15-884/484 – Control 4: Stochastic Control

J. Zico Kolter

November 14, 2013

Controlling Stochastic Systems

- What happens when the system we are trying to control is stochastic?

$$x_{t+1} = f(x_t, u_t) + \epsilon_t$$

- Remember, stochasticity can arise from "actual" noise in the system, errors in modeling, inaccurate predictions, etc

- We no longer know which state we will end up in, how do we define cost?

- A natural extension, pick $u_{1:T}$ to minimize the *expected* cost

$$J = \mathbf{E}\left[\sum_{t=1}^{H} C(x_t, u_t)\right] = \sum_{t=1}^{H} \mathbf{E}[C(x_t, u_t)]$$

- Not the only possibility, for example might want to minimize the probability that cost exceeds some threshold

$$J = P\left(\sum_{t=1}^{H} C(x_t, u_t) > c\right)$$

- However, expected cost is quite common, as it is often (relatively) easy to optimize

- **The bad news:** The general stochastic control problem can be extremely hard to solve optimally

- **The good news:** Certain special cases can be solved exactly (LQ control with Gaussian noise, Markov Decision Processes); an optimization-based approach known as *Model Predictive Control* (MPC) works well in many cases (even though it is sub-optimal)

Linear Quadratic Stochastic Control

- Linear quadratic setting: linear dynamics and quadratic costs, now with (zero-mean Gaussian) random noise

$$x_{t+1} = Ax_t + Bu_t + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \Sigma)$$

$$J = \mathbf{E}\left[\sum_{t=1}^{H} \left(\|Qx_t\|_2^2 + \|Ru_t\|_2^2\right)\right]$$

- A remarkable fact: optimal stochastic solution is the same as LQR for deterministic setting (just ignore noise entirely)

```
K = dlqr(A, B, Q'*Q, R'*R);
```

- Basic idea: as discussed before for linear-Gaussian systems, expected state of system is unaffected by noise

$$\mathbf{E}[x_{t+1}] = A\mathbf{E}[x_t] + Bu_t$$

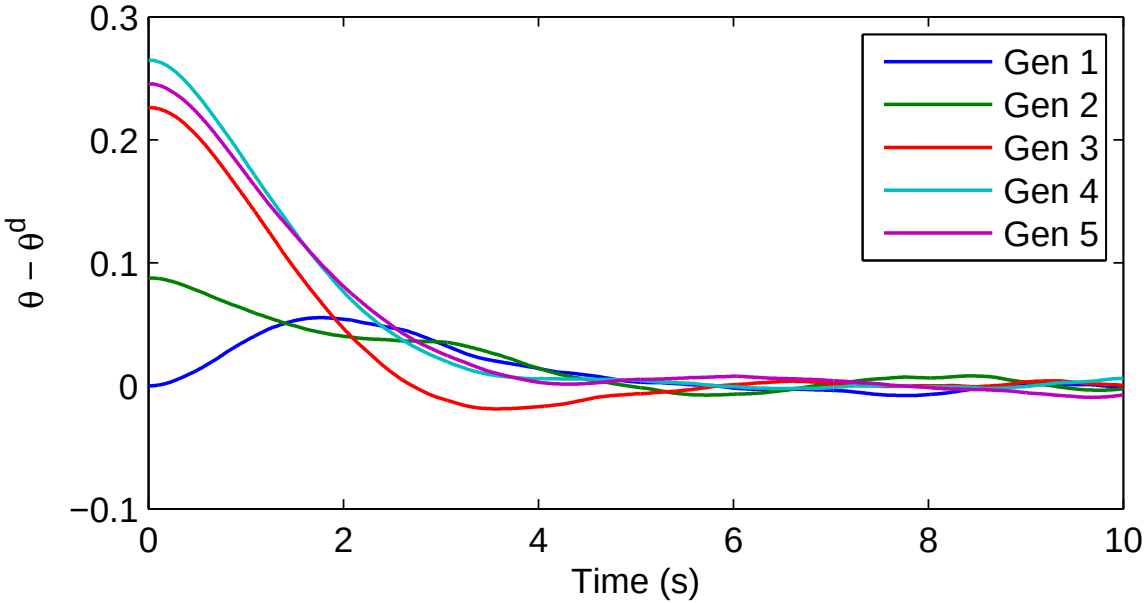
- Further, expected value of quadratic cost is just quadratic cost of expected state plus a constant

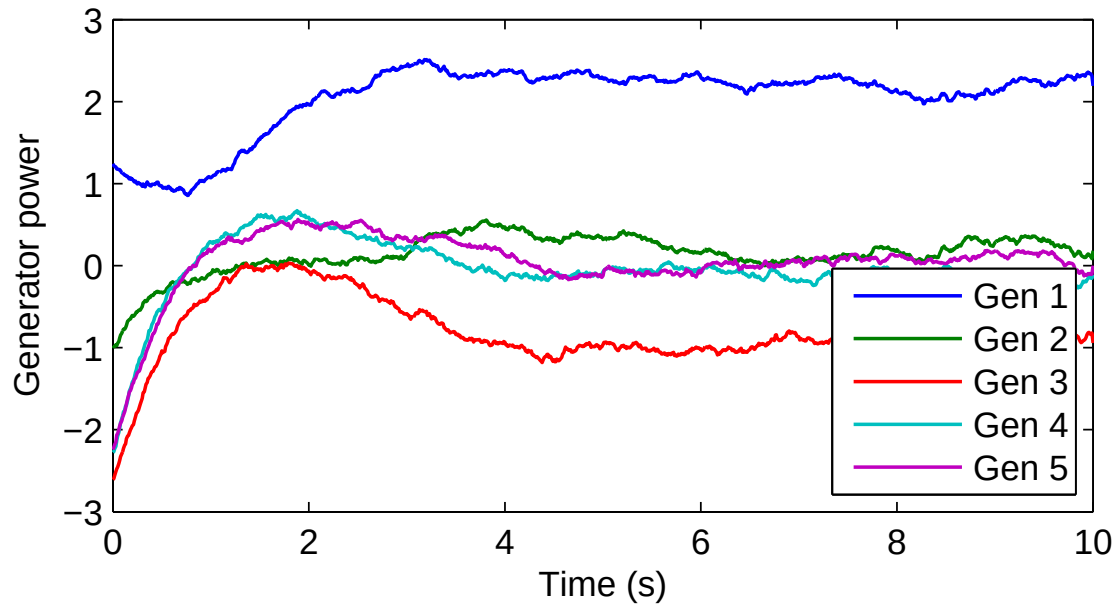
$$\mathbf{E}[\|Qx_t\|_2^2] = \|QE[x_t]\|_2^2 + c$$

  (constant term does depend on the distribution of x_t)

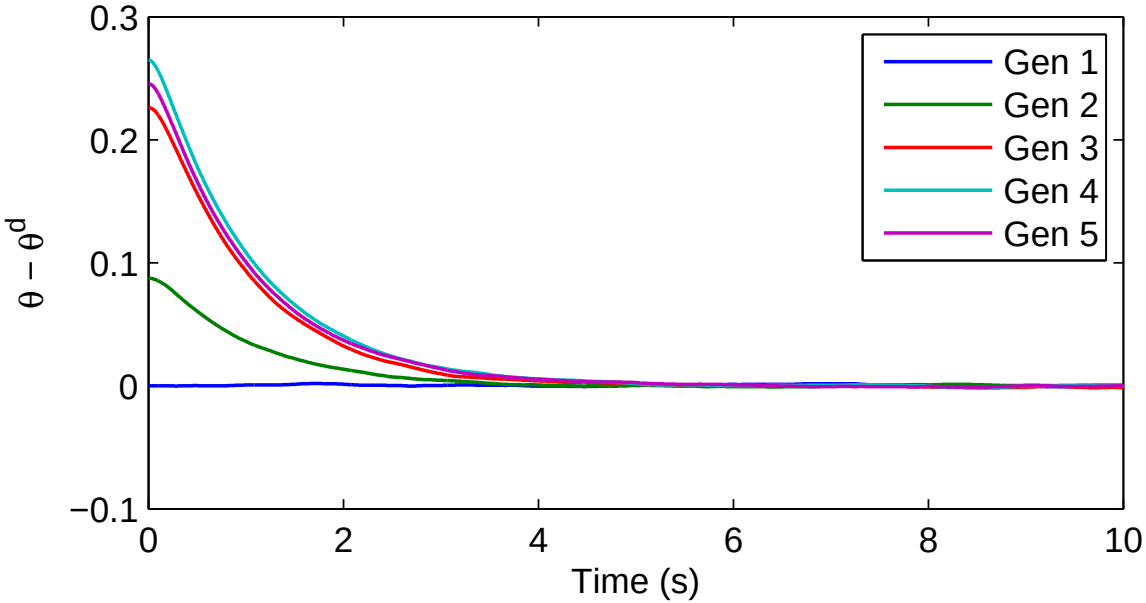
- Derivation of LQR solution ends up not depending on this constant term at all, just adds a fixed amount to the total expected cost

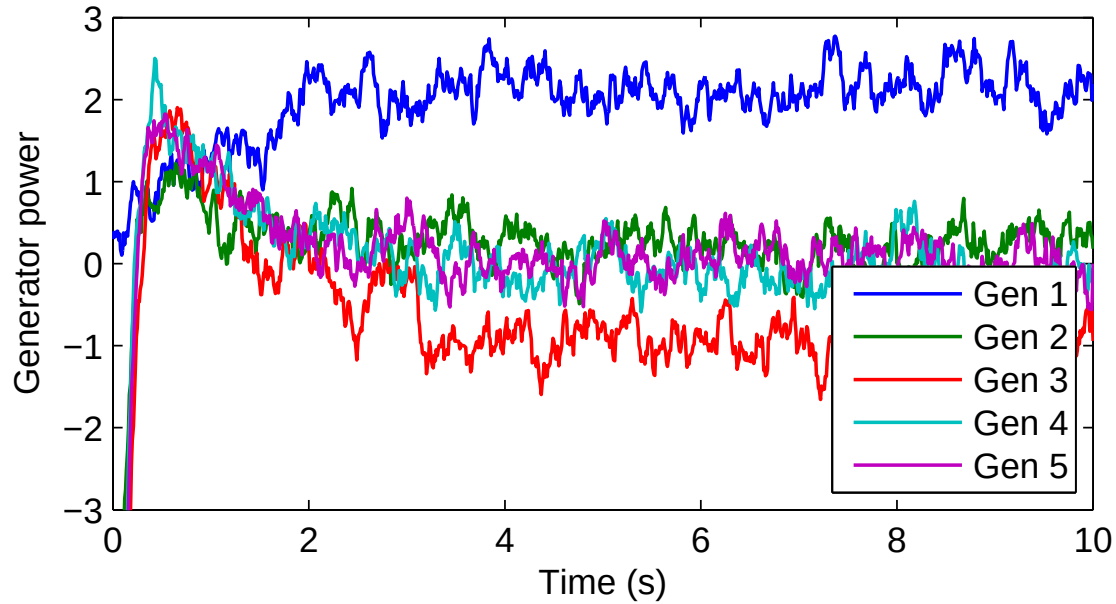
- Example: generator control with random disturbances to generator angular velocities (i.e., random changes to loads)





- Can regulate generators more closely around desired point (at the expense of larger inputs)





- Extension: partially observable stochastic linear quadratic Gaussian:

$$x_{t+1} = Ax_t + Bu_t + \epsilon_t \;\; \epsilon_t \sim \mathcal{N}(0, \Sigma)$$
$$y_t = Cx_t + q_t \;\; q_t \sim N(0, Q)$$

- In this setting the following procedure is optimal:

  1. Estimate current state $\hat{x}_t$ using Kalman filter (recall, like a least-squares estimate of x_t

  2. Apply LQR controller to state estimate $u_t = K\hat{x}_t$

- The fact that this is optimal is known as the *separation principle*, resulting algorithm called LQG

- *Very* rare for such procedures to be optimal

Aside: Markov Decision Processes

- Recall setting for Markov Decision Process:
  - **State**: $x_t \in \{1, 2, \ldots, n\}$
  - **Control**: $u_t \in \{1, 2, \ldots, m\}$
  - **Transition probabilities**: $P^u \in \mathbb{R}^{n \times n}$, $u = 1, \ldots, m$

$$p(x_{t+1} = j | x_t = i, u_t = k) = P^k_{i,j}$$

  - **Cost**: $C : \{1, \ldots, n\} \times \{1, \ldots, m\} \to \mathbb{R}$

- Approach (Value Iteration): iteratively build optimal cost-to-go function $J^\star$ (also called value function)
  - Initialize
$$J_T^\star(x) = \min_u C(x,u), \quad \forall x = 1, \ldots, n$$
  - Repeat for $t = T-1, \ldots, 1$
$$J_t^\star(x) = \min_u \left\{ C(x,u) + \sum_{x'=1}^{n} P_{x,x'}^u J_{t+1}^\star(x') \right\}$$
$$\forall x = 1, \ldots, n$$
  - Optimal policy is then
$$\pi_t^\star(x) = \arg\min_u J_t^\star(x)$$

- Again, method is only tractable for relatively small systems, so that total number of states n and m can be enumerated

- For continuous-state systems, typically discretizations will require a number of states that grows exponentially in system dimension

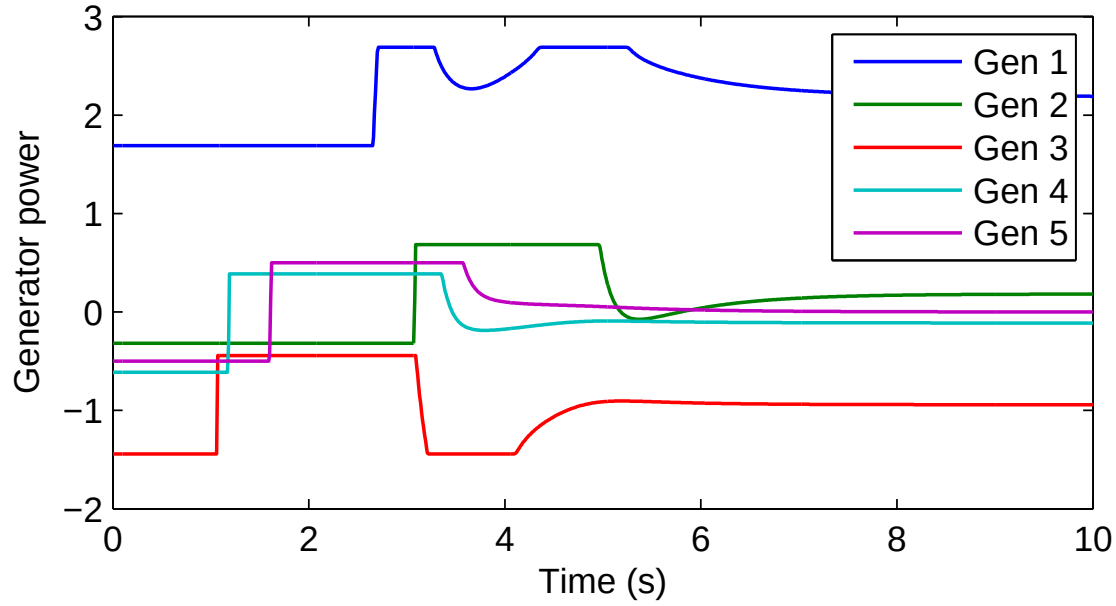
- Unlike LQG setting, partially observable systems are *much* more challenging; setting known as Partially Observable MDPs (POMDPs)

Model Predictive Control

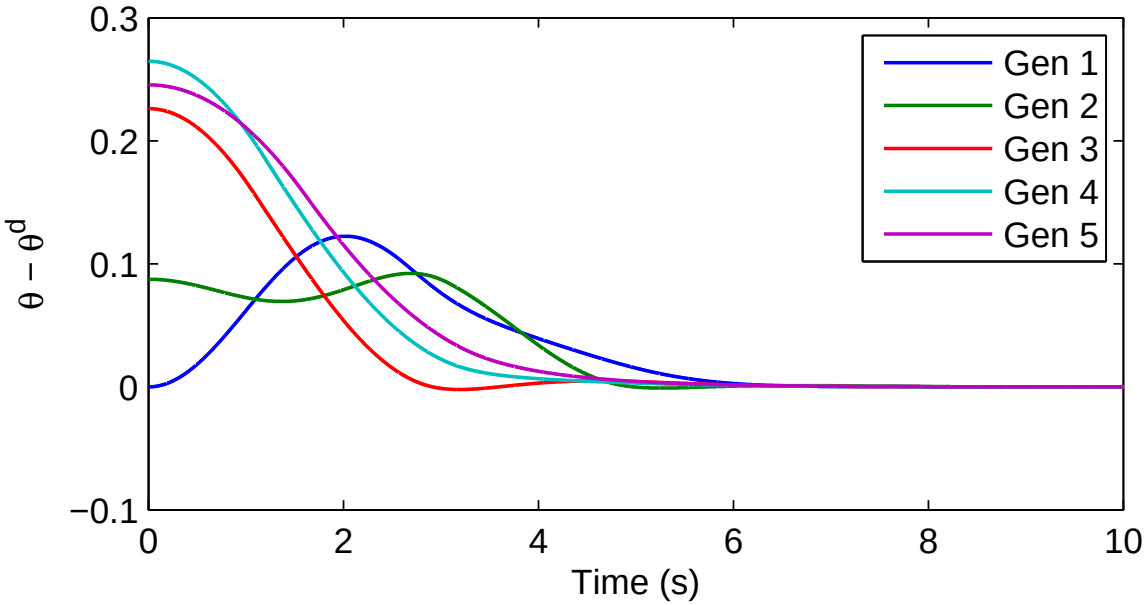
- How can we use the optimization approach to control in stochastic systems?

- Executing the controls we get back from our optimization problem can cause system to diverge

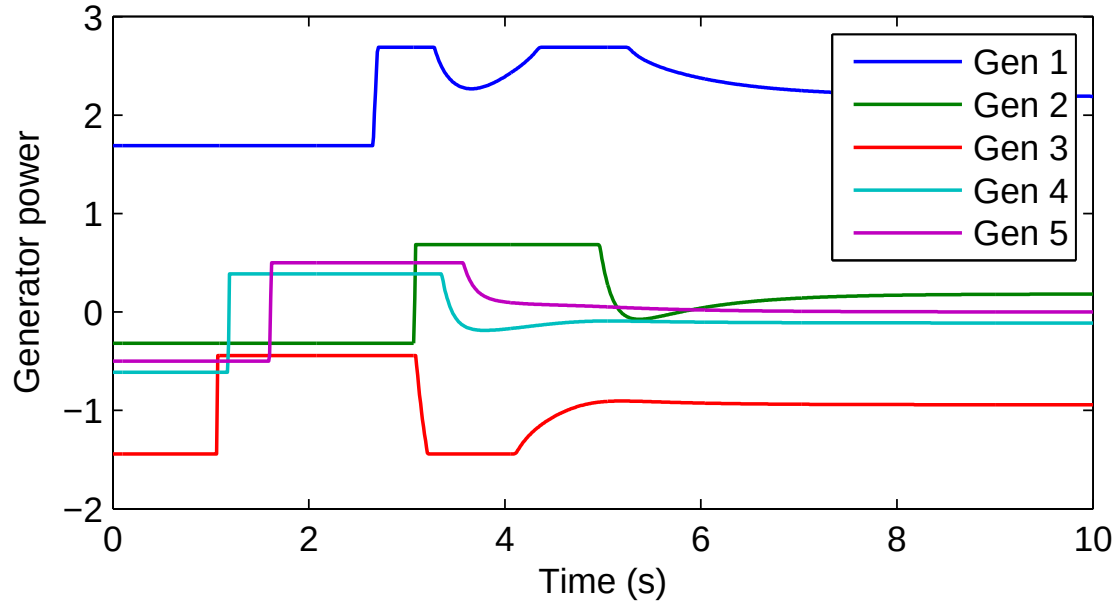
- Executing controls



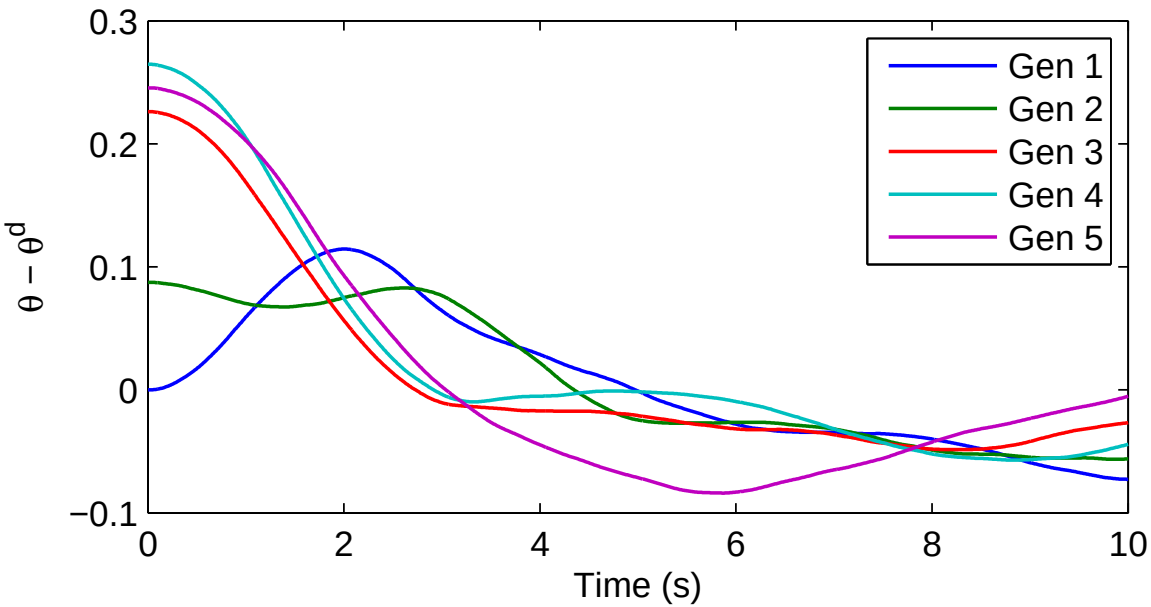
- What optimization method predicts



- Executing controls



- What actually happens



- The model predictive control (MPC) solution: re-run our optimization-based control approach at every time point, execute just the first returned control u_1

- Repeat for $t = 1, \ldots, T$:

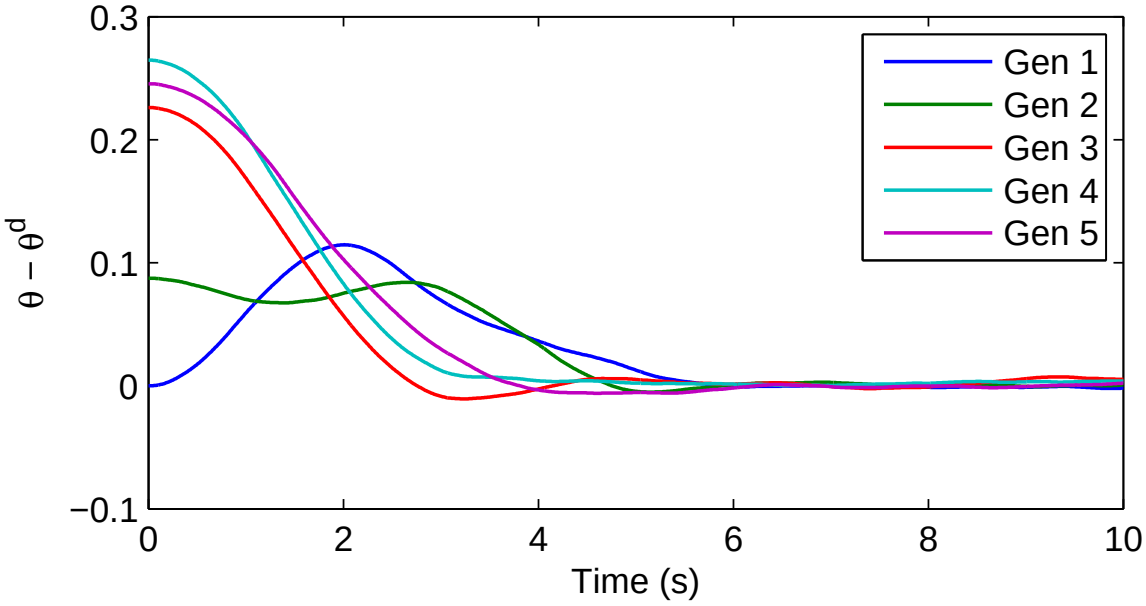
  1. Observe state of system x_t

  2. Solve optimization problem

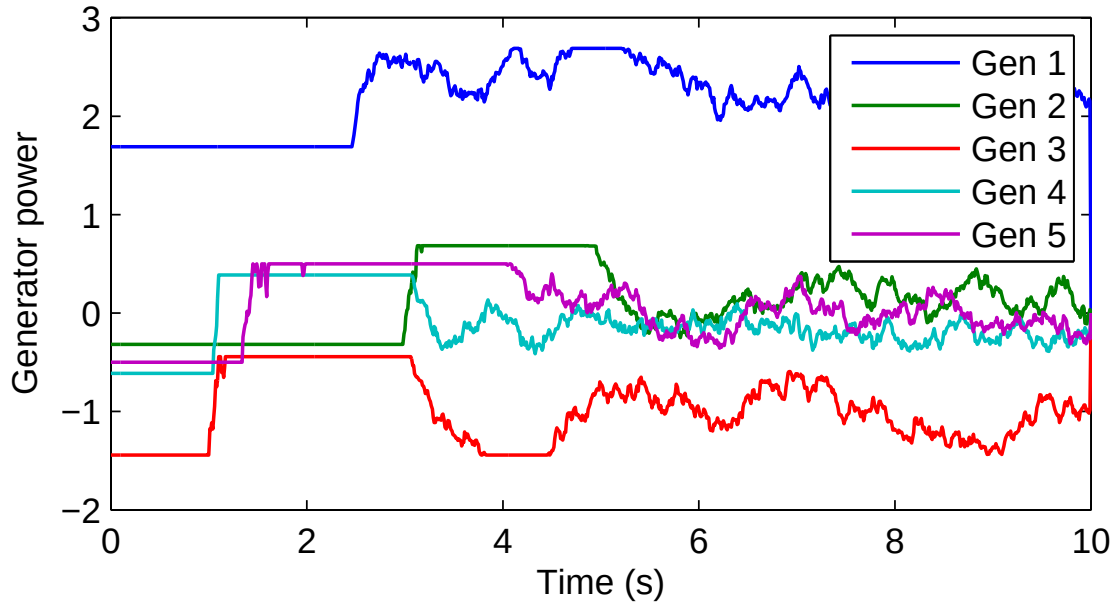
$$\begin{aligned} \underset{\tilde{x}_{1:H}, \tilde{u}_{1:H}}{\text{minimize}} \quad & \sum_{\tau=1}^{H} C(\tilde{x}_\tau, \tilde{u}_\tau) \\ \text{subject to} \quad & \tilde{x}_{\tau+1} = A\tilde{x}_\tau + B\tilde{u}_\tau \\ & \tilde{x}_1 = x_t \\ & \underline{u} \leq \tilde{u}_\tau \leq \overline{u}, \;\; \underline{x} \leq \tilde{x}_\tau \leq \overline{x} \end{aligned}$$

  3. Execute the first control input $u_t = \tilde{u}_1$

- Resulting trajectories from MPC



- Control inputs applied



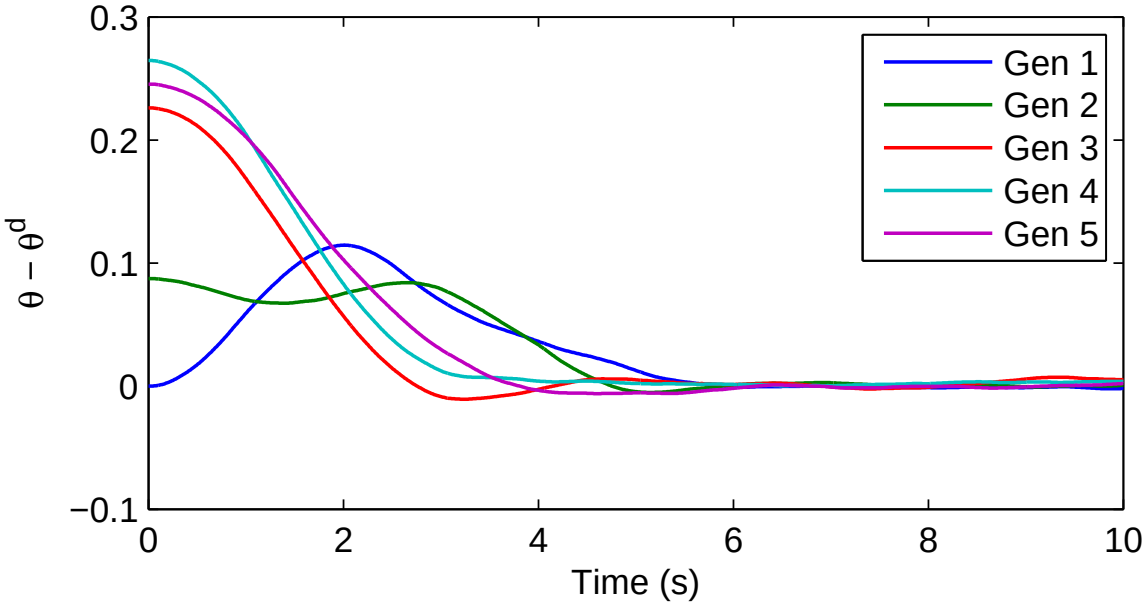
- Resolving optimization problem at each time point allows the system to adjust for actual evolution, deviations from expected states

- But remember, this requires solving an entire optimization problem at every time step (compare to LQR, which requires just a matrix multiplication)
  - Five generator system (very small as control tasks go)
  - For integration time step of 0.01 seconds, optimization over horizon of 10 seconds takes 0.25 second (using CPLEX solver directly)
  - We couldn't run this in real time even on this small system

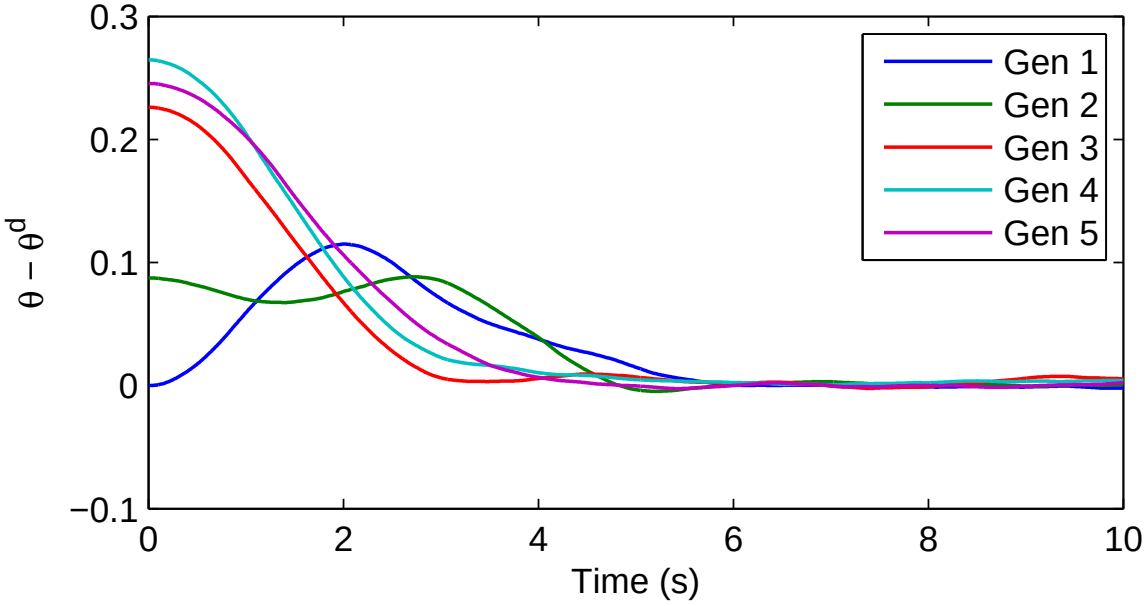
Speeding up MPC

- Speed-up #1: shorter optimization horizon
- Remember that MPC effectively had two horizons:
  1. T: actual horizon of control task
  2. H: horizon we use for optimization
- Often, we can use a much shorter horizon in the optimization problem (intuition: states far in the future won't affect what we do now very much)

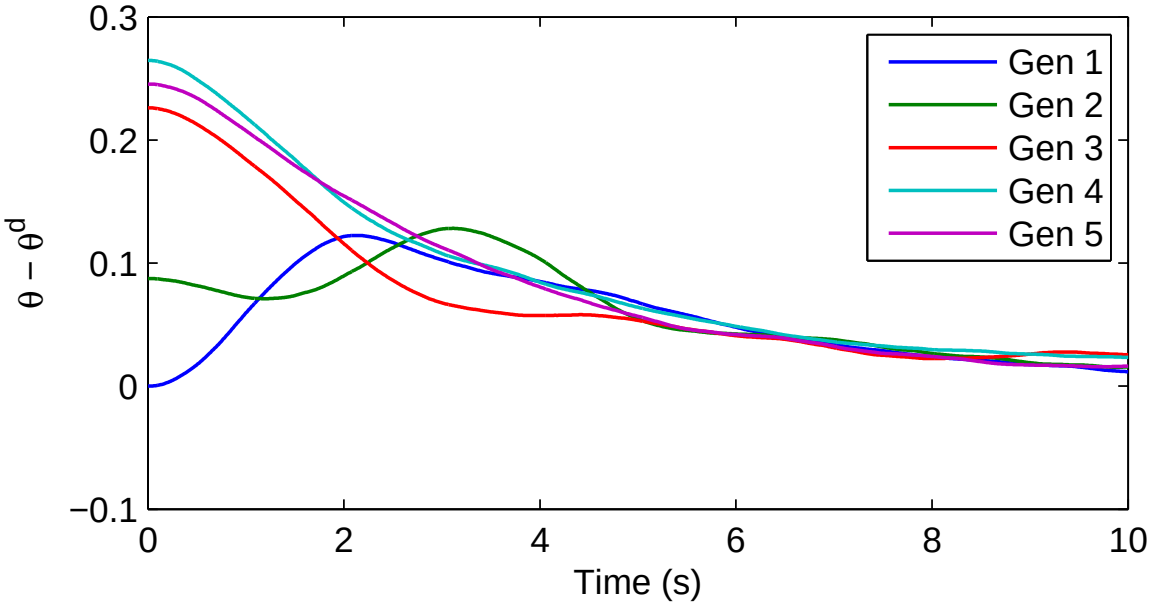
- $H = 10$ seconds (same as T)



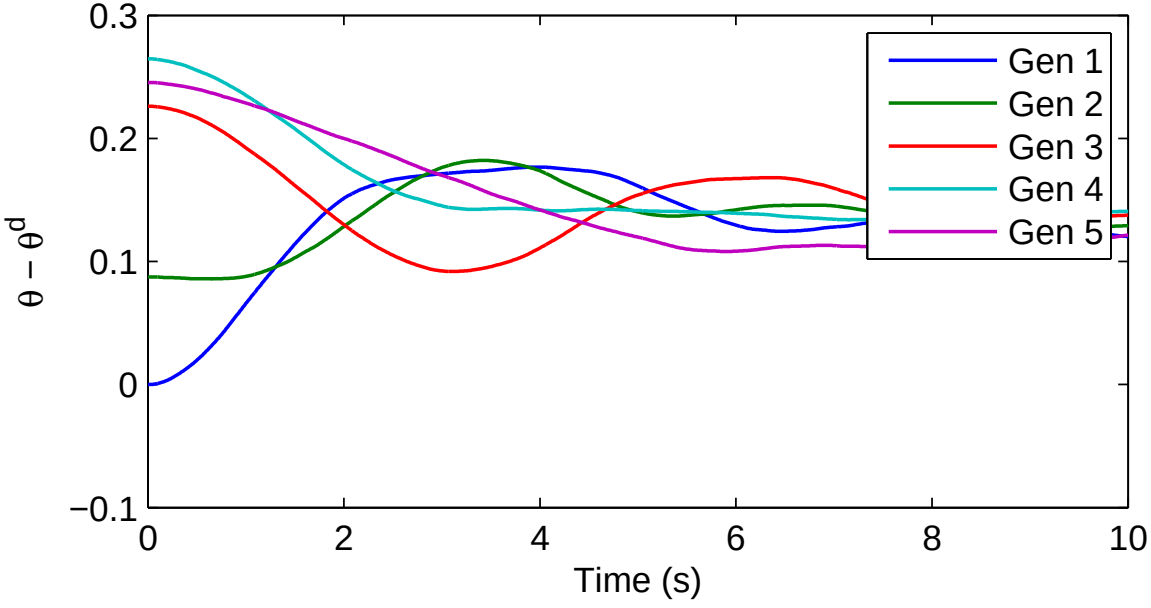
- $H = 2$ seconds



- $H = 0.5$ seconds

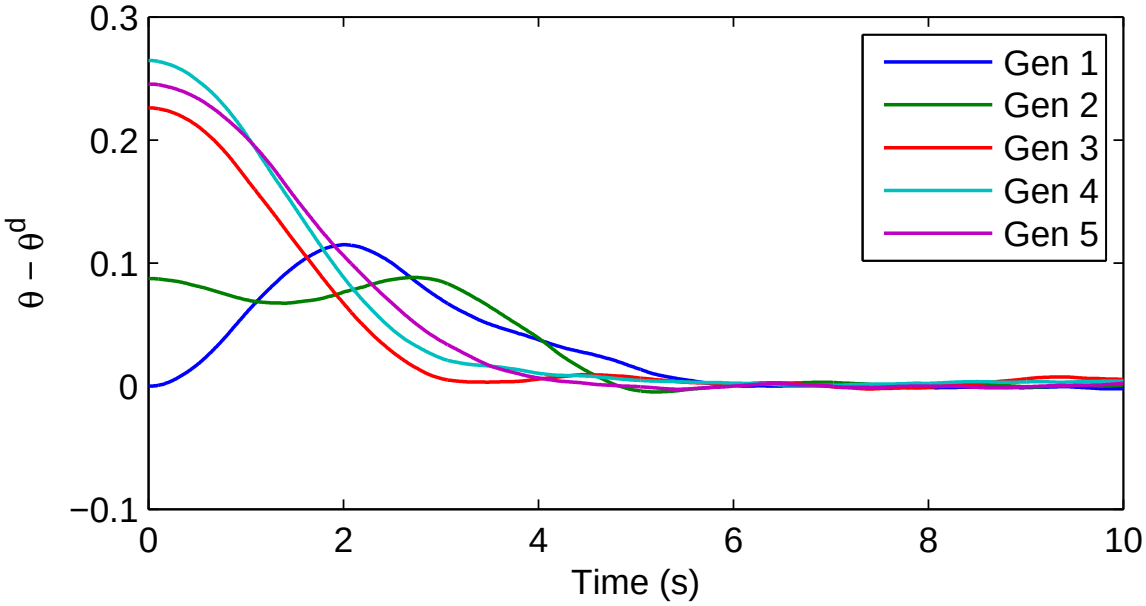


- $H = 0.1$ seconds

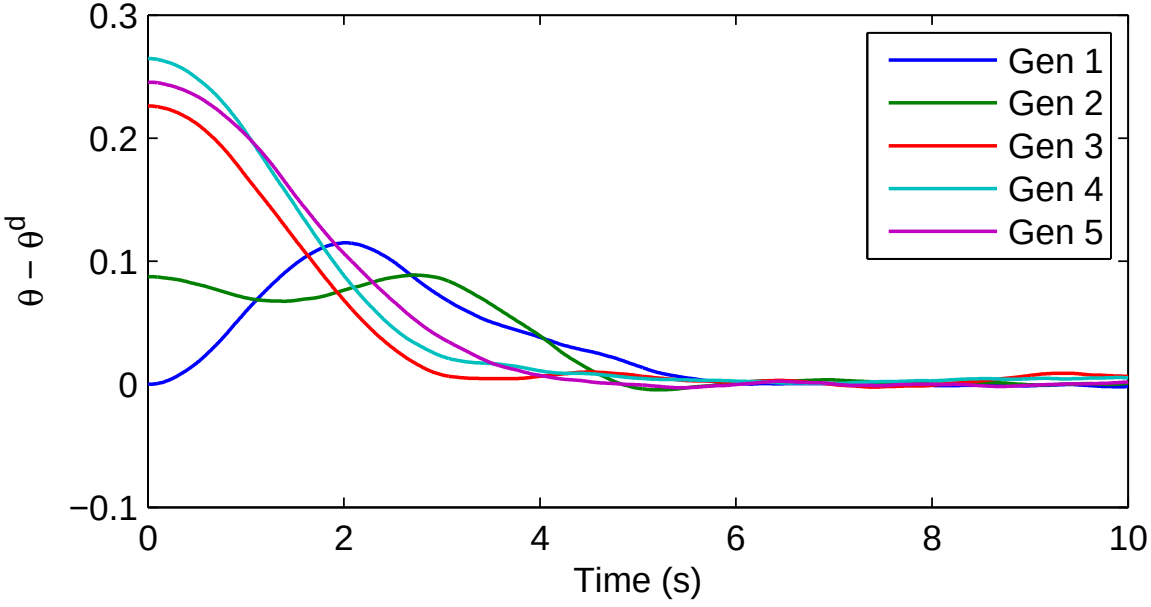


- Speed-up #2: less frequent updating
- In MPC we recompute the controls based upon observed states, but there is no need to do this at every single time step
- Instead, compute updates only every k time steps
- Intuition: over short time periods, effects of stochasticity aren't that bad

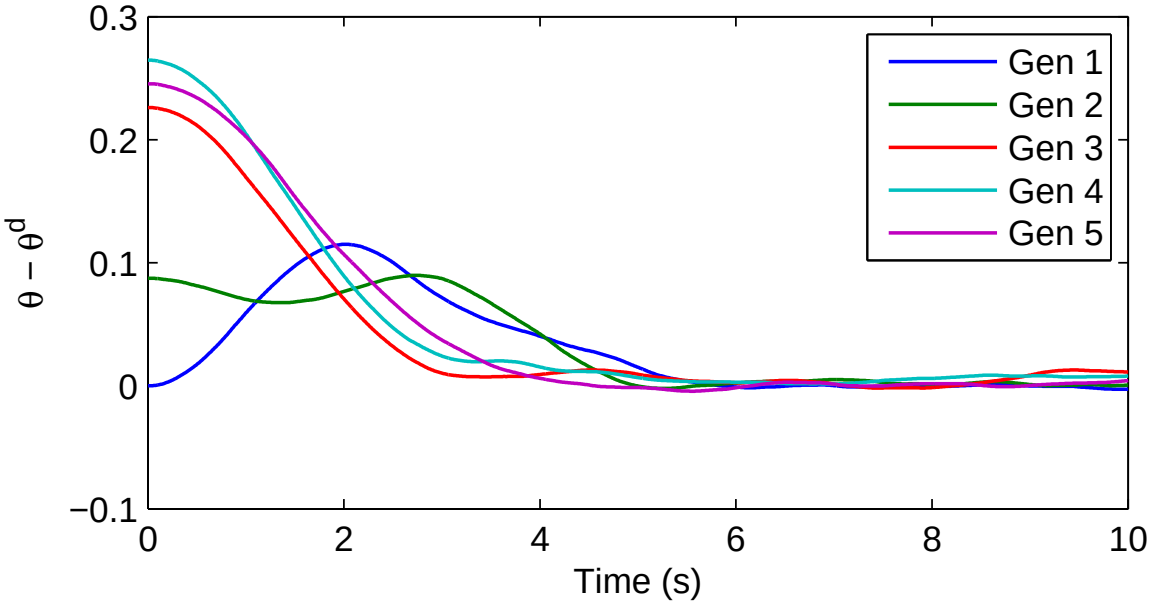
- Update every 0.01 seconds (same as integration time step)



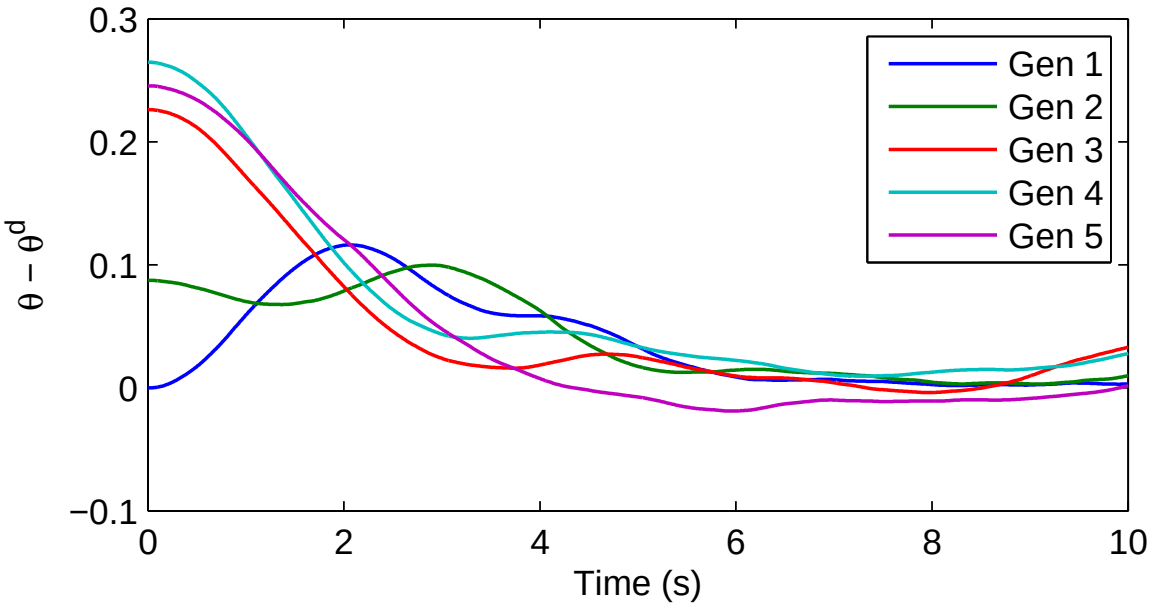
- Update every 0.1 seconds



- Update every 0.5 seconds



- Update every 2 seconds



- On my desktop (Core i7, 4-core, 3.4Ghz), $H = 2$ seconds, updating every 0.1 seconds takes 0.086 seconds per optimization (faster than real time)

- But admittedly, this is still a lot of computation for a relatively simple task like this

- MPC finds the most application in control or planning tasks where there is significantly longer time constants (e.g., the scheduling of generation and storage you will work on in the homework)

- Speed-up #3: faster optimization
- For MPC, using "standard form" solvers over YALMIP can be a big win
- Some customization can also pay off here (e.g., initializing subsequent optimization iterations with solutions from previous ones)
- Some recent research in code generation for small custom optimization solvers

Aside: Learning dynamical systems

- All control discussion thus far assumes that we know the dynamical system

$$x_{t+1} = Ax_t + Bu_t$$

- What if we don't know the A and B matrices?

- If we can observe a colleciton of data points generated by the system, we can use machine learning to estimate the model

  - Typically will want to use stochastic control methods for such models, because learning algorithms may not make perfect predictions

- In some detail: suppose we can observe a collection of data points from the system

$$x_1, u_1, x_2, u_2, \ldots, x_{T-1}, u_{T-1}, x_T$$

- Then we want to find A and B that solve optimization problem

$$\underset{A,B}{\text{minimize}} \sum_{t=1}^{T-1} \|x_{t+1} - Ax_t - Bu_t\|_2^2$$

  (or use a different loss function from ML lectures)

- This is a linear least-squares problem (actually n least squares problems, since x_t is multivariate)

- If the true system is linear, this works well, but if the true system is not quite linear, the distribution of states/control in the training set becomes a big issue