**ChBE 4412 Design Project Phase 2**

**Due *end of semester***

This case study was contributed by Dr. Tom Badgwell:



**Resources from Phase 1**

* Videos
	+ Overview
	+ Refinery Process Control
	+ Fired Heater Control Problem
	+ Fired Heater Manual Operation
* “Fired Heater MPC Simulation Tutorial”
* “Model Predictive Control in Practice” by Badgwell and Qin

**Resources for Phase 2**

* Videos
	+ Fired Heater PID Control Solution
	+ Fired Heater MPC Control Solution
	+ Fired Heather MPC Control Solution Demo
	+ The MPC Algorithm

|  |  |  |
| --- | --- | --- |
| **Quantity** | **Abbreviation** | **Units** |
| *Actuators* |  |  |
| Flow Controller 1 | FC1 | barrels per hour (BPH) |
| Flow Controller 2 | FC2 | barrels per hour (BPH) |
| Fuel Gas Flow | FG | thousand standard cubic feet per hour (MSCFH) |
| *Sensors* |  |  |
| Inlet temperature 1 | TI1 | °F |
| Inlet temperature 2 | TI2 | °F |
| Tubeskin temperature 1 | TS1 | °F |
| Tubeskin temperature 2 | TS2 | °F |
| Outlet temperature 1 | TO1 | °F |
| Outlet temperature 2 | TO2 | °F |
| Temperature of combined outlet | TO | °F |
| Total flow rate in output | FO | barrels per hour (BPH) |

**Learning Objectives**

* Compare the design process for multiple single-input single-output controllers to the design process for a single multivariable controller
* Understand how model predictive control utilizes weightings to prioritize objectives
* Observe how model predictive control enforces constraints, unlike the model-based LQG controller

**Assignment**

The state-space model of the fired heater is located in Software/mpcsim: the MATLAB command load('HeaterModelCont.mat') will read in *A, B, C,* and *D* for the system of five inputs and five outputs. Note that this model is in deviation variables. You can use the State Space block in Simulink to simulate this system. Given the difficulty of constructing a model based on balance equations (topic of Phase 1), we will use an empirical model for Phase 2.

Consider the scenario in which

1. The system is initially operating at steady state under closed-loop control with a temperature setpoint of 750°F, which corresponds to inputs of FC1 = FC2=100 BPH, FG = 95 MSCFH, and TI1 = TI2 = 540°F.
2. The temperature setpoint for TO is suddenly changed from 750 to 755°F.
3. After reaching the new steady state, the setpoint for FO is suddenly changed from 200 to 220 BPH.
4. After reaching the new steady state, the setpoint for FO is suddenly changed from 220 to 250 BPH.

Part A:

1. Calculate the steady-state gain matrix *K* and the Bristol relative gain array L for the 5x5 system of inputs and outputs.
2. In words, explain how the outputs are dependent on the inputs. Based on these matrices, which MVs would you pair with each CV?
3. Design a SISO PID-type controller that manipulates fuel gas to control the outlet temperature to a setpoint. Explain your design methodology and show your calculations.
4. Design a SISO PID-type control that adjusts the total flow (FC1+FC2) to control the outlet flow. Explain your design methodology and show your calculations.
5. How would you ensure that the tubeskins do not exceed their constraint temperature? Incorporate elements into your control strategy for this important safety priority.
6. How would you balance the passes to keep DT close to zero? Incorporate elements into your control strategy to reduce the risk of coking and clogging associated with large DT.
7. In Simulink, run your controller for the scenario described above. Plot the MVs and the CVs. Describe the characteristics of the response, including whether or not the controller performs well.

Part B: Linear quadratic Gaussian (LQG) control

The linear quadratic Gaussian (LQG) controller is a model-based multivariable feedback controller that calculates the adjustments to all MVs, using the tracking error for all CVs.

Type help lqg in MATLAB to get some description of the inputs and outputs for this function.

Here is some description taken directly from the MATLAB documentation:



In order to calculate the optimal linear controller (which is another state-space system), you must specify the weighting matrices Qxu and Qwv. (You will use these same matrices for Part C on MPC.) The idea is to minimize the cost function *J*, so that the state deviations are small, but the input action *u* is also small. The weighting matrix Qxu is what helps you balance these two objectives, depending on your preference to have low tracking error versus to keep the input action small. The matrix Qwv represents how large is the noise in the process and the sensor, based on the expected variance of each quantity.

Here is some code to help you calculate the LQG controller:

% Calculate the LQG controller

B\_MV = B(:,1:3); % only use the 3 MV's as inputs here, not the DV's

D\_MV = D(:,1:3);

n = size(A,1); % state dimension

m = size(B\_MV,2); % input dimension (MVs only)

p = size(C,1); % output dimension

sys = ss(A,B\_MV,C,D\_MV);

% Weightings on tracking error and on input (MV) for LQG and MPC

Qc = diag([100, 1, .001, .001, .001]); % Weighting on state

Rc = diag([.001, .001, .01]); % Weighting on input

% The Qc for MPC is for output control, so the Q weight for the LQG calculation is

Q = C'\*Qc\*C;

% Weights for the state estimation part

Qe = eye(n);

Re = diag([0.0001, 0.0001, 0.0001, 0.0001, 0.0001]);

Qwv = [Qe zeros(n,p); zeros(p,n) Re];

Qxu = [Q zeros(n,m); zeros(m,n) Rc];

% Final step is to call the LQG function, which returns a structure

Klqg = lqg(sys,Qxu,Qwv);

1. In Simulink, run your controller for the scenario described above. Plot the MVs and the CVs. Describe the characteristics of the response, including whether or not the controller performs well.

Part C: Model Predictive Control (MPC)

Model predictive control is similar to LQG, in that the optimal action is calculated based on a model of the process, to minimize the user-specified cost function *J*. However, with MPC we can also incorporate constraints, which are not a part of the LQG approach.

Using the graphical user interface from Part 1, turn on the closed-loop function with the third button, having the circular arrow. You can change the setpoint using the CV menu.

1. Simulate the system according to the scenario described above. Plot the MVs and the CVs. (Ok to take a screen shot.) Describe the characteristics of the response, including whether or not the controller performs well.
2. Flip the priorities for TO and FO by changing the weights in the Q’s. Document your work and interpret the plots.
3. Stop the pass flows from moving for a TO setpoint change by increasing their Rc weights. Document your work and interpret the plots.
4. Slow down the TO and FO setpoint responses by increasing the Sc weights on the three MVs. (Note: the Sc weight is for the steady-state behavior, see video.) Document your work and interpret the plots.