EBO780

Assignment 5
Theme 5: Introduction to Unconstrained MPC

MATLAB implementation of unconstrained linear model predictive control

The purpose of this assignment is to familiarize the student with the practical working of model predictive control (MPC) by implementing a simple unconstrained linear MPC without using the MATLAB MPC toolbox.

For this assignment you need to code a basic linear unconstrained MPC in MATLAB and Simulink using only the control system toolbox and the optimisation toolbox. Any use of the MPC toolbox will result in a mark of zero. The linear model for your model predictive control is given by:

$$G(s) = \begin{bmatrix} \frac{12.8}{16.7s+1}e^{-1s} & \frac{-18.9}{21s+1}e^{-3s} \\ \frac{6.6}{10.9s+1}e^{-7s} & \frac{-19.4}{14.4s+1}e^{-3s} \end{bmatrix}$$

1. Define a discrete-time linear unconstrained MPC by giving the objective function, optimization problem as well as all the variables/parameters involved. [10]

2. Convert the continuous-time model to a discrete-time model with sampling time of 1 second and document the model. (Use *c2d* in Matlab.) [5]

3. Convert the discrete-time model to a state-space representation and document the model. (Use *ss* in Matlab.) [5]

4. Create a discrete-time model that outputs the states of the model, by using the identity matrix as the *C*-matrix and document. The time delays can be expressed as explicit states. [5]

5. Document the prediction horizon that will lead to 99% of steady-state for the slowest model to a unit step. (You can use *step* in Matlab to get a step-response of the LTI model.) [5]

6. Document the control horizon that you have chosen. [5]

7. Code the Matlab function *ObjFunc* that calculates the objective value that will be optimized by *fminunc*. Your MPC controller must implement blocking to allow for a smooth closed-loop MV trajectory. [10]

8. Show the code to simulate the model predictions in your objective function using the *A, B, C* and *D* matrices of the discrete time state-space model explicitly and not *lsim*. [5]

9. Code a MATLAB Simulink s-function m-file to call *fminunc* on your function *ObjFunc* and connect it to the original continuous-time model $G(s)$. Remember to specify the sampling time of your s-function m-file to be 1 second. [10]

10. Run the Simulink model for 200 seconds and document the plot for the controlled variables and the manipulated variables. Use the fixed time step solver (*ode4*) with fundamental time-step of 1 second. This is set under the *Simulation->Configuration Parameters* menu. On the main screen under *Solver options* choose *Type* as *Fixed-step* and *Solver* as *ode4* (Runge-Kutta) and set *fixed-step size (fundamental sample time)* to 1 second. Start with the initial condition of all the states at 0 and then make a setpoint changes from [0;0] to [5;-5] at time 20 seconds and then to [-5;5] at time 100 seconds. [10]

11. Attach your MATLAB code to the end of the report. [-10 if not present]

12. Assume you have setup a MPC controller for a 2-by-2 system with $N_P = 20$ and $N_C = 3$. Determine what Q_1 and Q_2 should be in order for a 1% deviation of y_1 for the steady-state of 150 to contribute 5 times as much as a 7% deviation of y_2 from its steady-state value of 320 to the objective function. Further, determine what R_1 and R_2 should be that a 10% change relative to the range of u_1 of $0 - 400$ contribute the same to the objective function as a 20% change relative to the range of u_2 of $25 - 68$. Lastly, the contribution of y_1 and y_2 should be 60 times larger for the stated deviations than the contributions of u_1 and u_2, after the objective function was corrected for prediction and control horizon. Document your calculations in you report. [5]

Total: 75 marks