Model Predictive Control

An Introduction and Application to a Quadrotor

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?

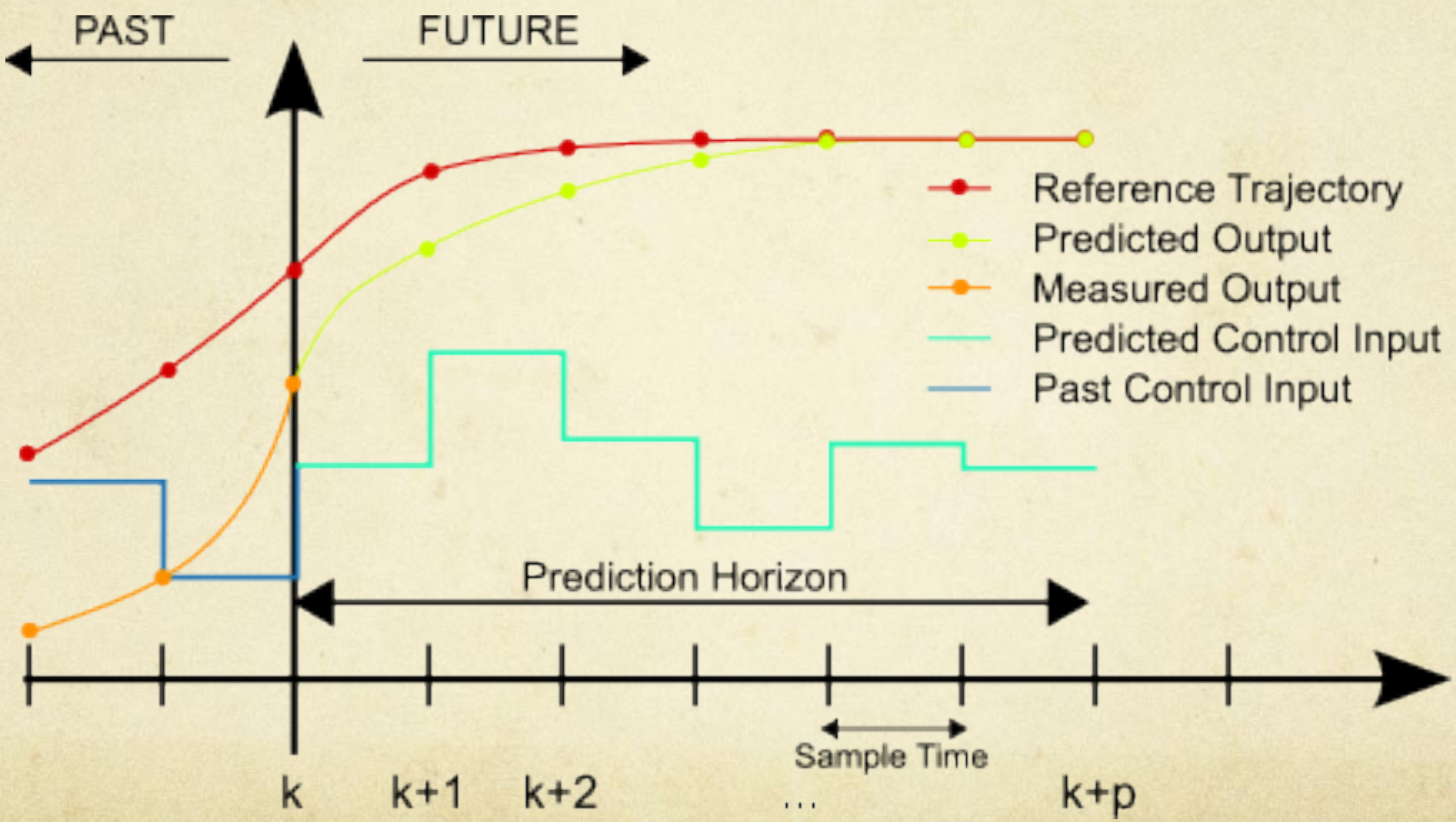
Literature Review

- 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

Literature Review

- 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:

$$\begin{aligned}
\underset{U_0}{\text{minimize}} \quad & J(x(0),U_0) \\
\text{subject to} \quad & x(k+1) = Ax(k) + Bu(k) \;\; k = 1,\ldots,N-1 \\
& x_0 = x(0) \\
& x(N) \in \chi_f \\
& GX \leq h \\
& FU \leq e
\end{aligned}$$

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 yref 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:
  - monolothic 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 Fx, Fz (commanded by a position/velocity controller)
- EOM are:

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

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

‘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’

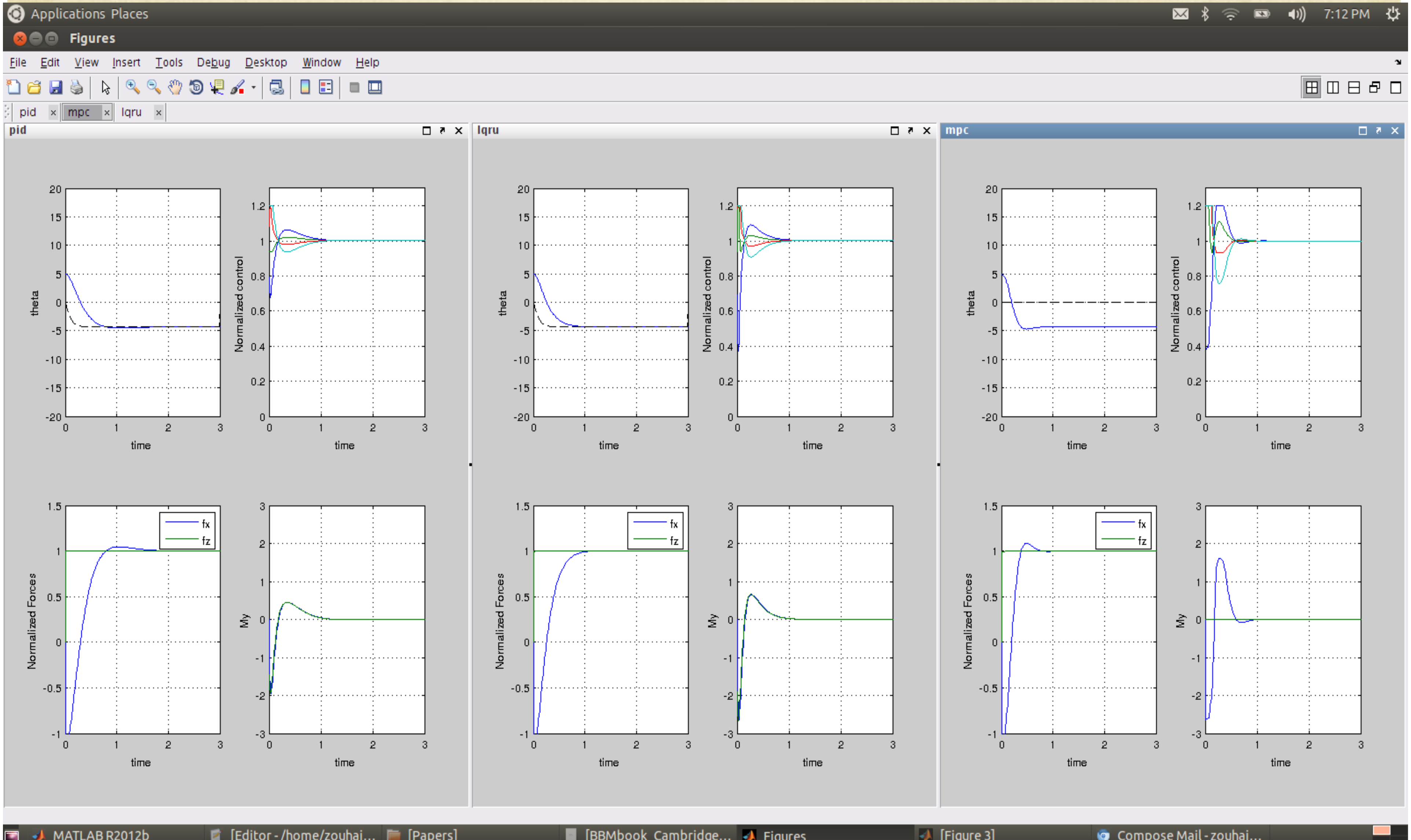
Cat & Mouse game

- 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 ☺

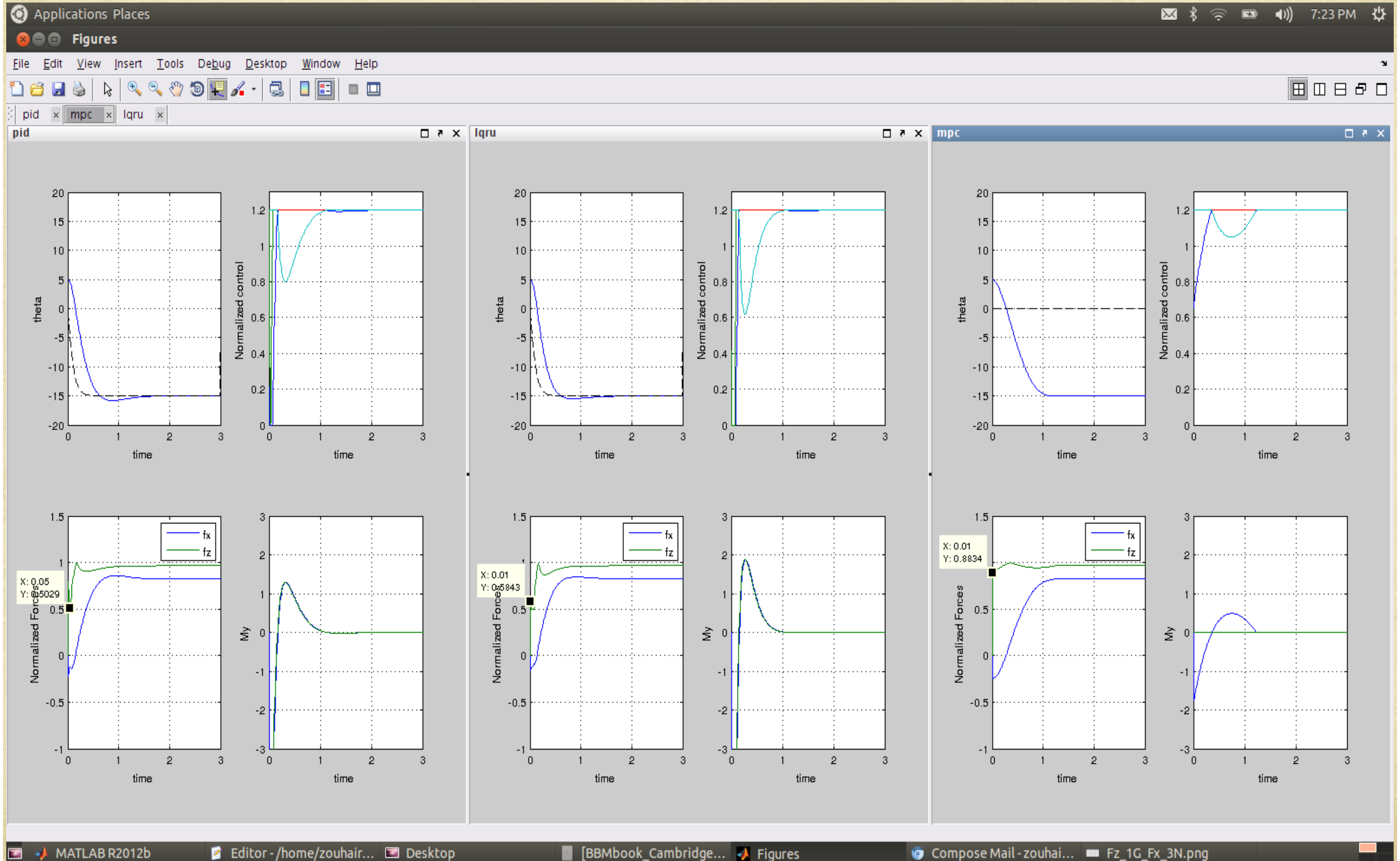
Game #1

- 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


Applications Places
7:12 PM
Figures
File Edit View Insert Tools Debug Desktop Window Help
pid mpc lqru
pid
lqru
mpc
theta
Normalized control
Normalized Forces
My
time
fx
fz
MATLAB R2012b
[Editor - /home/zouhai...
[Papers]
[BBMbook_Cambridge...
Figures
[Figure 3]
Compose Mail - zouhai...

Game #1



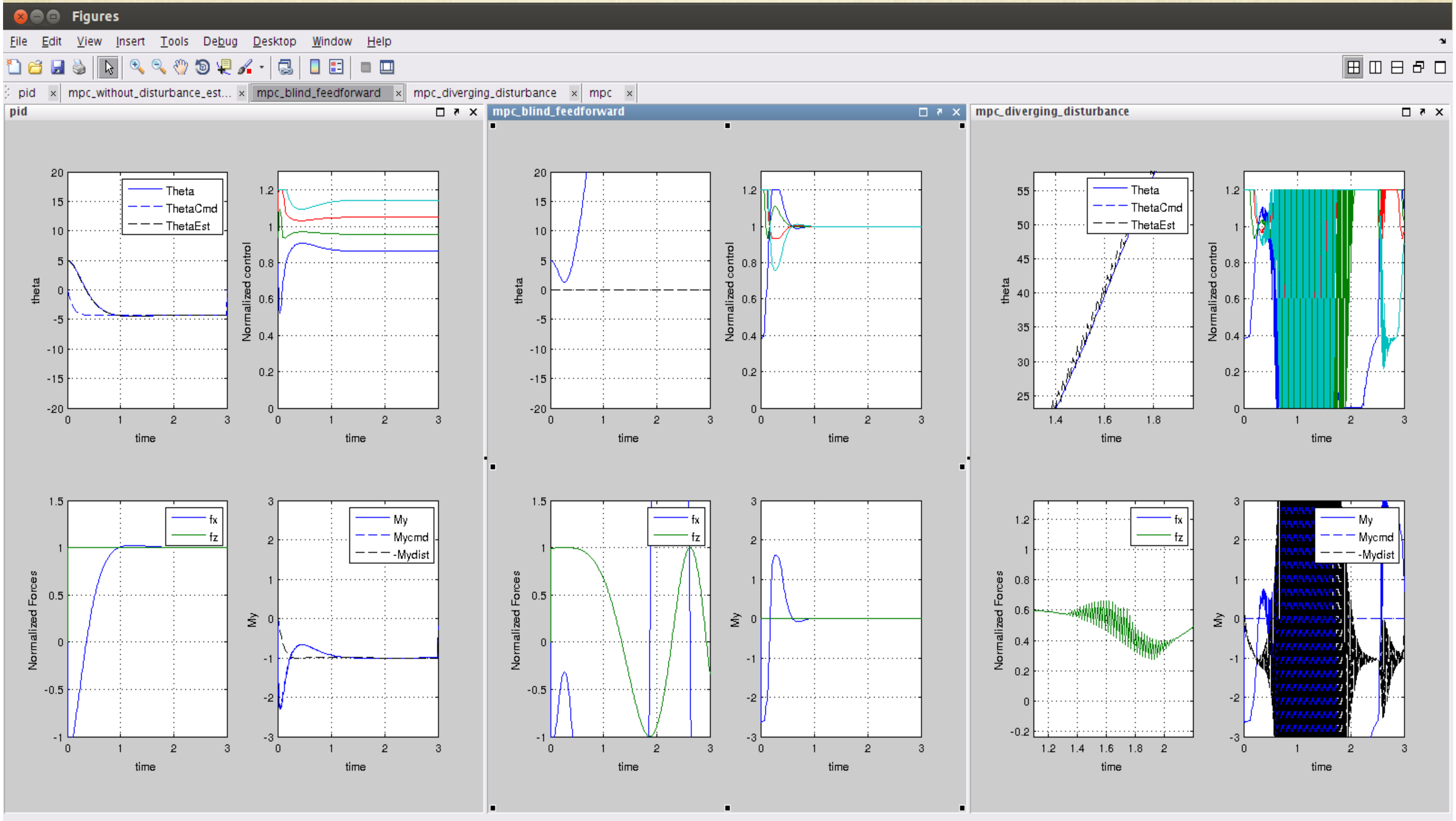
Applications Places
7:23 PM
Figures
File Edit View Insert Tools Debug Desktop Window Help
pid mpc lqru
pid
lqru
mpc
theta
time
Normalized control
Normalized Forces
My
fx
fz
X: 0.05
Y: 0.5029
X: 0.01
Y: 0.5843
X: 0.01
Y: 0.8834
MATLAB R2012b
Editor - /home/zouhair...
Desktop
[BBMbook_Cambridge...
Figures
Compose Mail - zouhai...
Fz_1G_Fx_3N.png

Game #2

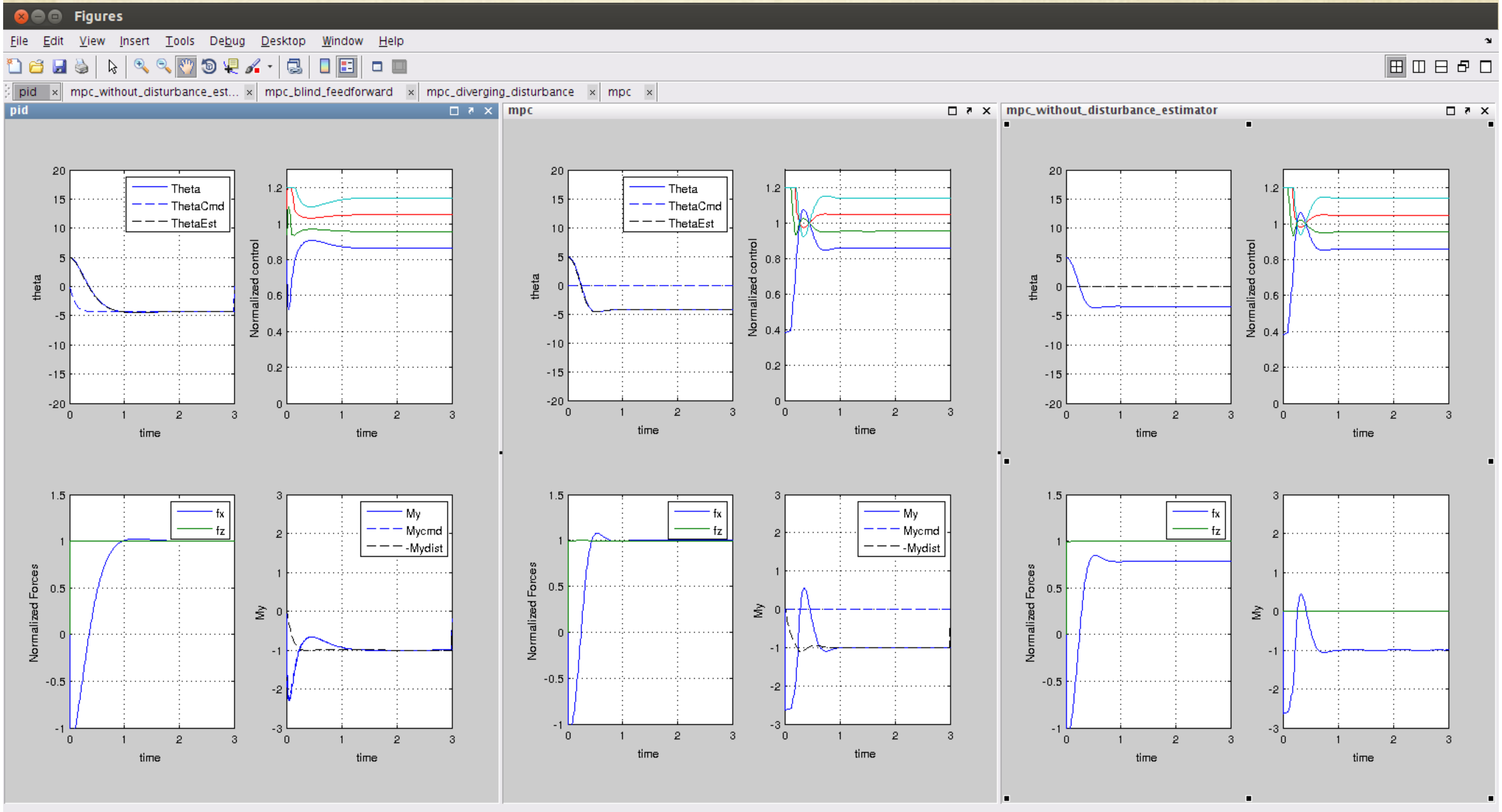
- Add a disturbance My to the system:
  - Feed-forward diverges
- How does one handle disturbances in MPC framework?
  - Literature review turned out two options:
    - Augment state with an integrator (not ideal…)
    - Synthesize a disturbance estimator and include disturbance in dynamics

$$x_{k+1} = Ax_k + Bu_k + B_d\hat{M}_{yd}$$

Game #2


Figures
File Edit View Insert Tools Debug Desktop Window Help
pid
mpc_without_disturbance_est...
mpc_blind_feedforward
mpc_diverging_disturbance
mpc
pid
mpc_blind_feedforward
mpc_diverging_disturbance
Theta
ThetaCmd
ThetaEst
theta
Normalized control
time
fx
fz
Normalized Forces
My
Mycmd
-Mydist

Game #2



Figures
File Edit View Insert Tools Debug Desktop Window Help
pid
mpc_without_disturbance_est...
mpc_blind_feedforward
mpc_diverging_disturbance
mpc
mpc_without_disturbance_estimator
Theta
ThetaCmd
ThetaEst
theta
time
Normalized control
fx
fz
Normalized Forces
My
Mycmd
-Mydist

Game #3

- Include a stabilizing final set constraint

$$x_N = Ax_N + Bu_{N-1} + B_d\hat{M}_{yd}$$

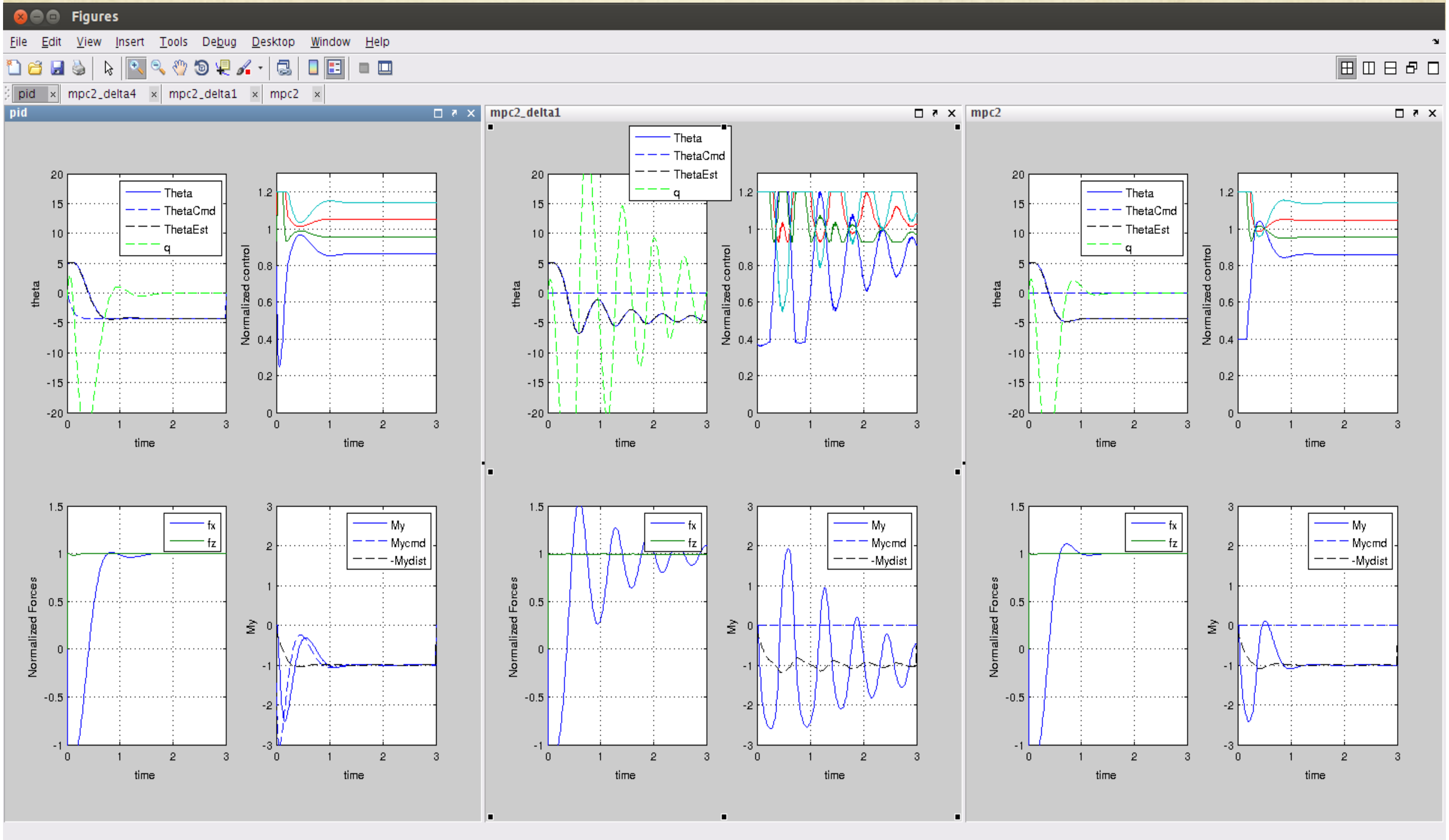
- Include more dynamics:
  - So far we ignored actuator dynamics. These are pretty important and performance is very bad if not accounted for; but to take them into account we introduce a lot of states:

$$x_k = \left[\theta, q, t1, \dot{t1}, \ldots, t4, \dot{t4}\right]_k^t$$

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

$$\begin{aligned} \underset{U}{\text{minimize}} \quad & J(U) = \frac{1}{2}U^t H U + c^t U \\ \text{subject to} \quad & GU = h \\ & FU \leq e \end{aligned}$$

Game #3



Figures
File Edit View Insert Tools Debug Desktop Window Help
pid
mpc2_delta4
mpc2_delta1
mpc2
Theta
ThetaCmd
ThetaEst
q
theta
time
Normalized control
fx
fz
Normalized Forces
My
Mycmd
-Mydist

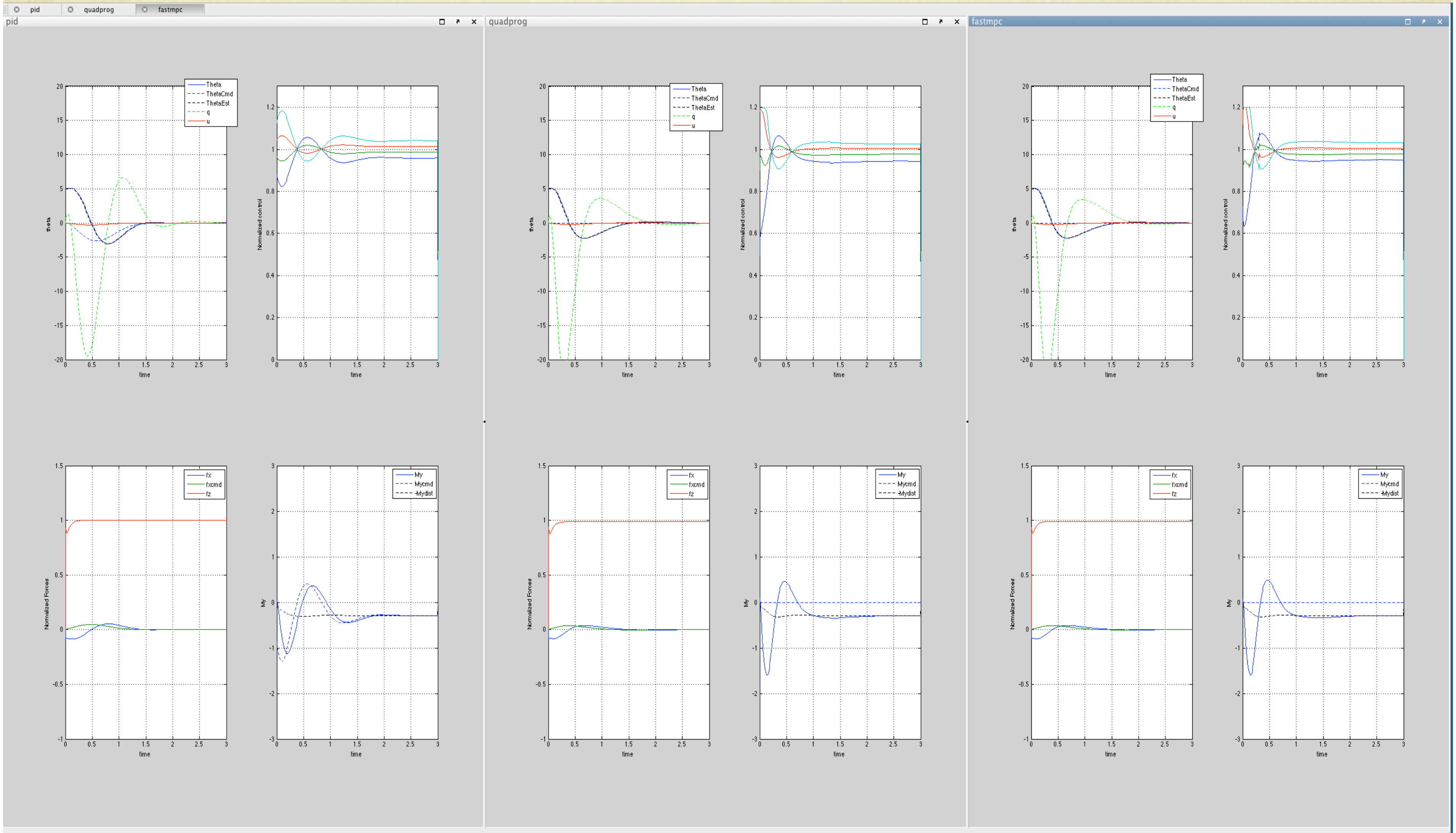
Game #4-5

- 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

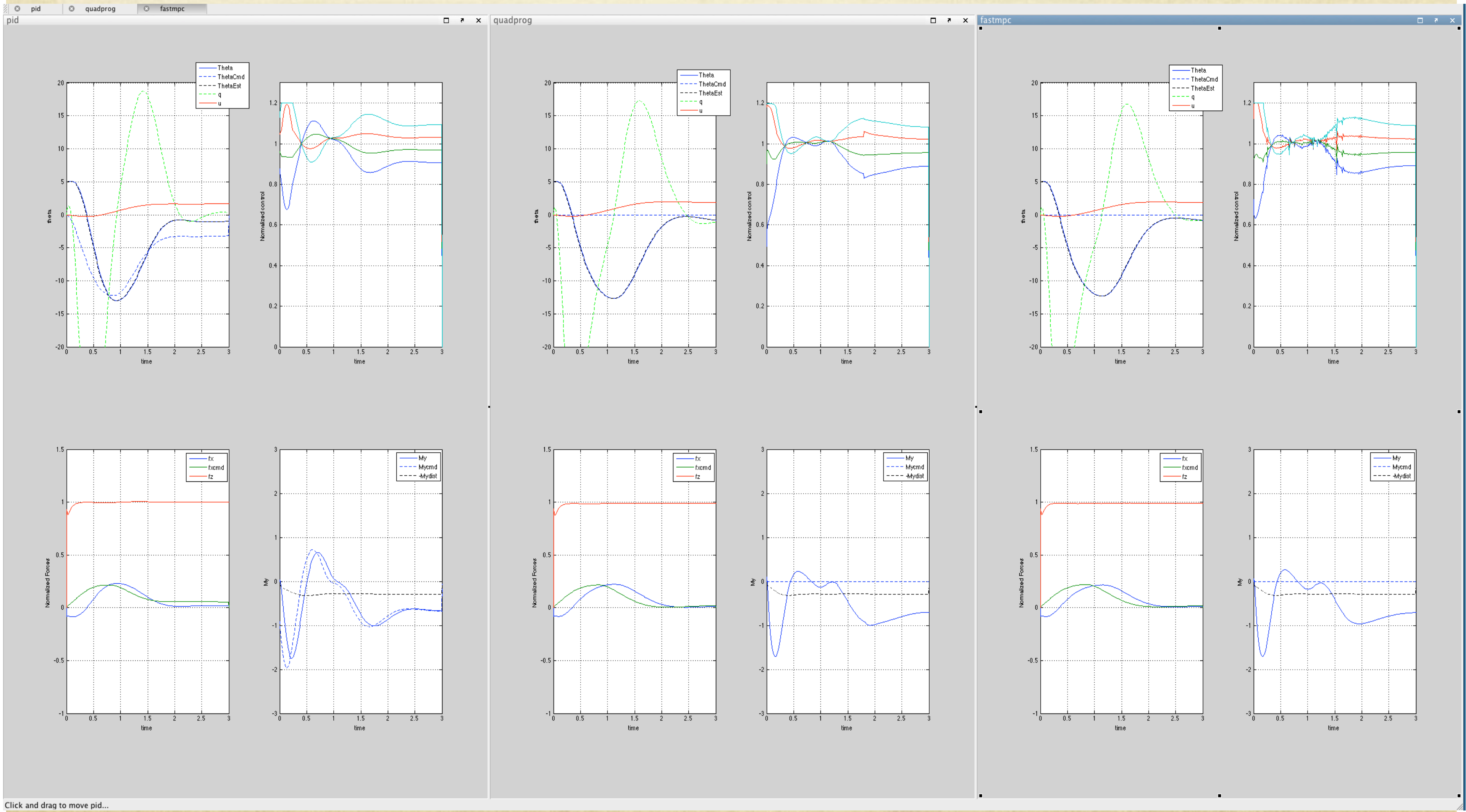
- Do it as fast as possible
  - Look into ‘inexact’ and fast solvers tailored for MPC
    - Work by Y.Wang and S.Boyd might be relevant
  - Investigate the trade-off between performance and computational delay w.r.t. changes in control rate, planning and re-planning horizon lengths, number of constraints,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



pid
quadprog
fastmpc
Theta
ThetaCmd
ThetaEst
q
u
theta
time
Normalized control
fx
fxcmd
fz
Normalized Forces
My
Mycmd
-Mydist

Game #4-5



pid
quadprog
fastmpc
Theta
ThetaCmd
ThetaEst
q
u
theta
time
Normalized control
fx
fxcmd
fz
Normalized Forces
My
Mycmd
-Mydist
Click and drag to move pid...

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;
  - but so far no one has used it on a flying vehicle online?

Equations ...

- Objective: J(x(0),U_0) = \overset{N-1}{\underset{k=0}{\Sigma}}(x^t_kQx_k + u^t_kRu_k) + x^t_NPx_N

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

- EOM: \dot{x} = \left[\begin{array}{c}\dot{\theta} \\\dot{q}\end{array}\right] = \left [\begin{array}{cc}0 & 1\\0 & 0\end{array}\right]\left[\begin{array}{c}\theta \\q\end {array}\right] + \frac{1}{I_{yy}}\left[\begin{array}{cccc}0 & 0 & 0 & 0\\d_1 & d_2 & d_3 & d_4\end{array}\right]\left[\begin{array}{c}t_1 \\\ldots\\t_4\end{array}\right] \\ \left[\begin{array}{c}F_{x,i} \\F_{z,i}\end{array}\right] = \left[\begin{array}{cc}\cos \theta & -\sin \theta \\-\sin \theta & -\cos \theta\end{array}\right]\left[\begin{array} {c}0 \\\Sigma t_i\end{array}\right]

- 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}\\