This is a sample model demonstrating how to use a PID controller. In this demonstration a PID controller is used to control the temperature of a water tank to 80F as the environment around the tank cycles between 62 and 81F and an additional heat loss of 25 BTU/hr is pulled from the tank.

MODEL DESCRIPTION

The tank (FLOW.LUMP.5) is connected to a fluid loop that has a constant mass flow rate modeled as an MFRSET (FLOW.PATH.1). Downstream of the MRFSET is a tee (FLOW.LUMP.3) which splits flow into two parallel paths to the water tank, a bypass line (FLOW.PATH.5) and a line which goes through a heat exchanger (FLOW.LUMP.4) which heats the water to 120F. A control valve (FLOW.PATH.3) located between the tee and the heat exchanger is used to control how much fluid goes through the heat exchanger. A PID controller (Logic Object 2) is used on this control valve. Based on the temperature of the tank the PID controller adjusts the valve opening to increase or decrease the flow through the heat exchanger, thus maintaining the tank at the desired temperature of 80F.

Cycling of Ambient Temperature

The boundary node (main.2), which represents the ambient air, varies over a 24 hour period. The temperature profile is defined using an XY table of air temperature as a function of time. Logic in the Logic Object Manager (Logic Object 1) is to interpolate the XY table in Variables 0 logic block.

External Convection on the Tank

The exterior surface of the tank convects to ambient air with a surface area of 35 ft2. This is modeled using a convection conductor between the tank wall and the air.

Convection Internal to the Tank

Inside the tank, the heat transfer between the fluid and the tank wall is modeled using a tie (FLOW.TIE.1).

Water Tank

The water tank is modeled as a Fluint tank (FLOW.LUMP.5). The heat loss of 25 BTU/hr is applied to the tank as a negative QL term. Logic Object 6 calculates the value of SUMOFDT which is used as the OBJECT for the optimization case.

Heat Exchanger

The heat exchanger is modeled as a perfect heat exchanger by add a call to HTRLMP on FLOW.LUMP.4. This call is defined in Logic Object 4. The call to heater lump will add or subtract heat from the lump as required to maintain it at the initial temperature of 120F.

Control Valve and PID Controller

FLOW.PATH.3 is modeled as a control valve with PID control on the stem position. Logic Object 3 interpolates the value of CTLCRV based on the valves stem position as defined in a table of CTLCRV as a function of stem position. CTLCRV is then used to calculate the k-factor loss on the valve, effectively opening and closing the control valve.

Logic Object 2 provides an interface for setting up the PID control. The inputs on the form are used to set up the appropriate calls in SINDA/FLUINT for the PID solution.

The form is set up to model a discrete sampling rate. In most applications a PID controller samples the process variable (what is being controlled) using a specific time interval. For this example the PIDS routine will be used with a sampling interval of 10 seconds. When a discrete sampling rate is defined, a submodel is created specifically for the PID and OUTPUT CALLS logic of that submodel is used for the sampling. Due to the nature of the GLOBAL submodel, the desired OUTPUT interval on the Output tab of the Case Set is placed in the MAIN submodel.

CASE DEVELOPMENT

Solver Case

The Solver case is used to find the best values for the proportional, integral, and differential gains. In the operations block, an initial steady state run is made to provide a starting point for subsequent transient runs which will be invoked by the Solver. An optional call to DSCANLH can be used to run a Latin hypercube scan on the three design variables to find a good starting point for the Solver.

The design variables are set as exponent terms for the three PID gain terms, for example GDTC = 10**GDTCPow where GDTC is the differential gain and GCTCPow is the design variable used in the Solver. Since the performance of the PID controller is very sensitive to the gain values, a small change in one of the design variables can result in a very large change in the value of the object. By using an exponent term for the design variables instead of the actual gains, we are effectively minimizing the change in the value of the object.

For this example there are no constraints other than the side constraints applied to the design variables.

The goal is to minimize the temperature difference between the water tank and the set point over the course of a 24 hour period. This is achieved by setting the OBJECT as the register SUMOFDT (calculated in Logic Object 6). SUMOFDT is calculated as the 24 hour summation of the square of the temperature difference.

A note on using DSCANLH: this option may produce better results by exploring the design space to seek the best values for the design variables with which to start the Solver. This reduces the chance of finding a local minimum value of the OBJECT based on the starting values of the design variables. In this example if the design space is not scanned, the Solver does find a local minimum which works to control the water tank. Currently the model has the DSCANLH option commented out simply because it results

in a significantly longer run time. The results without the design space pre-scan are adequate for the purposes of this demonstration, however in a real application running a scan of the design space is highly recommended.

Transient Case

The transient case is a 24 hour transient set up to verify the values the Solver found for the various design variables. The case is set up with Case-level register overrides for GDTCpow, GITCpow, and GPTCpow. After running the solver case, place the resulting values of the design variables in these register overrides and run the case.