., Gevers, M., and Wertz, V., *Adaptive Optimal Control: The Thinking Man's GPC*, International Series in Systems and Control Engineering, Prentice–Hall, Upper Saddle River, NJ, 1990.

[5]Soeterboek, R., *Predictive Control: A Unified Approach*, International Series in Systems and Control Engineering, Prentice–Hall, Upper Saddle River, NJ, 1992.

[6]Rodellar, J., and Martín Sánchez, J., *Adaptive Predictive Control*, International Series in Systems and Control Engineering, Prentice–Hall, Upper Saddle River, NJ, 1996.

[7]Shamma, J., and Cloutier, J., "Gain-Scheduled Missile Autopilot Design Using Linear Parameter Varying Transformation," *Journal of Guidance, Control, and Dynamics*, Vol. 16, No. 2, 1993, pp. 256–263.

[8]Primbs, J., "Nonlinear Optimal Control: A Receding Horizon Approach," Ph.D. Dissertation, Control and Dynamical Systems Dept., California Inst. of Technology, Pasadena, CA, Jan. 1999.

[9]Jadbabaie, A., Yu, J., Primbs, J., and Huang, Y., "Comparison of Nonlinear Control Designs for a Ducted Fan Model," International Federation of Automatic Control World Congress, IFAC-2c-112, Beijing, July 1999.

[10]Milam, M., and Murray, R., "A Testbed for Nonlinear Systems: The Caltech Ducted Fan," *IEEE International Conference on Control Applications*, Vol. 1, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 1999, pp. 345–351.

[11]Bryson, A. E., Jr., and Ho, Y.-C., *Applied Optimal Control*, Hemisphere, Washington, DC, 1975, p. 131.

[12]Keerthi, S., and Gilbert, E., "Optimal Infinite-Horizon Feedback Laws for a General Class of Constrained Discrete-time Systems," *Journal Optimization Theory and Application*, Vol. 57, 1988, pp. 265–293.

[13]Mayne, D., and Michalska, H., "Receding Horizon Control of Nonlinear Systems," *IEEE Transactions on Automatic Control*, Vol. 35, No. 7, 1990, pp. 814–824.

[14]Mayne, D., and Michalska, H., "Robust Receding Horizon Control of Constrained Nonlinear Systems," *IEEE Transactions on Automatic Control*, Vol. 38, No. 11, 1990, pp. 1623–1633.

[15]Parisini, T., and Zoppoli, R., "A Receding-Horizon Regulator for Nonlinear Systems and a Neural Approximation," *Automatica*, Vol. 31, No. 10, 1995, pp. 1443–1451.

[16]De Nicolao, G., Magni, L., and Scattolini, R., "Stabilizing Receding-Horizon Control of Nonlinear Time-Varying Systems," *IEEE Transactions on Automatic Control*, Vol. 43, No. 7, 1998, pp. 1030–1036.

[17]Jadbabaie, A., Yu, J., and Hauser, J., "Stabilizing Receding Horizon Control of Nonlinear Systems: A Control Lyapunov Function Approach," *American Control Conference*, Vol. 3, IEEE Publ., Piscataway, NJ, 1999, pp. 1535–1539.

[18]Gill, P., Murray, W., Saunders, M., and Wright, M., NPSOL-Nonlinear Programming Software, Stanford Business Software, Inc., Mountain View, CA.

[19]Stevens, B., and Lewis, F., *Aircraft Control and Simulation*, Wiley-Interscience, New York, 1992.

[20]Nguyen, L., Ogburn, M., Gilbert, W., Kibler, K., Brown, P., and Deal, P., "Simulator Study of Stall/Post-Stall Characteristics of a Fighter Airplane with Relaxed Longitudinal Static Stability," NASA TR 1538, Dec. 1979.

[21]Shin, J. Y., "Worst-Case Analysis and Linear Parameter-Varying Gain-Scheduled Control of Aerospace Systems," Ph.D. Dissertation, Dept. of Aerospace Engineering and Mechanics, Univ. of Minnesota, Minneapolis, MN, Oct. 2000.

[22]Hull, D., "Conversion of Optimal Control Problems into Parameter Optimization Problems," *Journal of Guidance, Control, and Dynamics*, Vol. 20, No. 1, 1997, pp. 57–60.

[23]de Boor, C., *A Practical Guide to Splines*, Springer-Verlag, New York, 1978.

[24]Schwartz, A., "Theory and Implementation of Numerical Methods Based on Runge–Kutta Integration for Solving Optimal Control Problems," Ph.D. Dissertation, Electronics Research Lab., Univ. of California, Berkeley, CA, April 1996.