Model Predictive Control for Temperature Regulation

The purpose of this lab is to reinforce the concepts taught in class about dynamic process models, estimation, and model predictive control. It involves real-time sensing and calculations on an Arduino device that is a portable pocket-sized lab experiment.

Problem Statement

Model Predictive Control with an Empirical Model

1. Perform a doublet test on the system, varying the control output in manual mode. See the exercise at http://apmonitor.com/do/index.php/Main/DataSimulation to download sample data.
2. From the manual-mode test calculate linear, first-order constants (τ_p, K_p) fitting the data to the equation $\tau_p \frac{dx}{dt} = -x + K_p u$.
3. Develop a model predictive controller based on the linear, first-order model.
4. Perform set point changes and introduce disturbances (blow on the thermistor) to test the performance of the controller. Comment on the performance of the controller using the empirically derived constants.
5. Tune the controller to improve performance.

Model Predictive Control with a First-Principles / Hybrid Model

1. Derive the form of a first principles model for the relationship between input voltage and output temperature. There is no need to directly measure all parameters in the model; engineering judgement is sufficient.
2. Simulate the first principles model and compare the results to the data that were collected during the doublet test. Adjust the parameters in your model to align the model and measured values.

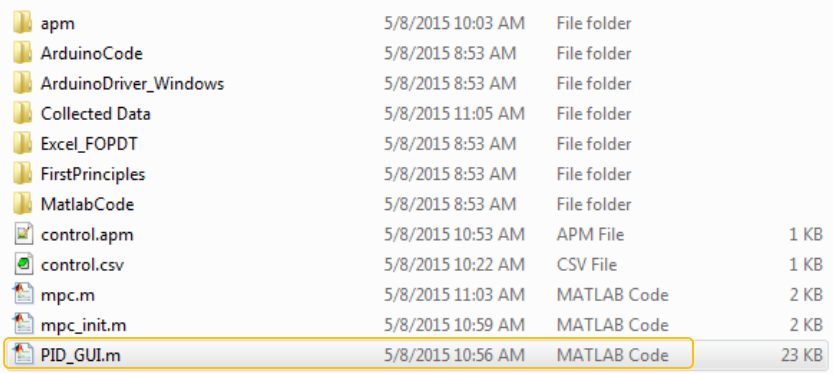
3a. For linear MPC: Linearize the adjusted first principles model and compare it to the empirical model. Comment on the similarities or differences between the two.

3b. For nonlinear MPC: Create a model predictive controller to adjust transistor voltage to regulate temperaute.

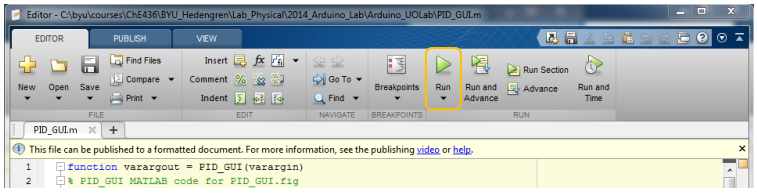
4. Perform set point changes and introduce disturbances (blow on the thermistor) to test the performance of the controller. Comment on the performance of the controller using the empirically derived constants.
5. Tune the controller to improve performance.

Setup for the Temperature Control Device

1. Plug in power supply to electrical outlet and USB connection to UO Lab computer
2. Download required files from course website and extract files from zipped archive
3. Open PID_GUI.m from extracted folder (not zipped folder)



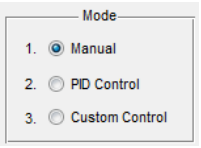
4. Click the green Run button



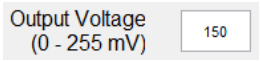
Obtain a Dynamic Model from Step Test Data

1. Click the green **Start** button.

2. Once the module has initialized, select **Manual** mode.



3. An input box will appear to allow changes to the input voltage.



4. Input manual values of output voltage to implement either a step, doublet, or PRBS input signal to the Arduino device. The *Enter* key is required to implement a change. The temperature may take a couple minutes to reach a new steady state value.

5. When the test is complete, select **Stop**.

6. Retrieve data from the Folder **Collected Data**. If multiple tests were performed, the data files are named according to the test time stamp.

Determine a Linear, First-Order Model

Fit data to a FOPDT model using Excel, MATLAB, Python, or another analysis tool. There is an Excel template in the folder **Excel_FOPDT**. Values from the generated data should be copied into the appropriate locations on the Excel worksheet as shown below.

Insert time	These Input	Values Measured
0.01667	70	3.9885658
0.16667	70	4.0013333
0.31667	70	3.9986239
0.46667	70	3.9908719
0.61667	70	3.9775105
0.76667	70	3.9935262

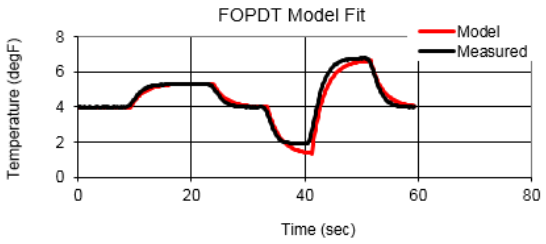
While it is not necessary to modify the columns that calculate the model mismatch, it may be necessary to fill down the equations in these columns if the number of data points exceeds the template default.

Don't change these columns, only copy down to match number of measurements					
Model	Model Slope	Model Intercept	Model with Delay	abs(error)	error^2
3.9885658	0	3.9885658	3.9885658	0	0
3.9885658	0	3.9885658	3.9885658	0.0127675	0.000163009
3.9885658	0	3.9885658	3.9885658	0.0100581	0.000101165
3.9885658	0	3.9885658	3.9885658	0.0023061	5.3181E-06

Values of K_p and τ_p can be obtained by either manually changing the values in the parameters section or using Excel Solver to find the values that minimize either the Sum of Squared Errors or else the Sum of Absolute Errors.

Model Parameters	
Kp (Gain)	0.135127622
tau (Time Constant)	2.229643365
theta (Time Delay)	2
Doesn't seem to change with solver	

Minimize Either of These	
Sum of Squared Errors	157.5540833
Sum of Absolute Errors	154.7466485



PID Controller Tuning

Once the linear, first order model (K_p and τ_p) is determined, use tuning correlations to select acceptable starting values for the PID controller (K_c, τ_I, and τ_D).

1. Click the green Start button.
2. Select Control mode.
3. Enter **Proportional (P)**, **Integral (I)**, and **Derivative (D)** terms for the PID controller.
    a. Proportional (P) = K_c

    b. Integral (I) = $\frac{K_c}{\tau_I}$
    c. Derivative (D) = $K_c\tau_D$
4. Tune controller to achieve improved performance.
5. Select **Stop** when the test is complete.
6. Retrieve the saved data file from the **Collected Data** folder.

MPC Tuning (Custom Control Option)

Once the linear, first order model (K_p and τ_p) is determined, implement the model in **control.apm** and tune the controller in **mpc_init.m** to achieve acceptable performance.

1. Click the green Start button.
2. Select Control mode.
3. Enter controller setpoint.
4. Tune controller to achieve improved performance.
    a. Common MV tuning parameters
        i. COST = (+) minimize MV, (-) maximize MV
        ii. DCOST = penalty for MV movement
        iii. DMAX = maximum that MV can move each cycle
        iv. FSTATUS = feedback status with 1=measured, 0=off
        v. LOWER = lower MV bound
        vi. MV_TYPE = MV type with 0=zero-order hold, 1=linear interpolation
        vii. STATUS = turn on (1) or off (0) MV
        viii. UPPER = upper MV bound
    b. Common CV tuning parameters
        i. COST = (+) minimize MV, (-) maximize MV
        ii. CV_TYPE = CV type with 1=ℓ_1-norm, 2=squared error
        iii. FSTATUS = feedback status with 1=measured, 0=off
        iv. SP = setpoint with CV_TYPE = 2
        v. SPLO = lower setpoint with CV_TYPE = 1
        vi. SPHI = upper setpoint with CV_TYPE = 1
        vii. STATUS = turn on (1) or off (0) MV
        viii. TR_INIT = trajectory type, 0=dead-band, 1,2=trajectory
        ix. TR_OPEN = opening at initial point of traj compared to end
5. Select **Stop** when the test is complete.
6. Retrieve the saved data file from the **Collected Data** folder.