16.323 Principles of Optimal Control
Spring 2008

16.323 Lecture 16

Model Predictive Control

- Allgower, F., and A. Zheng, Nonlinear Model Predictive Control, Springer-Verlag, 2000.
- Camacho, E., and C. Bordons, Model Predictive Control, Springer-Verlag, 1999.
- Kouvaritakis, B., and M. Cannon, Non-Linear Predictive Control: Theory & Practice, IEE Publishing, 2001.
- Maciejowski, J., Predictive Control with Constraints, Pearson Education POD, 2002.
- Rossiter, J. A., Model-Based Predictive Control: A Practical Approach, CRC Press, 2003.

MPC

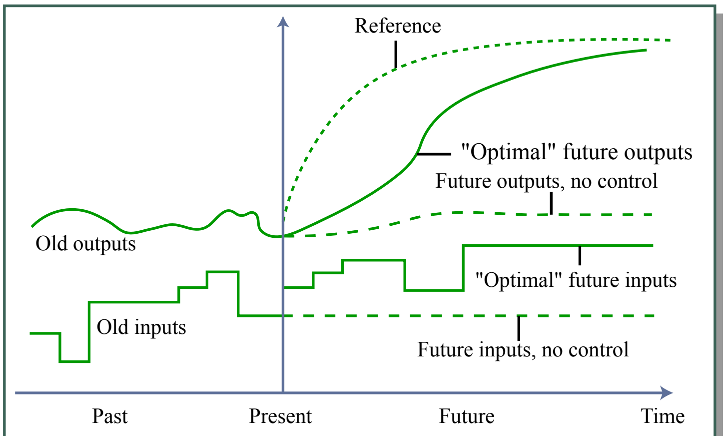
- Planning in Lecture 8 was effectively "open-loop"
  - Designed the control input sequence $\mathbf{u}(t)$ using an assumed model and set of constraints.
  - Issue is that with modeling error and/or disturbances, these inputs will not necessarily generate the desired system response.

- Need a "closed-loop" strategy to compensate for these errors.
  - Approach called **Model Predictive Control**
  - Also known as **receding horizon control**

- Basic strategy:
  - At time k, use knowledge of the system model to design an input sequence

$$\mathbf{u}(k|k), \mathbf{u}(k+1|k),\ \mathbf{u}(k+2|k),\ \mathbf{u}(k+3|k), \ldots, \mathbf{u}(k+N|k)$$

    over a finite horizon N from the current state $\mathbf{x}(k)$
  - Implement a fraction of that input sequence, usually just first step.
  - Repeat for time $k+1$ at state $\mathbf{x}(k+1)$



MPC: basic idea (from Bo Wahlberg)

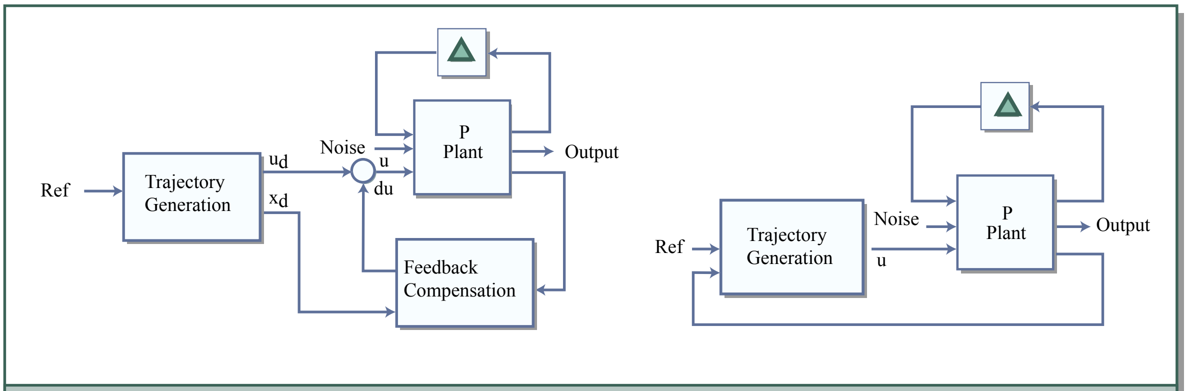
Figure by MIT OpenCourseWare.

- Note that the control algorithm is based on numerically solving an optimization problem at each step
  – Typically a constrained optimization

- Main advantage of MPC:
  – Explicitly accounts for system constraints.
    ◇ Doesn't just design a controller to keep the system away from them.
  – Can easily handle nonlinear and time-varying plant dynamics, since the controller is explicitly a function of the model that can be modified in real-time (and plan time)

- Many commercial applications that date back to the early 1970's, see http://www.che.utexas.edu/~qin/cpcv/cpcv14.html
  – Much of this work was in process control - very nonlinear dynamics, but not particularly fast.

- As computer speed has increased, there has been renewed interest in applying this approach to applications with faster time-scale: trajectory design for aerospace systems.



Implementation architectures for MPC (from Mark Milam)

Figure by MIT OpenCourseWare.

Basic Formulation

- Given a set of plant dynamics (assume linear for now)

$$\begin{aligned}\mathbf{x}(k+1) &= A\mathbf{x}(k) + B\mathbf{u}(k)\\ \mathbf{z}(k) &= C\mathbf{x}(k)\end{aligned}$$

  and a cost function

$$J = \sum_{j=0}^{N}\{\|\mathbf{z}(k+j|k)\|_{R_{zz}} + \|\mathbf{u}(k+j|k)\|_{R_{uu}}\} + F(\mathbf{x}(k+N|k))$$

  - $\|\mathbf{z}(k+j|k)\|_{R_{xx}}$ is just a short hand for a weighted norm of the state, and to be consistent with earlier work, would take

$$\|\mathbf{z}(k+j|k)\|_{R_{zz}} = \mathbf{z}(k+j|k)^T R_{zz}\mathbf{z}(k+j|k)$$

  - $F(\mathbf{x}(k+N|k))$ is a terminal cost function

- Note that if $N \to \infty$, and there are no additional constraints on $\mathbf{z}$ or $\mathbf{u}$, then this is just the discrete LQR problem solved on page 3–14.
  - Note that the original LQR result could have been written as just an input control sequence (**feedforward**), but we choose to write it as a linear state **feedback**.
  - In the nominal case, there is no difference between these two implementation approaches (feedforward and feedback)
  - But with modeling errors and disturbances, the state feedback form is much less sensitive.

$\Rightarrow$ This is the main reason for using feedback.

- **Issue:** When limits on $\mathbf{x}$ and $\mathbf{u}$ are added, we can no longer find the general solution in analytic form $\Rightarrow$ must solve it numerically.

- However, solving for a very long input sequence:
  - Does not make sense if one expects that the model is wrong and/or there are disturbances, because it is unlikely that the end of the plan will be implemented (a new one will be made by then)
  - Longer plans have more degrees of freedom and take much longer to compute.

- Typically design using a small $N \Rightarrow$ short plan that does not necessarily achieve all of the goals.
  - Classical hard question is how large should N be?
  - If plan doesn't reach the goal, then must develop an estimate of the remaining **cost-to-go**

- Typical problem statement: for finite N $(F = 0)$

$$
\begin{aligned}
\min_u J &= \sum_{j=0}^{N} \{ \|\mathbf{z}(k+j|k)\|_{R_{\rm zz}} + \|\mathbf{u}(k+j|k)\|_{R_{\rm uu}} \} \\
\text{s.t.} \quad \mathbf{x}(k+j+1|k) &= A\mathbf{x}(k+j|k) + B\mathbf{u}(k+j|k) \\
\mathbf{x}(k|k) &\equiv \mathbf{x}(k) \\
\mathbf{z}(k+j|k) &= C\mathbf{x}(k+j|k)
\end{aligned}
$$

$$
\text{and} \quad |\mathbf{u}(k+j|k)| \leq u_m
$$

- Consider converting this into a more standard optimization problem.

$$
\begin{aligned}
\mathbf{z}(k|k) &= C\mathbf{x}(k|k) \\
\mathbf{z}(k+1|k) &= C\mathbf{x}(k+1|k) = C(A\mathbf{x}(k|k) + B\mathbf{u}(k|k)) \\
&= CA\mathbf{x}(k|k) + CB\mathbf{u}(k|k) \\
\mathbf{z}(k+2|k) &= C\mathbf{x}(k+2|k) \\
&= C(A\mathbf{x}(k+1|k) + B\mathbf{u}(k+1|k)) \\
&= CA(A\mathbf{x}(k|k) + B\mathbf{u}(k|k)) + CB\mathbf{u}(k+1|k) \\
&= CA^2\mathbf{x}(k|k) + CAB\mathbf{u}(k|k) + CB\mathbf{u}(k+1|k) \\
&\vdots \\
\mathbf{z}(k+N|k) &= CA^N\mathbf{x}(k|k) + CA^{N-1}B\mathbf{u}(k|k) + \cdots \\
&\quad + CB\mathbf{u}(k+(N-1)|k)
\end{aligned}
$$

- Combine these equations into the following:

$$
\begin{bmatrix} \mathbf{z}(k|k) \\ \mathbf{z}(k+1|k) \\ \mathbf{z}(k+2|k) \\ \vdots \\ \mathbf{z}(k+N|k) \end{bmatrix} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^N \end{bmatrix} \mathbf{x}(k|k)
$$

$$
+ \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ CB & 0 & 0 & & 0 \\ CAB & CB & 0 & & 0 \\ \vdots & & & & \\ CA^{N-1}B & CA^{N-2}B & CA^{N-3}B & \cdots & CB \end{bmatrix} \begin{bmatrix} \mathbf{u}(k|k) \\ \mathbf{u}(k+1|k) \\ \vdots \\ \mathbf{u}(k+N-1|k) \end{bmatrix}
$$

- Now define

$$Z(k) \equiv \begin{bmatrix} \mathbf{z}(k|k) \\ \vdots \\ \mathbf{z}(k+N|k) \end{bmatrix} \quad U(k) \equiv \begin{bmatrix} \mathbf{u}(k|k) \\ \vdots \\ \mathbf{u}(k+N-1|k) \end{bmatrix}$$

then, with $\mathbf{x}(k|k) = \mathbf{x}(k)$

$$Z(k) = G\mathbf{x}(k) + HU(k)$$

- Note that

$$\sum_{j=0}^{N} \mathbf{z}(k+j|k)^T R_{zz} \mathbf{z}(k+j|k) = Z(k)^T W_1 Z(k)$$

with an obvious definition of the weighting matrix W_1

- Thus

$$\begin{aligned} Z(k)^T W_1 Z(k) &+ U(k)^T W_2 U(k) \\ &= (G\mathbf{x}(k) + HU(k))^T W_1 (G\mathbf{x}(k) + HU(k)) + U(k)^T W_2 U(k) \\ &= \mathbf{x}(k)^T H_1 \mathbf{x}(k) + H_2^T U(k) + \frac{1}{2} U(k)^T H_3 U(k) \end{aligned}$$

where

$$H_1 = G^T W_1 G, \quad H_2 = 2(\mathbf{x}(k)^T G^T W_1 H), \quad H_3 = 2(H^T W_1 H + W_2)$$

- Then the MPC problem can be written as:

$$\begin{aligned} \min_{U(k)} \tilde{J} &= H_2^T U(k) + \frac{1}{2} U(k)^T H_3 U(k) \\ \text{s.t.} &\quad \begin{bmatrix} I_N \\ -I_N \end{bmatrix} U(k) \leq u_m \end{aligned}$$

Toolboxes

- **Key point:** the MPC problem is now in the form of a standard **quadratic program** for which standard and efficient codes exist.

```
QUADPROG Quadratic programming. %
X=QUADPROG(H,f,A,b) attempts to solve the %
quadratic programming problem:

min 0.5*x'*H*x + f'*x   subject to:  A*x <= b
 x

X=QUADPROG(H,f,A,b,Aeq,beq) solves the problem %
above while additionally satisfying the equality%
constraints Aeq*x = beq.
```

- Several Matlab toolboxes exist for testing these ideas
  – MPC toolbox by Morari and Ricker – extensive analysis and design tools.
  – MPCtools [32] enables some MPC simulation and is free `www.control.lth.se/user/johan.akesson/mpctools/`

[32] Johan Akesson: "MPCtools 1.0 - Reference Manual". Technical report ISRN LUTFD2/TFRT–7613–SE, Department of Automatic Control, Lund Institute of Technology, Sweden, January 2006.

MPC Observations

- Current form assumes that full state is available - can hookup with an estimator

- Current form assumes that we can sense and apply corresponding control immediately
  - With most control systems, that is usually a reasonably safe assumption
  - Given that we must re-run the optimization, probably need to account for this computational delay - different form of the discrete model - see F&P (chapter 2)

- If the constraints are not active, then the solution to the QP is that

$$U(K) = -H_3^{-1} H_2$$

which can be written as:

$$\begin{aligned} u(k|k) &= -\left[\begin{array}{cccc} 1 & 0 & \dots & 0 \end{array}\right] (H^T W_1 H + W_2)^{-1} H^T W_1 G \mathbf{x}(k) \\ &= -K\mathbf{x}(k) \end{aligned}$$

which is just a state feedback controller.

– Can apply this gain to the system and check the eigenvalues.

- What can we say about the stability of MPC when the constraints are active? [33]
  – Depends a lot on the terminal cost and the terminal constraints.[34]

- Classic result:[35] Consider a MPC algorithm for a linear system with constraints. Assume that there are terminal constraints:
  – $\mathbf{x}(k+N|k) = 0$ for predicted state $\mathbf{x}$
  – $\mathbf{u}(k+N|k) = 0$ for computed future control u

  Then if the optimization problem is feasible at time k, $\mathbf{x} = 0$ is stable.

  Proof: Can use the performance index J as a Lyapunov function.
  – Assume there exists a feasible solution at time k and cost J_k
  – Can use that solution to develop a feasible candidate at time $k+1$, by simply adding $\mathbf{u}(k+N+1) = 0$ and $\mathbf{x}(k+N+1) = 0$.
  – **Key point:** can estimate the candidate controller performance

$$\begin{aligned}\tilde{J}_{k+1} &= J_k - \{\|\mathbf{z}(k|k)\|_{R_{\text{zz}}} + \|\mathbf{u}(k|k)\|_{R_{\text{uu}}}\} \\ &\leq J_k - \{\|\mathbf{z}(k|k)\|_{R_{\text{zz}}}\}\end{aligned}$$

  – This candidate is suboptimal for the MPC algorithm, hence J decreases even faster $J_{k+1} \leq \tilde{J}_{k+1}$
  – Which says that J decreases if the state cost is non-zero (observability assumptions) $\Rightarrow$ but J is lower bounded by zero.

- Mayne *et al.* [2000] provides excellent review of other strategies for proving stability – different terminal cost and constraint sets

[33] "Tutorial: model predictive control technology," Rawlings, J.B. American Control Conference, 1999. pp. 662-676

[34] Mayne, D.Q., J.B. Rawlings, C.V. Rao and P.O.M. Scokaert, "Constrained Model Predictive Control: Stability and Optimality," Automatica, 36, 789-814 (2000).

[35] A. Bemporad, L. Chisci, E. Mosca: "On the stabilizing property of SIORHC", Automatica, vol. 30, n. 12, pp. 2013-2015, 1994.

Example: Helicopter

- Consider a system similar to the Quansar helicopter[36]

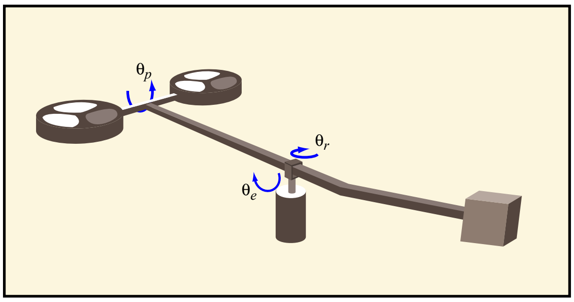


Figure by MIT OpenCourseWare.

- There are 2 control inputs – voltage to each fan V_f, V_b
- A simple dynamics model is that:

$$\begin{aligned}\ddot{\theta}_e &= K_1(V_f + V_b) - T_g/J_e \\ \ddot{\theta}_r &= -K_2 \sin(\theta_p) \\ \ddot{\theta}_p &= K_3(V_f - V_b)\end{aligned}$$

and there are physical limits on the elevation and pitch:

$$-0.5 \leq \theta_e \leq 0.6 \qquad -1 \leq \theta_p \leq 1$$

- Model can be linearized and then discretized $T_s = 0.2$sec.

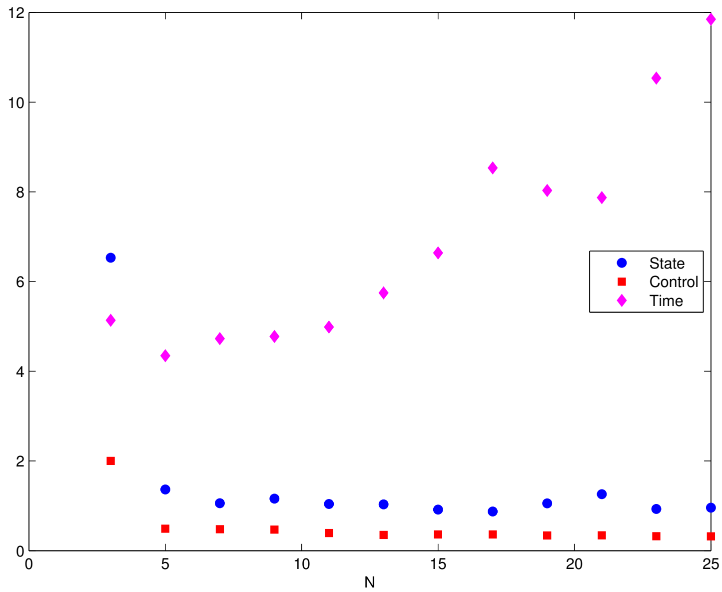


Figure 16.3: Response Summary

[36]ISSN 02805316 ISRN LUTFD2/TFRT- -7613- -SE MPCtools 1.0 Reference Manual Johan Akesson Department of Automatic Control Lund Institute of Technology January 2006

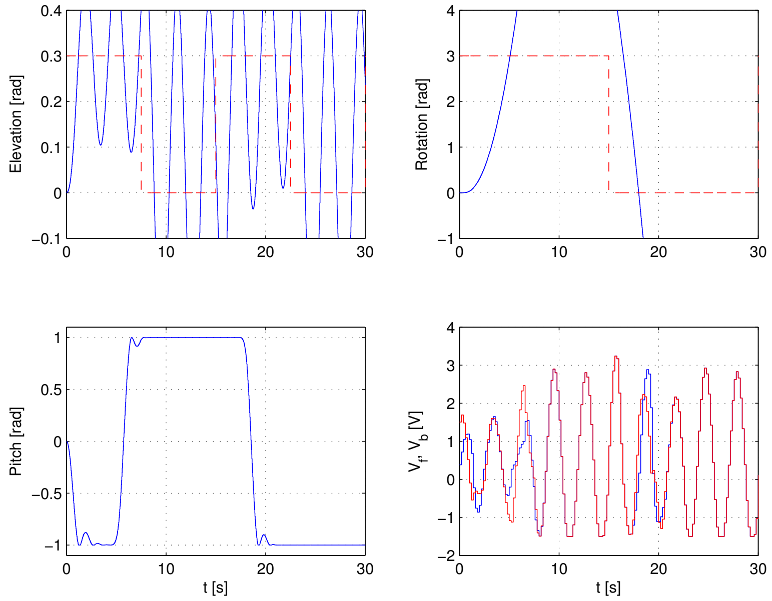


Figure 16.4: Response with $N = 3$

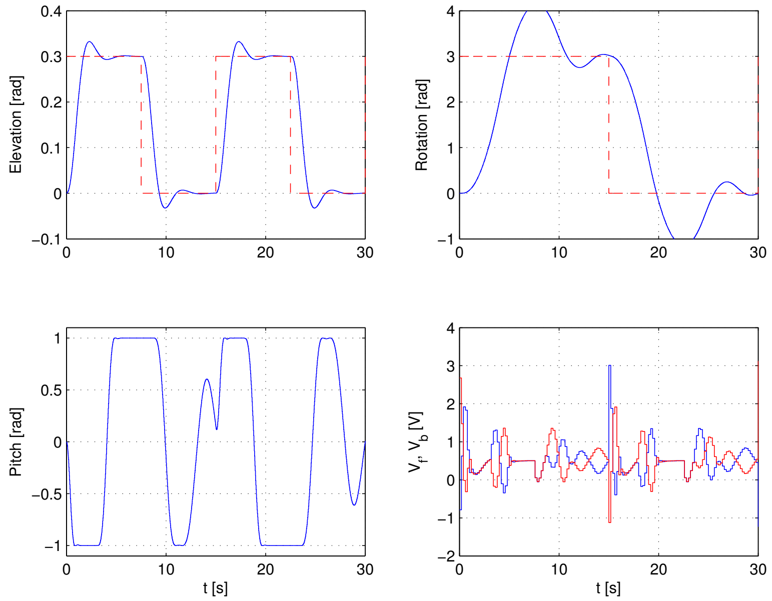


Figure 16.5: Response with $N = 10$

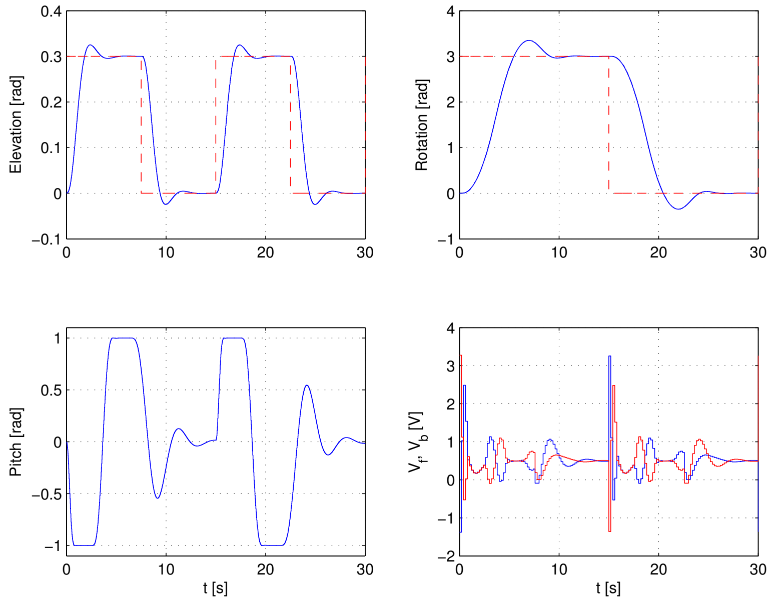


Figure 16.6: Response with $N = 25$