Toward Design Parameterization Support for Model Predictive Control

Jonathan Sprinkle, J. Mikael Eklund and S. Shankar Sastry
Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA 94720-1774
Email: {sprinkle,eklund,sastry}@EECS.Berkeley.Edu

***Abstract*— Research into the autonomous behavior of Unmanned Aerial Vehicles (UAVs) requires concise and dependable specification techniques in order to provide behavioral descriptions for the controllers of these aircraft. Practical issues with autonomous aircraft involve the safety and reliability of the controller (e.g., guarantee of stability), as well as verification of the high-level intention of the autonomous behavior. Since most behaviors are implemented orthogonally to their high-level specification, improvements in the ability to rapidly specify the models that govern the behavior are certainly welcome. This paper describes the framework for decreasing the abstraction required to specify the behavior of an autonomous controller implemented through model predictive control.**

I. Introduction

The increased usage of Unmanned Aerial Vehicles (UAVs) is practically guaranteed for the foreseeable future. Recently, UAVs are most famously deployed for military or surveillance purposes, due to their reduced risk of loss-of-life, decreased visibility by adversaries (when compared to manned vehicles) as well as drastically reduced cost of production.

For the most part, however, UAVs have literally been an extended version of manned flight, since notable UAVs such as the Predator are (for the the most part) deployed, flown, and retrieved by a remote pilot, who is supported by a large staff that monitors the aircraft and its surroundings. This implementation is required, because the safety and reliability of the UAV demands human intervention, due to the real-time constraints of its behavior.

The use of models in the specification of behaviors of UAVs tends to be more important when specifying the mathematical definition of the dynamics of the aircraft, rather than the actual autonomous behavior of the aircraft. The state of the art for behavior specification is through the use of (Nonlinear) Model Predictive Control (MPC, or NMPC), which provides a guide for desired behavior under certain environmental stimuli. The complexity of the models of the system dynamics, in combination with the highly *ad hoc* nature of the behavioral specifications, renders this problem as (1) computationally intensive and (2) prone to iterative designs that require trial and error testing.

This paper describes a proposed framework for model-based specification of the autonomous behavior of systems that may be governed by NMPC techniques. Section II gives background information about the definition and behavior of NMPC. Section III discusses the physical constraints of an avionics system (UAV), and justifies the requirement that these constraints be satisfied in real-time. Section IV describes the limitations and advantages of using models to meet the constraints of the system, as well as possible ways in which the desired behavior of the system can be mapped directly into the format required by an NMPC controller. Finally, conclusions and ideas for future work are given.

II. NMPC

Nonlinear Model Predictive Control (NMPC) is a method by which behaviors may be encouraged in system controllers. NMPC is performed through the global or local optimization of some cost function that describes the encoded behavior of the system. By using predictions of the future behavior of the cost function, a controller utilizing NMPC can give an autonomous set of behaviors to a system.

A. Cost Functions

In [1] the use of NMPC allowed for a fixed-wing aircraft to participate in evasive maneuvers when confronted by an adversary with similar behavioral characteristics. The cost function, J, is generically stated as,

$$J = \phi(\mathbf{b}_{1_N..M_N}) + \sum_{k=0}^{N-1} L(\mathbf{x}, \mathbf{u}, \mathbf{b}_{1..M}), \tag{1}$$

where,

$$\phi(\mathbf{b}_{1_N..M_N}) = C \sum_{m=1}^{m=M} \mathbf{b}_m^{\mathrm{T}} \mathbf{B}_{0_m} \mathbf{b}_m \tag{2}$$

and,

$$L(\mathbf{x}_k, \mathbf{u}_k, \mathbf{b}_{k_{1..M}}) \triangleq C \left(\mathbf{x}_k^{\mathrm{T}} \mathbf{X}_0 \mathbf{x}_k + \mathbf{u}_k^{\mathrm{T}} \mathbf{U}_0 \mathbf{u}_k + \sum_{m=1}^{m=M} \mathbf{b}_{m_k}^{\mathrm{T}} \mathbf{B}_{0_m} \mathbf{b}_{m_k} \right) \tag{3}$$

where N represents the number of timesteps the NMPC controller looks ahead for its predictive calculations, $\mathbf{b}$ is the set of M different encoded axes of behaviors for the NMPC controller, $\mathbf{x}$ is the state of the system, and $\mathbf{u}$ is the system input interface. The function $L(\cdot)$ is used to calculate the behavior of the system according to the relative cost of certain behaviors.

B. Optimization

The gradient descent algorithm described in [2], [3] can be used to find a locally optimal solution to the equation $J = 0$. The input interface $\mathbf{u}$, being part of this equation, can be directly applied to the controller, yielding the behavior that was *predicted* by the NMPC controller (assuming that the environment continues as was predicted as well).

At least one implementation of the gradient descent algorithm (in use in [1], [2]) can be employed to run as a real-time task for fixed-wing aircraft control. The optimization procedures, when given a simple-enough characteristic model of the behavior of the aircraft, can execute rapidly enough to allow a 1–3 Hz refresh rate. While this is not sufficient for an inner-loop controller, it is an acceptable rate for an outer-loop controller, e.g., control through an autopilot interface that guarantees basic stability of the aircraft.

III. Avionics Constraints

As detailed in [1], NMPC can be used to give behaviors to aircraft based on the *a priori* design requirements of the system modeler. Included in these behavioral models are not only the dynamical equations that govern the Eulerian motion of the aircraft, but also the hard constraints that describe the physical (or empirical) limits of the aircraft. Examples of these kinds of avionics constraints are the maximum/minimum velocity, climb/descent rate, turn rate, attitude, wind speed, etc.

These avionics constraints find there way into the $\mathbf{u}$ vector mapping, which constrains the inputs possible to the system. In an autopilot situation, the inner-loop calculations and feedback definitions are not used, but high-level inputs (such as the turn rate command) are given. The avionics constraints for physical limits of the aircraft in [1] are given as,

$$
\begin{aligned}
map(u_{\dot{v}}) &= \begin{cases} -50[\text{f/s}] & -\infty < u_{\dot{v}} < -1 \\ [-50, 50]\,[\text{f/s}] & -1 \leqslant u_{\dot{v}} \leqslant 1 \\ 50[\text{f/s}] & 1 < u_{\dot{v}} < \infty \end{cases} \\
map(u_{\dot{\psi}}) &= \begin{cases} -\pi/50[\text{s}^{-1}] & -\infty < u_{\dot{\psi}} < -1 \\ [-\pi/50, \pi/50]\,[\text{s}^{-1}] & -1 \leqslant u_{\dot{\psi}} \leqslant 1 \\ \pi/50[\text{s}^{-1}] & 1 < u_{\dot{\psi}} < \infty \end{cases} , \\
map(u_{\dot{z}}) &= \begin{cases} -10[\text{ft/s}] & -\infty < u_{\dot{z}} < -1 \\ [-10, 10]\,[\text{ft/s}] & -1 \leqslant u_{\dot{z}} \leqslant 1 \\ 10[\text{ft/s}] & 1 < u_{\dot{z}} < \infty \end{cases}
\end{aligned}
\tag{4}
$$

with,

$$
\mathbf{u} = \left[u_{\dot{v}}, u_{\dot{\psi}}, u_{\dot{z}}\right] \in [-1, 1]^3 \in \mathbb{R}^{n_u} \tag{5}
$$

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.

Other constraints, beyond the scope of this paper, restrict the latitude/longitude of flight, may restrict certain headings while in fly-zones, or possibly even disallow input to the system in a no-fly-zone (e.g., if a secondary autonomous unit, such as the Soft Walls [4], [5] implementation, takes control of the aircraft in an emergency situation).

The importance of these constraints to an NMPC solution is that they are required to guarantee the high-level stability of the system. That is, the NMPC controller is designed to account for a predetermined refresh in the system state—if there is an overflow or backup in the real-time behavior, then the system will slowly (or, perhaps rapidly) become unstable.

IV. Use of Models

In order to ensure the high-level stability of the aircraft, then, we must restrict (or approximate) the models that describe the behavior of the aircraft (as they are represented in the NMPC model) such that

- the behavior is simple enough to allow real-time execution of N steps of evaluation in the future, and
- the behavior is accurate enough to reflect the actual behavior (with some acceptable difference).

For example, the precision of the flight dynamics need not be hundredths of an inch, but should be more accurate than 100 miles. When designing NMPC controllers, the determination of dynamics is second only to the definition of the $\mathbf{X}_0, \mathbf{U}_0$, and $\mathbf{B}_{0_{1..M}}$ matrices in terms of difficulty.

A. Determining Matrix Values

The matrices $\mathbf{X}_0, \mathbf{U}_0$, and $\mathbf{B}_{0_{1..M}}$ are used to obtain scalar values for individual portions of the cost function (note that $\mathbf{x}_k^{\mathrm{T}}\mathbf{X}_0\mathbf{x}_k$ is a scalar, and that the dimensions of $\mathbf{X}_0$ are $n_{\mathbf{x}} \times n_{\mathbf{x}}$). The basic scalar portions, we will call them s_x and s_u, are present in each NMPC implementation, regardless of whether it is an aerial vehicle, ground vehicle, or some other controlled system.

These matrices are usually diagonal, although occasionally the dynamics of the system require non-zero values for non-diagonal elements of the matrix in order to maintain certain behaviors. The difference in value of each element in the matrix reflects the relative importance of that value to each individual scalar portion. For instance, the relative importance of the $u_{\dot{\psi}}$ portion of $\mathbf{u}$ can be obtained by dwarfing the $u_{\dot{v}}$ and $u_{\dot{\phi}}$ values in the $\mathbf{U}_0$ matrix. Usually, however, these values are different because of unit differences, (for example, the values in the cost function should reflect whether the amount is given in radians vs. degrees).

Tuning the individual matrix values of the $\mathbf{X}_0$ and $\mathbf{U}_0$ generally yield an NMPC definition that controls the system to maintain its current state; this only part of the overall NMPC definition, however. The $\mathbf{B}_{0_{1..M}}$ matrices usually outnumber those of the basic aircraft, sometimes substantially, and it is these matrices that give the unique behavioral characteristics of the system under certain stimuli. That is, the $\mathbf{B}_{0_{1..M}}$ matrices override the steady-state behavior that would ensue if only $\mathbf{X}_0$ and $\mathbf{U}_0$ were present in the cost function.

The determination of these individual axis values is by far the most complex portion of the NMPC development process. Generally, an *ad hoc* method is used, which involves closed-loop experimentation with a systems expert. Combined with extended testing and verification of intent, this allows a

system designer to implement desired behaviors under certain conditions.

B. Determining Models of Dynamics

The dynamics of the system are important for the look-ahead portion of the NMPC controller using gradient descent methods. Mathematical definitions (rather than data lookup-table models obtained by system identifaction methods) are required, since the gradient descent algorithm requires functional based lookup of the partial derivatives at certain points with respect to the input vector and the state vector.

Certainly, given the mathematical versions of the dynamics, it would be possible to do a symbolic manipulation of the dynamics to obtain the proper partial derivatives. The dynamics models, however, are not trivial to determine, and require the input of a skilled domain expert who has deep knowledge of the system.

V. Model Generation

We have discovered for our aerial application that using basic Eulerian rules of motion we are able to sufficiently approximate the actual behavior of the aircraft under flight conditions[1]. This was an arduous process however, and now that we have settled on using the Eulerian equations, we realize that we have more computational time left to look further "ahead" in time, that is, increase our value of N.

It is this kind of support that should enable another layer of models for NMPC *code generation*. The gradient descent algorithm is domain independent (meaning that it exists in mathematical form, and orthogonal to any particular application that is encoded into the $\mathbf{x}, \mathbf{u}$, and $\mathbf{b}_{1..M}$ axes). In fact, it is this encoding that is the "hard" part of the system specification—yet, the problem of creating a rapid implementation that is suitable to run in real time and on the appropriate operating system is not a trivial one. Furthermore, much of the work required to provide this NMPC implemenation is *exactly the same* from system to system, differing only by,

1) the system dynamics, and partial derivatives;
2) the input and state vectors;
3) the lookahead count; and
4) the axes vectors $\mathbf{b}_{1..M}$ used to determine behavior.

The question, then, is how best to provide the above inputs to a system expert, while generating optimal (or at least an acceptable local maximal) code that manages the "implemenation details" of an NMPC controller.

Generating the NMPC controller in a fast language such as C++ is an interesting exercise in software engineering. Providing template classes and code generators that "fill in the blanks" of the above design parameters is certainly a useful task, but is not the most interesting science involved in this research.

Where we see a need (and indeed, some possibility) for support is in the *high-level* specification of desired behaviors that generates the *low-level* axes vectors $\mathbf{b}_{1..M}$, along with the initial population their coefficient matrices $\mathbf{B}_{0_{1..M}}$. It may be possible to capture certain known behaviors through case study or recording, and determine how they should be encoded. For example, pursuit/evasion of fixed-wing aircraft has a large following among war aviators, and today's pilots study tactics and maneuvers to escape from adversaries in certain scenarios. Texts such as [6] describe these maneuvers in detail; however, it is not intuitively obvious how to encode them into an NMPC controller, much less guarantee that such a behavior will occur under those conditions, and not be blocked by another behavior that is "cheaper" according to the cost function.

Such a possibility indicates the possible use of hybrid models to switch between optimal control laws in different scenarios. This is an intersting use of hybrid models that will be under our future study. Methods such as those described in [7] have been used to explore the reachable states of a hybrid (or other) model, and can perhaps be used to check to ensure that only the desired behaviors issue in the appropriate scenarios.

VI. Current and Future Support for Generation

The current state of the generative support is in theory only. There do not exist any tools or frameworks for the true parameterization of NMPC-based controllers. However, our empirical software construction suggests that a large majority of the existing codebase is parameterizable—with emphasis on the configuration of the lookahead length and the equations governing the dynamics of the aircraft.

Support for model generation should be initially related to the actual dynamics of the system and the final "requirements" of the system. Once these two portions of the cost function are formally defined, then the gradient-descent method can be applied to their formal specification. Once support for this has been established, additional axes of the requirements (e.g., constraints around certain execution paths or state values) can be introduced through additional specification. Possibly the most difficult portion of the generative support is the extraction of the system dynamics from experimental runs or desired outputs. This is similar to controller synthesis (given a desired path of the controller, and providing an input that gives that path), so it will be possible to get inspiration from system identification and existing controller synthesis engines to provide these mappings.

VII. Conclusion and Outlook

We have described the use of models in the definition of a nonlinear model predictive controller, and how the decision to choose the precision of those models can affect the ability of the controller to run as a real-time task. We also discussed how the cost functions that give the stability and behavior of a controller using NMPC can be difficulty to encode, and obfuscate the actual behavior of the system when that behavior is encoded.

Our position is that the behavior that is encoded into an NMPC controller can to a large degree be automated,

[1]This guarantee is given through a real-time simulator provided by Boeing, not through actual flight tests. However, the Boeing simulator is guaranteed by that company to accurately reflect the true behavior of the aircraft.

reducing the time to develop the controller. Furthermore, the specification of the encoded behavior, as the most difficult portion of the controller implementation, is in need of a higher-level form of specification, and we hope to explore how best to capture this information.

ACKNOWLEDGMENT

Many thanks to Jin Kim and Ian Mitchell, who have worked hard to implement and use model checking software, as well as provide assistance in the understanding of NMPC and reachability calculations.

This work was supported in part by the DARPA Software Enabled Control (SEC) project.

REFERENCES

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

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

[3] ——, "Decentralized nonlinear model predictive control of multiple flying robots in dynamic environments," in *IEEE Conference on Decision and Control*. IEEE Press, Dec. 2003.

[4] A. Cataldo, "Control algorithms for soft walls," Master's thesis, University of California, Berkeley, Berkeley, CA 94720, Jan. 2004, technical Memorandum UCB/ERL M03/42.

[5] E. A. Lee, "Soft walls - modifying flight control systems to limit the flight space of commercial aircraft," University of California, Berkeley, Berkeley, CA, 94720, Tech. Rep. Revised from UCB/ERL Memorandum M001/31, Oct. 2001.

[6] R. L. Shaw, *Fighter Combat: Tactics and Maneuvering*. United States Naval Inst., Sept. 1988.

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