MATLAB EXPO 2015
KOREA
2015년 5월 21일 목요일
인터컨티넨탈 코엑스, 서울
Simulink을 이용한 다양한 제어 시스템 설계 방법

김종현 차장
Senior Application Engineer
MathWorks Korea
**Challenge:**

- Manual tuning of control systems is time-consuming and often suboptimal.
- Not easy to make a simulation model for control design.

**Solution:**

- Various control design features provided in MATLAB® & Simulink® enable **fast, easy, and rigorous** tuning
Agenda

- Control Design using Robot Arm example
  - Data-Driven Modeling (System Identification)
  - Automatic PID Tuning (PID Tuner)
  - Response Optimization
  - Control System Tuner
  - Online Parameter Estimation

- Extras using Inverted Pendulum
  - LQG & LQR Controller Design
  - Gain Scheduling Controller Design
Robot Arm Example

- Six joints
  - DC motor at each joint
  - PID controller for each motor
Robot Arm Modeling

- SimMechanics model of the robot arm can be generated directly from a CAD tool...
DC Motor Modeling

- To make more accurate simulation model, the effect of actuator dynamics should be considered.
Modeling Approach for well-known Actuator
First Principle Approach

First Principles Equations

Physical Components Libraries

Advanced Components Libraries

Take advantage of the flexibility provided by Simscape and the advanced physical component libraries in Simulink to create models for all kinds of actuators and sensors

Modeling Approach for unknown Actuator
Data-Driven Approach : System Identification

Input
Voltage

Output
Angle or Angular Velocity

System
Modeling Approach for unknown Actuator Data-Driven Approach: System Identification

\[ x(t+1) = Fx(t) + Gu(t) \]
\[ y = Hx(t) \]

Input:Voltage
Output:Angle or Angular Velocity

Model
\[ u, y: \text{measured time or frequency domain signals} \]
The System and the Model

- Measured input
- System
- Model
- +
- -
- error
- Minimize
Adding a controller

- Modeling both, plant and controller in a single environment allows us to better understand and optimize the performance of the entire system.
We can use very powerful linear control techniques to design our controller...

Use *Simulink Control Design* and the *Control System Toolbox* to linearize the plant and design the controller against requirements in the time and frequency domain.
Or we can fine tune our controller gains by optimizing the system response...

Use **Simulink Design Optimization** to set graphical constraints on relevant signals and optimize the controller gains against the nonlinear system response.
Use *Simulink Control Design* and the *Control System Toolbox* to linearize the plant and design the controller against requirements in the time and frequency domain.
Let’s apply PID Tuner to Robot Arm….

- Six joints
  - DC motor at each joint
  - PID controller for each motor
Tuning Individual PID Loops

Quick and simple workflow for tuning single PID:
Tuning Multiple Loops

*Standard approach*: Tune each loop individually, then iterate until overall response is satisfactory.

**Challenges:**

- Loop interactions complicate tuning
- Robustness is harder to analyze
- May not yield best overall design
- Time consuming
- Requires expertise
Solution for Systems with Complex Architecture

Control System Tuner

Multiple Loop

Multi-Input Multi-Output
Solution for Systems with Complex Architecture
Control System Tuner

- **Effective, flexible, and easy to use!**
  - Tuning for multiple plant models or operating conditions
  - Soft vs. hard requirements
  - Band-limited requirements
  - Custom parameterization of tuned blocks
Tune against rich collection of requirements

Time Response

Loop Shaping

Performance

Robustness
Tune against rich collection of requirements

- **Gain Constraints**
  - Reduce cross-coupling
  - Reduce peak sensitivity
  - Limiting gain to reduce loop interaction \( S = I - T \)

- **Disturbance Rejection**
  - Push gain down at low frequency
  - Disturbance rejection as a gain constraint
Tune against rich collection of requirements

\[ J = \lim_{T \to \infty} \mathbb{E} \left( \frac{1}{T} \int_{0}^{T} (x^T Q x + u^T R u) \, dt \right) \]
And we need to consider how to detect fault when motor is broken.

- Use **Online Parameter Estimation**
  - to identify changes in system dynamics as they happen
  - to adapt to operating conditions unknown at design time

- Other Applications: Adaptive Control
Online Parameter Estimation Capabilities

Current Features in System Identification Toolbox

- For Linear System
  - ARX, ARMAX (2014a)
  - AR, ARMA, OE, BJ (R2014b)

- For Non-linear System
  - Models linear in estimated parameters, can be nonlinear in regressors (R2014a)
Easy Deployment: Code Generation

- All blocks support C, C++ code, PLC structured text generation

- Demos
  - DC motor fault detection algorithm running on Arduino
Further investigation.....

- How about
  - LQG/LQR controller?
  - Gain scheduling controller?

- Demos
  - Balancing a pendulum on a cart while tracking the cart position
  - Input
    - Force on the cart
  - Outputs
    - Angular velocity of the pendulum ($\dot{\theta}$),
    - Velocity of the cart ($\dot{x}$)
LQR Controller Design

- Can get optimal K using the **linearized plant model** and **lqr** function.

For a continuous-time linear system, defined on $t \in [t_0, t_1]$, described by

$$\dot{x} = Ax + Bu$$

with a quadratic cost function defined as

$$J = \frac{1}{2} x^T(t_1)F(t_1) x(t_1) + \int_{t_0}^{t_1} (x^T Q x + u^T R u) \, dt$$

the feedback control law that minimizes the value of the cost is

$$u = -Kx$$

where $K$ is given by

$$K = R^{-1}B^T P(t)$$

and $P$ is found by solving the continuous time Riccati differential equation.

$$A^T P(t) + P(t) A - P(t)B R^{-1} B^T P(t) + Q = -\dot{P}(t)$$

with the boundary condition

$$P(t_1) = F(t_1).$$
LQR Controller Design

- Can get optimal K using the linearized plant model and `lqr` function.

1. Set operating point
2. Set parameters to make quadratic cost function
3. Calculate K to minimize quadratic cost
LQG Controller Design

- For control system having incomplete state information, easy-to-use `kalman` function enable to design optimal LQG controller.
LQG Controller Design

- For control system having incomplete state information, easy-to-use Kalman function enables the design of optimal LQG controller.

1. Set operating point

2. Calculate Kalman gain to estimate

3. Set parameters to make quadratic cost function

4. Calculate K to minimize quadratic cost
Gain Scheduling Controller Design

- Use Parallel Computing to speed up calculation to get K matrices at multiple operating points.
Gain Scheduling Controller Design

- For control systems with incomplete state information, it is easy to design Kalman functions to enable the design of optimal LQG controllers.

1. Set operating point

```matlab
for i = 1:length(angle_ranges)
    theta_op = opspec.States(3);
    theta_op.Known = 1;
    theta_op.x = angle_ranges(i);
    op = findop('pendcart_GS',opspec);

    % Find a linearized system for the operating point.
    linsys = linearize('pendcart_GS',iopts,op);

    % LQR Weighting Matrix
    w_x  = 1000;
    w_th = 1000;
    Q = [w_x 0 0 0;
         0 1 0 0;
         0 0 w_th 0;
         0 0 0 1];
    R = 1;

    % Compute LQR Controller
    K(i,:)= lqr(linsys.A,linsys.B,Q,R);
end
```
Gain Scheduling Controller Tuning Using Parallel Computing

- Steps to compare simulation methods

1. Open pool of MATLAB sessions
   ```
   >> matlabpool(2)
   ```

2. Generate parameter sets
   ```
   Angle_operating_points = linspace(0, 7*pi/16, numCtrls);
   ```

3. Run simulations **serially**
   ```
   for i = 1:length(angle_range)
   ~~~~~~~
   end
   ```

4. Run simulations in **parallel**
   ```
   parfor i = 1:length(angle_range)
   ~~~~~~~
   end
   ```
Summary

Quick and simple workflows for tuning control systems in MATLAB & Simulink:

- PID Tuner for PID loops
- Control System Tuner for everything else
- MATLAB functions from Control System Toolbox and Robust Control Toolbox

Benefits:

- Avoids time-consuming hand tuning
- Takes guesswork out of tuning
- Uses rigorous, state-of-the-art methodology
- Provides solid initial designs for further nonlinear optimization
Thank You...