Berkeley
University of California

Toward Design Parameterization Support for Model Predictive Control

Jonathan Sprinkle, J. Mikael Eklund, S. Shankar Sastry

University of California, Berkeley
Department of Electrical Engineering & Computer Sciences

Overview

- What is MPC?
- How does it work?
- Example : aircraft control
- Motivation for parameterization
- Room for parameterization
- Planned work
- Conclusions

Model Predictive Control

- MPC is a method for restricting/encouraging behavior
- A "fortune teller" controller
- Restricts input ranges, as well as encourages some inputs based on safety/stability concerns
- Very useful for *nonlinear* systems, due to the ability to get good optimizations with non-linear abstractions

How does it work?

- Basic algorithm:
  - Examine the mathematical abstraction of the system (PDE)
  - Determine value along N time steps into the future
  - Optimize this value, according to some *a priori* specifications (to $J = 0$)

$$
\begin{aligned}
J &= \phi(\mathbf{b}_{1_N..M_N}) + \sum_{k=0}^{N-1} L(\mathbf{x}, \mathbf{u}, \mathbf{b}_{1..M}) = 0 \\
\phi(\mathbf{b}_{1_N..M_N}) &= C \sum_{m=1}^{m=M} \mathbf{b}_m^{\mathrm{T}} \mathbf{B}_{0_m} \mathbf{b}_m \\
L(\mathbf{x}_k, \mathbf{u}_k, \mathbf{b}_{k_{1..M}}) &\triangleq C \left(\mathbf{x}_k^{\mathrm{T}} \mathbf{X}_0 \mathbf{x}_k + \mathbf{u}_k^{\mathrm{T}} \mathbf{U}_0 \mathbf{u}_k + \sum_{m=1}^{m=M} \mathbf{b}_{m_k}^{\mathrm{T}} \mathbf{B}_{0_m} \mathbf{b}_{m_k} \right)
\end{aligned}
$$

System Model

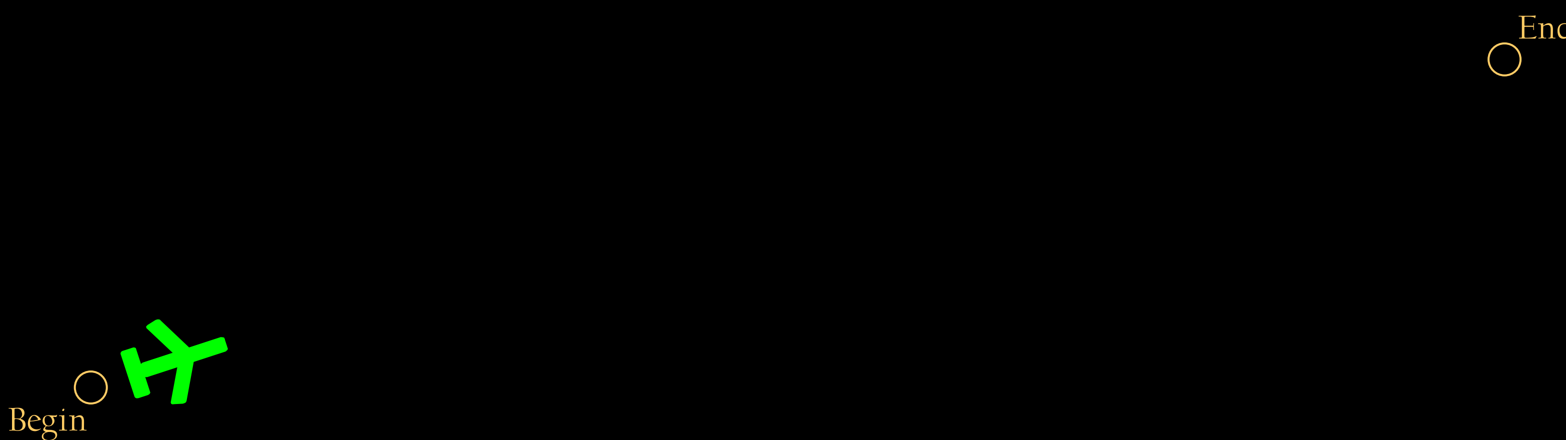
- In the form of,

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$$

- Obviously, very system-dependent
- Sometimes an abstraction of the *actual* system in order to speed up computation
- Accuracy of the prediction, directly tied to the abstraction
- Eventually, arrive at a snapshot N steps in the future

$$[\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N}]$$

Example: Aircraft Control



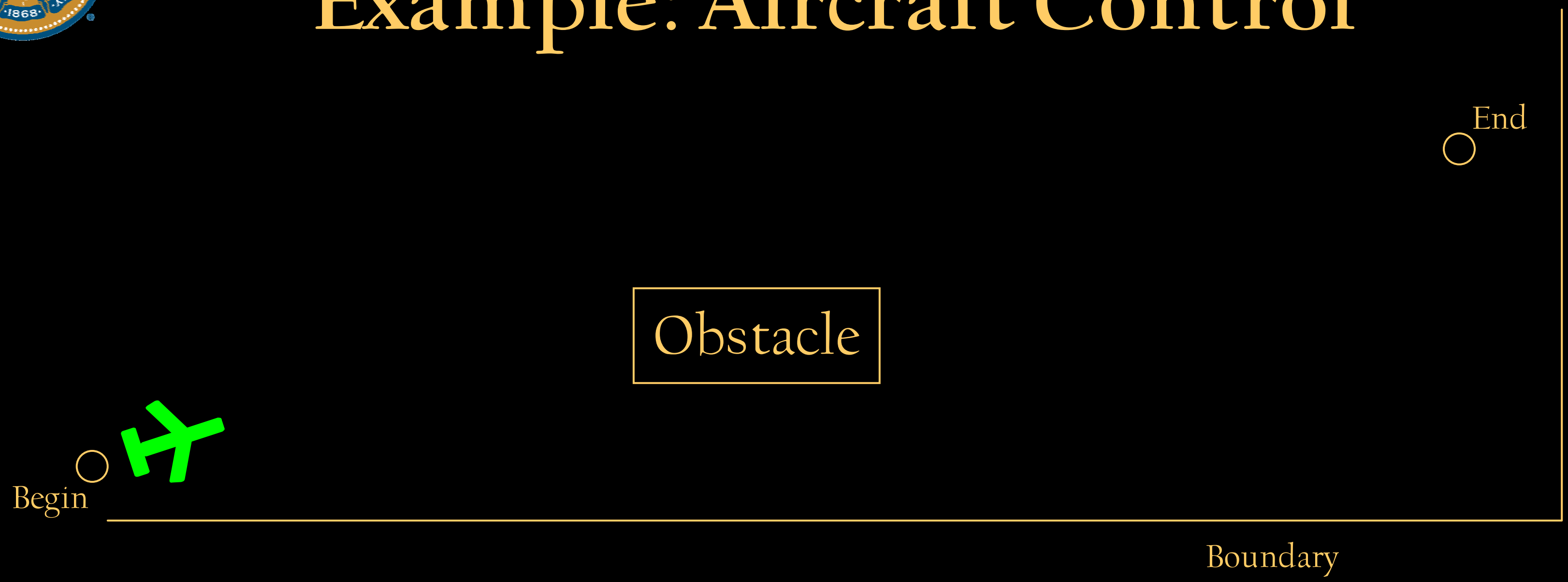
$$L(\cdot) \triangleq \mathbf{x}_k^{\mathrm{T}} \mathbf{X}_0 \mathbf{x}_k + \mathbf{u}_k^{\mathrm{T}} \mathbf{U}_0 \mathbf{u}_k \quad + \quad \mathbf{b}_{m_1}^{\mathrm{T}} \mathbf{B}_{0_1} \mathbf{b}_{m_1}$$

Berkeley
University of California

Example: Aircraft Control

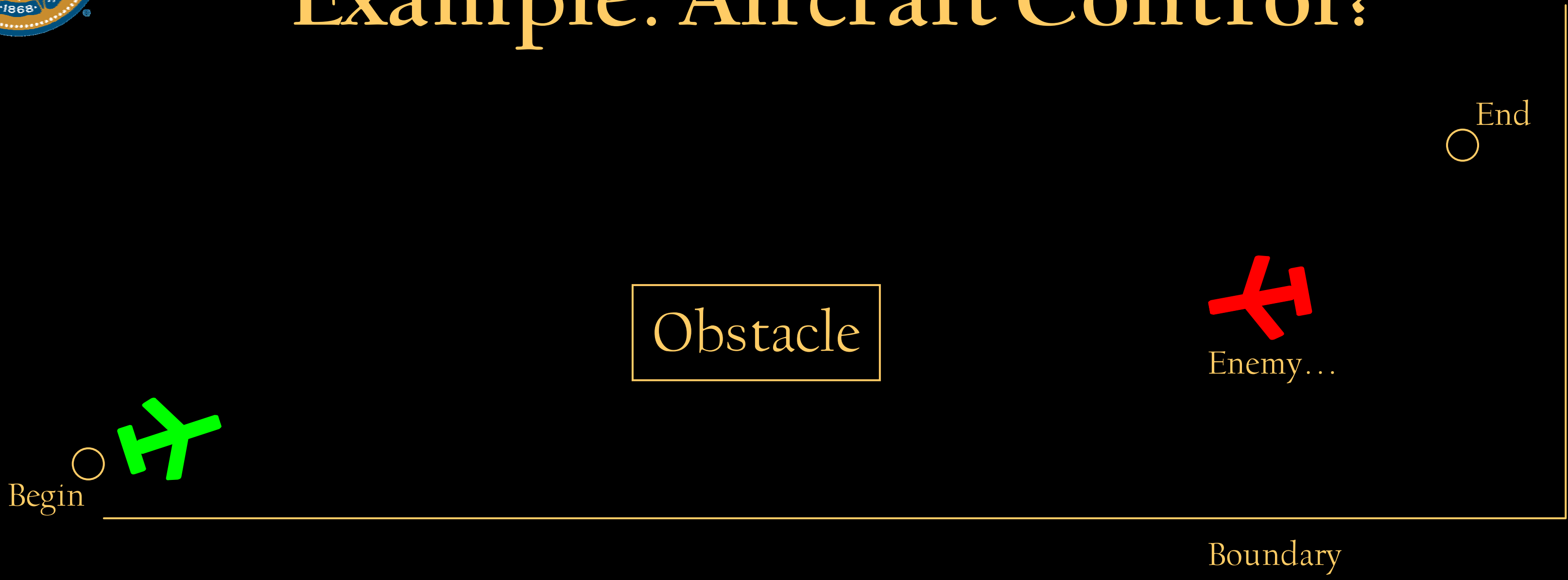
$$L(\cdot) \triangleq \mathbf{x}_k^{\mathrm{T}}\mathbf{X}_0\mathbf{x}_k + \mathbf{u}_k^{\mathrm{T}}\mathbf{U}_0\mathbf{u}_k \quad + \quad \mathbf{b}_{m_1}^{\mathrm{T}}\mathbf{B}_{0_1}\mathbf{b}_{m_1}$$
$$\quad + \quad \mathbf{b}_{m_2}^{\mathrm{T}}\mathbf{B}_{0_2}\mathbf{b}_{m_2}$$

Example: Aircraft Control



$$
\begin{aligned}
L(\cdot) \triangleq \mathbf{x}_k^{\mathrm{T}}\mathbf{X}_0\mathbf{x}_k + \mathbf{u}_k^{\mathrm{T}}\mathbf{U}_0\mathbf{u}_k \quad & + \quad \mathbf{b}_{m_1}^{\mathrm{T}}\mathbf{B}_{0_1}\mathbf{b}_{m_1} \\
& + \quad \mathbf{b}_{m_2}^{\mathrm{T}}\mathbf{B}_{0_2}\mathbf{b}_{m_2} \\
& + \quad \mathbf{b}_{m_3}^{\mathrm{T}}\mathbf{B}_{0_3}\mathbf{b}_{m_3}
\end{aligned}
$$

Example: Aircraft Control?



$$
\begin{aligned}
L(\cdot) \triangleq \mathbf{x}_k^{\mathrm{T}}\mathbf{X}_0\mathbf{x}_k + \mathbf{u}_k^{\mathrm{T}}\mathbf{U}_0\mathbf{u}_k \quad & + \quad \mathbf{b}_{m_1}^{\mathrm{T}}\mathbf{B}_{0_1}\mathbf{b}_{m_1} \\
& + \quad \mathbf{b}_{m_2}^{\mathrm{T}}\mathbf{B}_{0_2}\mathbf{b}_{m_2} \\
& + \quad \mathbf{b}_{m_3}^{\mathrm{T}}\mathbf{B}_{0_3}\mathbf{b}_{m_3} \\
& + \quad \mathbf{b}_{m_?}^{\mathrm{T}}\mathbf{B}_{0_?}\mathbf{b}_{m_?}
\end{aligned}
$$

Example: Aircraft Control

- Now, what do you do?
  - Hope that you don't get caught?
  - First, fight with you left hand, and then surprise you opponent by not being left-handed
  - Encode "getting away" from your opponent into the cost-function

"I admit it, you are better than I am"
"Then why are you smiling?"
*"Because *I* am not left-handed"*

Pursuit/Evasion: $\mathrm{D}_{\mathrm{D}}\mathrm{e}_{\mathrm{e}}\mathrm{t}_{\mathrm{v}}\mathrm{a}_{\mathrm{i}}\mathrm{i}_{\mathrm{l}}\mathrm{ls}$

Berkeley
University of California

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

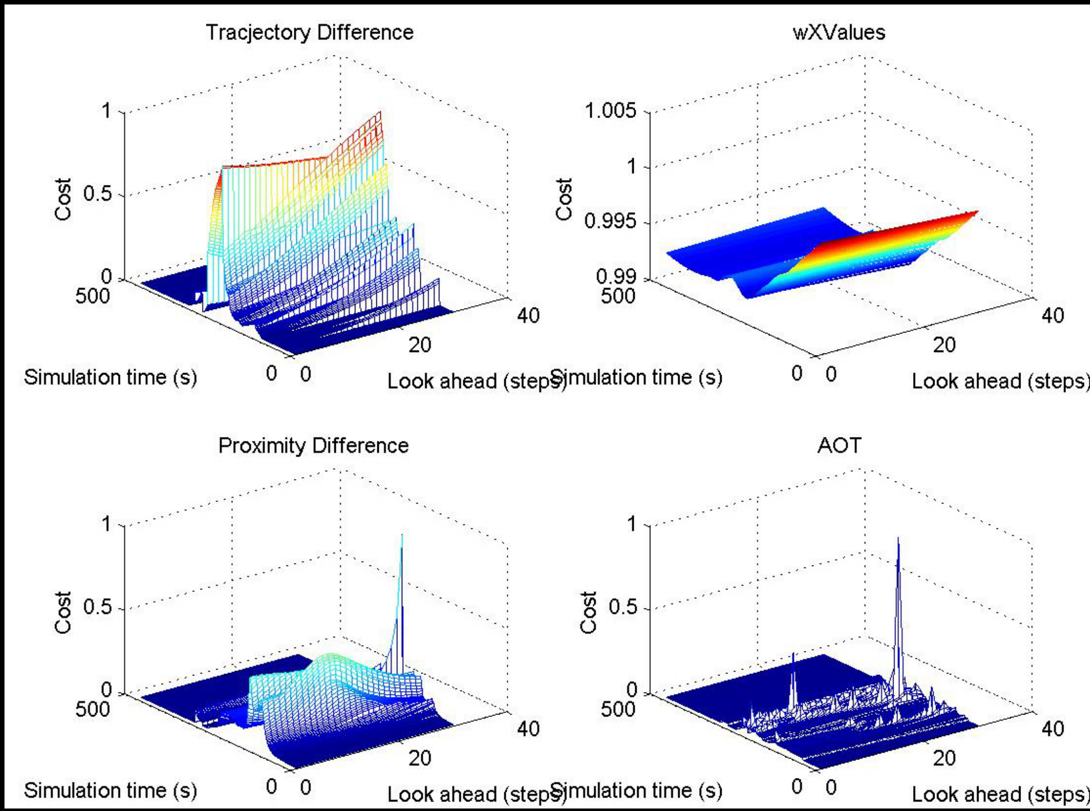
where

$$\phi(\tilde{y}_N) \triangleq \frac{1}{2}\left(\tilde{y}_N^T P_0 \tilde{y}_N\right)$$

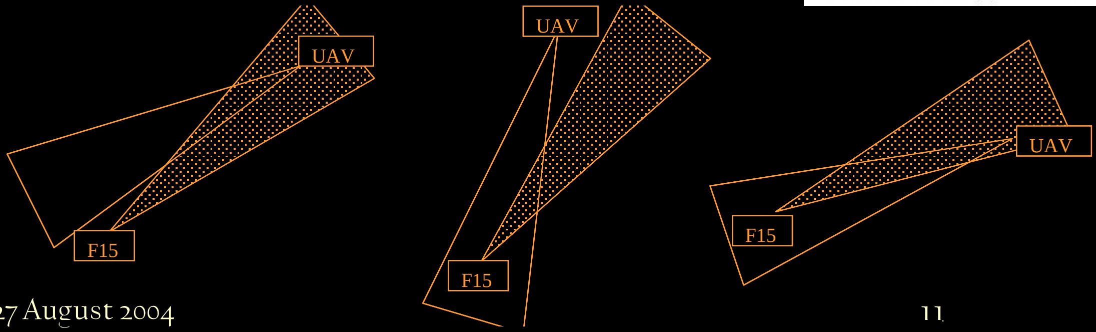
and

$$L(x, \tilde{y}, u, d, a) \triangleq \frac{1}{2}\left(\tilde{y}_k^T Q \tilde{y}_k + x_k^T S x_k + u_k^T R u_k + \frac{1}{\left(d_k^T G d_k\right)^{\frac{1}{2}}} + \frac{1}{\left(a_k^T T a_k\right)^{\frac{1}{4}}}\right)$$

- x is the state vector
- u is the control vector
- $\tilde{y}$ is the trajectory error
- d is the pursuer/evader position difference
- a is the angle off tail or other tactical functions



Targeting information:



The Real Problem:

- Making it work is nice, *but*
  - How in the devil did we come up with those
    - Equations
    - Individual components
    - Matrix values
  - Is there a way to derive these from the application constraints?
- Additionally
  - How hard was it to write a fast optimizer?
  - Is there a way to make this interface easily usable?

Toward a solution:

System-dependent, Behavior-dependent, Independent

System Model

CostFunction

- Input Constraints
- Safety Constraints
- Application Constraints
  - **x** ranges
  - **u** ranges
  - Behaviors

Optimizer

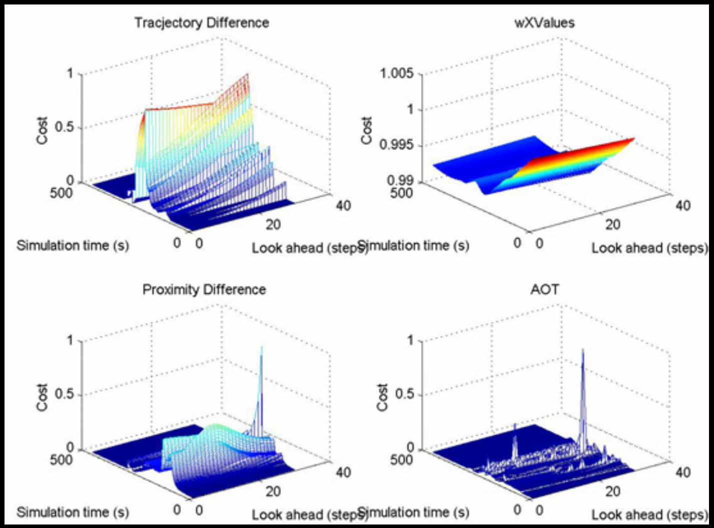
$(J = 0)$

Toward a solution

- System-dependent
  - Can be derived for a particular system's mathematical definition
  - In general, quite easy to obtain
- Independent
  - Software engineering exercise
  - Once defined, will be reused
- Behavior-dependent
  - By far the hardest piece of the solution
  - Not generally derivable, but there are tricks that should be available for all future implementers, that a parameterized approach can provide

Behavior-dependent tricks

- Use the itemized pieces of the cost function to examine overall volatility under certain criteria
- Steer inputs to provide an "order of magnitude" cost function behavior
- Provide a mechanism to translate math definitions into computer code



Future plans

- Currently implementing a new NMPC problem using different models and designs
- Will be developing the NMPC interface to provide the behavior for this new application, using
  - Ideas presented here
  - Suggestions received here
- Evaluate the new MATLAB MPC toolbox, to see what benefits it offers

Conclusions

- MPC can be used to provide interesting behaviors for linear and non-linear control systems, but not necessarily a fast development cycle
- We hope to reduce the development cycle by at least
  - Providing a cost-function independent optimizer
  - Inventing an intuitive interface to generate the cost function
  - Developing a method/tool to tune the cost function for desired behaviors
  - Experimenting with ways to reverse engineer values for the matrices, based on desired behaviors under stimuli

Questions

"Well HAL, I'm damned if I can find anything wrong with it."
"Yes. It's puzzling. I don't think I've ever seen anything quite like this before."
-- *2001: A Space Odyssey*