CME 334 Project:
Model Predictive Control
An Introduction and Application to a Quadrotor

Zouhair Mahboubi (11/2012)
Outline

- Introduction
  - Motivation
  - Literature Review

- Quadrotor
  - Equations of Motion
  - Problem Formulation

- Results
  - Cat & mouse investigation approach
  - Comparison with PID and discrete LQR
Introduction

- Model Predictive Control
  - Subset of Optimal Control Theory
  - Relies on model of dynamical system to control a plant
  - Formulated as an optimization problem
    - Objective is a function of the states and controls
    - Constraints can be on both states and controls
    - Design variables are the control inputs
Motivation

- Achieves optimal objective
- Can deal with complex dynamical systems
  - Including non-linear dynamics
- Can account for actuator saturation
- Can provide ‘envelope’ protection through state constraints
Drawbacks

- Requires the solution of an optimization problem
  - Feasibility? Convergence? Speed?
  - Problem size can grow quickly
    - Except for slow systems, or simplified models; online solution is difficult

- Difficult to establish stability guarantees & margins
  - Lack of ‘classical’ damping, gain-phase margin metrics
  - Robustness to model uncertainty?
  - Disturbance rejection?
Literature Review

- Predictive Control for linear and hybrid systems; F. Borrelli, A. Bemporad, M. Morari; 2012
- MPC: Review of Three Decades; J.H. Lee; 2011
- Robust MPC: A survey; Bemporad et Al.; 1999
- Constrained Optimal Attitude Control of a Quadrotor; K.Alexis et Al.; 2010
- State and Output Feedback Nonlinear MPC; R. Findeise et Al.; 2003
MPC is a success story in process industry
- Chemical plants have complex but slow dynamics
  - i.e. more time to solve the optimization problem

But generally, NMPC has a large computational complexity
- We will therefore focus on “Linear Quadratic Optimal Control”
  - Convex quadratic objective function
  - Linear dynamics and Convex linear constraints

Fast MPC
- Existence of offline solutions to optimization problem
- Customized algorithms to exploit MPC structure
Literature Review: Receding Horizon Control
Constrained Linear Quadratic Optimal Control (CLOQC)

- Discrete LTI dynamics

- Quadratic Objective Function

\[ J(x(0), U_0) = \sum_{k=0}^{N-1} (x_k^t Q x_k + u_k^t R u_k) + x_N^t P x_N \]

- Convex Optimization problem:

\[
\begin{align*}
\text{minimize} & \quad J(x(0), U_0) \\
\text{subject to} & \quad x(k + 1) = Ax(k) + Bu(k) \quad k = 1, \ldots, N - 1 \\
& \quad x_0 = x(0) \\
& \quad x(N) \in \chi_f \\
& \quad GX \leq h \\
& \quad FU < e
\end{align*}
\]
CLOCQ: Observations

- Without constraints:
  - Collapses to discrete LQR
  - Lots of existing work on stability, robustness, design methodology (to pick Q & R)

- The final cost and set $P, \chi_f$ are used to impose stability requirements
- Borelli et al. use set-theory to obtain conditions for feasibility and stability

- Previous formulation is a regulator
  - Can be easily expanded to the tracker problem
  - i.e. drive a desired output $y$ to $y_{ref}$ instead of $x$ to 0
CLOCQ: Solution

- Offline solution to constrained problem:
  - The optimization problem is a multi-parametric QP
  - Solution is piece-wise affine in polyhedra regions of $x$
    - Compare to linear in $x$ for discrete LQR
  - Unfortunately this has some limitations:
    - Polyhedra regions are exponential with number of constraints and horizon length
    - Only works for the regulator problem

- Online solution possible but depends on:
  - Required control rate and available computational power
  - Planning and re-planning horizons
  - Number of states, control inputs, constraints
Quadrotor

- Lots of interest by research community

- Applications as UAVs:
  - Power lines, oil rigs or wind turbine inspection
  - Border patrol, perimeter search, surveillance
Quadrotor & MPC

Three options:
- monolithic MPC to find thrust commands to get to a desired position. Fast dynamics and requires long horizon)
- MPC for high-level trajectory planning and leave inner loop control to classical controller. Slow dynamics, but requires longer horizon.
- MPC for inner-loop control and classical controller for outer loop. Fast dynamics, but requires shorter horizon.

Interested in the last one:
- The complex dynamics that need to be handled are those of the inner loop (actuator saturation, control allocation, etc.)
- Non-linearities in outer loops are easier to handle if the inner loop is well behaved
Equations of Motion

- Focusing on 2 DOF, pitch and vertical axis
- The goal is to achieve inertial lateral and vertical forces $F_x, F_z$ (commanded by a position/velocity controller)
- EOM are:

$$\dot{x} = \begin{bmatrix} \dot{\theta} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ q \end{bmatrix} + \frac{1}{I_{yy}} \begin{bmatrix} 0 & 0 & 0 & 0 \\ d_1 & d_2 & d_3 & d_4 \end{bmatrix} \begin{bmatrix} t_1 \\ \ldots \\ t_4 \end{bmatrix}$$

$$\begin{bmatrix} F_{x,i} \\ F_{z,i} \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ -\sin \theta & -\cos \theta \end{bmatrix} \begin{bmatrix} 0 \\ \Sigma t_i \end{bmatrix}$$
‘Classical’ Controller

- We use as a benchmark a classical PID controller:
  - Successive loop PID controller:
    - Given Fx command, synthesize pitch command
    - Run pitch controller, synthesize My command
    - Solve control allocation problem with (Fz, My) to find required controls
      - Note that this can be posed as an optimization problem as well, but is relatively ‘trivial’
Cat & Mouse game

- Start with a mouse that’s easy to catch
- Teach the cat how to catch it
- Make the mouse harder to catch
- Teach the cat a new trick
- Repeat until the mouse is the desired problem and the cat is the solution 😊
Game #1

- MPC has perfect knowledge of the model
  - Should be able to solve the optimization problem once
  - And then apply the solution in a feed-forward manner

- The solution should compare to the discrete LQR in the absence of constraints

- This was mostly to verify that the problem formulation and solver worked as expected

- A lot of the work was to setup the framework...
Game #1
Game #1
Game #2

- Add a disturbance $M_y$ to the system:
  - Feed-forward diverges

- How does one handle disturbances in MPC framework?
  - Literature review turned out two options:
    - Augment state with an integrator (not ideal...)
    - Synthesize a disturbance estimator and include disturbance in dynamics

\[ x_{k+1} = Ax_k + Bu_k + B_d \hat{M}_{yd} \]
Game #2
Game #2
Game #3

Include a stabilizing final set constraint

\[ x_N = Ax_N + Bu_{N-1} + B_d \dot{M}_{yd} \]

Include more dynamics:

- So far we ignored actuator dynamics. These are pretty important and performance is very bad if not accounted for; but to take them into account we introduce a lot of states:

\[ x_k = [\theta, q, t1, \dot{t}1, \ldots, t4, \dot{t}4]^t_k \]

- The increase in the number of states slows things down...
- Fortunately, reformulating the problem allows for an amazing simplification

\[
\text{minimize } \quad J(U) = \frac{1}{2} U^t HU + c^t U \\
\text{subject to } \quad GU = h \\
FU \leq e
\]
Game #4-5

Now that we can handle more states:

- Include even more dynamics (velocity-pitch interaction)
- Add additional axes (roll-Fy and yaw-Mz)
  - We expect to still have the same problem to solve, which is promising!
Game #4-5

- Do it as fast as possible
  - Look into ‘inexact’ and fast solvers tailored for MPC
    - Work by Y.Wang and S.Boyd might be relevant
  - Investigate the trade-off between performance and computational delay w.r.t. changes in control rate, planning and re-planning horizon lengths, number of constraints, move-blocking, etc.

- Can this be done online?
  - If not with a CPU, what about FPGA or other parallel architectures?
  - If not, how much speed-up are we missing?
Game #4-5
Game #4-5
Current Status

- Fast MPC implementation from Wang/Boyd does not handle tracker case:
  - Expanded implementation of the code.
    - Takes about 10ms on PC
  - Formulation in Wang/Boyd requires a full-rank hessian
    - Discuss implications...

- Literature indicates that people are using FPGAs to speed-up MPC problems;
  - Seems like the computation power is there;
  - Tomlin’s group at Berkeley has flown MPC online at 40Hz control rate
Objective: \( J(x(0),U_0) = \sum_{k=0}^{N-1} (x_k^tQx_k + u_k^tRu_k) + x_N^tPx_N \)

Optimization: \[ \begin{aligned} & \text{minimize} & & J(x(0),U_0) \\ & \text{subject to} & & x(k+1) = Ax(k) + Bu(k) \quad k = 1, \ldots, N-1 \\ & & & x_0 = x(0) \\ & & & x(N) \in \chi_{f} \\ & & & G X \leq h \\ & & & F U \leq e \end{aligned} \]

EOM: \[
\dot{x} = \begin{bmatrix} \dot{\theta} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ q \end{bmatrix} + \frac{1}{I_{yy}} \begin{bmatrix} 0 & 0 & 0 & 0 \\ d_1 & d_2 & d_3 & d_4 \end{bmatrix} \begin{bmatrix} t_1 \\ \ldots \\ t_4 \end{bmatrix}
\]

\[
\begin{bmatrix} F_{x,i} \\ F_{z,i} \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ -\sin \theta & -\cos \theta \end{bmatrix} \begin{bmatrix} 0 \\ \Sigma t_i \end{bmatrix}
\]

Final Set constraint: \( x_N = Ax_N + Bu_{N-1} + B_d \hat{M}_{yd} \)

Disturbance: \( x_{k+1} = Ax_{k} + Bu_{k} + B_d \hat{M}_{yd} \)