Date: _____ Name: _____

DEMONSTRATION EXERCISE 21
CHARACTERISTICS OF MODEL-BASED CONTROL SYSTEMS

OBJECTIVE: To demonstrate the behavior of several forms of model-based control systems, in particular, the Smith Predictor, a simple form of Internal Model Control and Dahlin's algorithm.

PREREQUISITE: Completion of Exercise

9 Tuning from Open Loop Tests

BACKGROUND: Model-based control uses an explicit model of the process, either as a portion of the control algorithm or to derive parameters for the control algorithm. A frequent application of model-based control is to control processes with a long dead time, relative to the dominant time constant. With these applications, a model-based control can usually exhibit an improved performance over a PID algorithm, particularly for set point changes.

There are several forms of model-based control techniques. Some are based on approximating the process with a simple transfer function model, some incorporate explicit physical phenomena in the model, and some techniques capture the process model as a series of data points on the pulse response curve. Only the transfer function model approach is covered in this laboratory exercise. However, both the continuous form and the sampled data form of the transfer function are represented.

Although real processes operate continuously in real-time, some physical property sensors and/or converters provide information only intermittently. Furthermore, digital algorithms are processed only periodically, hence provide new results only at discrete intervals of time. These two factors give rise to sampled data control systems, one of the types of model-based control techniques explored here (Dahlin's algorithm).

A challenge often faced by control systems designers is that of processes with long dead time. This has motivated the development of a technique known as "control algorithm synthesis". This is a mathematical technique which, based upon minimal data input from the user, calculates both the form and numerical values for parameters of the control algorithm. Control algorithm synthesis can be applied to either continuous control (Internal Model Control) or sampled data control systems (Dahlin's algorithm).

The requirements of the user are:

An estimate of three parameters representing a simplified model of the process:

(1) steady state gain,
(2) first order lag time constant and
(3) dead time;

The time period at which the control algorithm is processed (for sampled data control systems)

A parameter representing the desired closed loop response to set point change. The desired response can be approximated as a dead time (equal to the process dead time) followed by a first order lag. The single tuning parameter which can be set by the user, then, is the time constant of the desired first order lag response.

Although the algorithm synthesis technique utilizes sophisticated and somewhat tedious mathematics, this tedium can be pre-programmed and made transparent to the user. This laboratory exercise demonstrates the use of a sampled data algorithm which is calculated using a synthesis technique.

> **A matter of terminology:** In all other laboratory exercises, and in the drop down menu under Process, we refer to the process simulation, which is constructed by BUILDER, saved as a file, then selected and read into PC-ControLAB as the "process model." In this laboratory exercise, the term "process model" has another meaning. Hence, we will refer to the process simulation constructed by BUILDER as the "actual process" and our approximation of it by tests made in PC-ControLAB as the "process model."

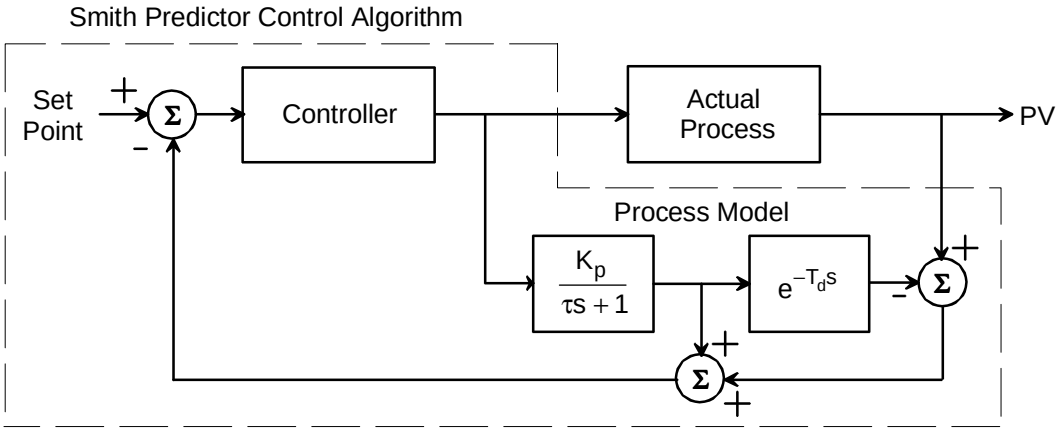
1. RUNNING THE PROGRAM

Start **Windows.**

Run **PC-ControLAB.**

NOTE: Each of the following sections explores a particular one of the model-based control techniques.

2. SMITH PREDICTOR CONTROLLER



2.1 Using a "perfect" process model

Select **Control | Select Strategy | Smith Predictor**.

Select **Process | Select Model**. Highlight "FOLPDT.mdl" (First order lag plus dead time) process) and press **Open**.

> *We will start out with a simple process where our process model will be an exact representation of the actual process. Then we will progress to a more complex process, where our process model will be only an approximation of the actual process.*

The first step is to determine data which must be input into the setup portion of the control algorithm.

Put the controller in Manual and increase the controller output by 10% (of full scale).

Observe the process response. Characterize this response by a first order lag plus dead time model. This requires three parameter values which may be determined from the observed open loop response. Record these values:

Process Gain	K_p	1.5 .
First-order-lag	τ	5.0 mins .
Dead time (minutes)	T_d	6.0 mins .

When the process model parameters have been determined, return the controller output to its original value.

Press **TUNE** then select the **Setup** tab. This permits entry of the values of the simplified process model into the controller set up.

If the process model is an exact representation of the process, then the structure of the Smith Predictor controller causes the apparent loop to be that of a controller with the feedback loop closed around the first order lag only. It appears as if the dead time has been relocated to a point following the feedback control loop.

Tuning parameters for the PID portion of the Smith Predictor must now be determined. One approach is the "algorithm synthesis" approach, wherein the form of the desired response of the PV (of the apparent loop, without dead time) to a set point change is formulated as a unity gain first order lag. The only parameter which must be specified is the time constant, λ, of the desired closed loop response.

Suppose we want the close loop response to have a time constant of 3 minutes. Hence

λ = 3 minutes.

From this data, PID tuning parameters can be calculated.

Gain:	$K_C = \frac{\tau}{K_p \lambda}$	= 1.11 .
Reset:	$T_I = \tau$	= 5.0 min/rpt.
Deriv:	T_D	= 0.0 .

Press **Tune** and enter these parameter values.

Adjust the set point to match the present value of the PV, then put the controller in Auto. Increase the set point by 10% of measurement span.

You should observe the following response: The controller output jumps immediately, then begins cutting back, even though the PV has not changed, and in fact, is still below set point. (With a PID controller, the integral action would now take over and begin ramping the controller output farther in the same direction as the proportional response.) There is a pure delay in the PV by the amount of the dead time, then the PV makes an exponential rise (as a first order lag) to the new set point, with a time constant of λ.

2.2 A more realistic process model

Select **Process | Select Model**. Highlight "Generic.mdl" and press **Open**.

Since a common application of model-based control is for process with long dead time (more specifically, a large dead time to time constant ratio, say, greater than 1.0), then to demonstrate the efficacy of this control strategy, we will modify the characteristics of the actual process. Select **Process | Change Parameters**. Make the following parameter changes:

Block labeled		Change from	To
BLK 30 1-Ord Lag	Time Const T	3.0	1.0 minutes
BLK 35 1-Ord Lag	Time Const T	5.0	1.0 minutes
BLK 40 DeadTime	Dead Time Td	0.0	5.0 minutes

Save this process model for use later. Select **Process | Save Process Model As**. DO NOT SAVE THIS UNDER THE EXISTING MODEL NAME. In the dialog box, enter a new process model name. (Suggestion: Enter CLASS). Press **Save**.

Make the usual step test to determine an approximate model for the "real" process. With the controller in Manual, increase the controller output by 10%; from the resulting response, estimate the process gain, dead time and time constant. (Listed below are reasonable values. You may observe slightly different values.)

K_p = 1.5 .
τ = 2.0 .
T_d = 6.0 .

Press **TUNE** then **Setup** tab and enter these values. Then press **Clear**..

Assume that we want a 3 minute closed loop time constant. Therefore

λ = 3.0 .

Use the equations in Section 2.1 to calculate the controller tuning parameters, then enter these parameters.

Gain: K_C = 0.5 .
Reset: T_I = 2.0 .

Put the controller in Auto and change the set point by 10%.

You should observe that the response, although similar in character to that obtained when we had a perfect process model, is not as ideal as before. The controller output responds exactly as expected until the dead time expires. Then because the process model is not exact, the controller detects and error between the actual PV and its expected PV, and begins to act on this error.

After the system is in equilibrium, press **StepIncr** once to make a 5% load increase.

Observe:

Maximum deviation from set point 3.6% .

Duration of time off set point
(for instance, deviation greater than 1%): Approx 30 mins .

We will now compare the results of the last two tests (set point response and load upset response) with equivalent tests using a standard PI controller.

Put the controller in Manual.

To eliminate the Smith Predictor portion of the control algorithm (i.e., to revert to a standard PID):

Put the controller in Manual

Select **TUNE** then **Setup**.

Enter 0.0 for the estimated process gain.

Calculate PI controller tuning parameters using the traditional Ziegler-Nichols equations. (Use Table 1 in Laboratory Exercise 9, Tuning from Open Loop Tests.)

Gain: $K_C = \frac{0.9 \times 6.0}{1.5 \times 2.0}$ = 0.2

Reset: $T_I = 3.33 \times 6.0$ = 20.0 min/rpt .

Deriv: T_D = 0.0

Enter these parameters.

Make the SP equal to the PV. Change to Automatic, then make a set point change.

It quickly becomes obvious that the response is unacceptable. The Z-N calculations are not intended for use when the dead time is much longer than the time constant.

Enter the following parameters, which were determined experimentally to give an

acceptable response (you may want to improve upon these):

Gain: K_C = 0.55 .

Reset: T_I = 10.0 min/rpt .

With the controller in Auto make a 10% increase in set point.

Is this a reasonable response? Somewhat .

Press **StepIncr** to make a 5% load change. Observe:

Maximum deviation from set point 2.8% .

Duration of time off set point
(for instance, deviation greater than 1%): Approx 30 mins .

Review the performance of the Smith Predictor and standard PI for these two tests, one a set point change, the other a load upset. The observation should be that the Smith Predictor considerably outperforms the PI controller for set point changes. For load upsets, the Smith Predictor performs no better than a PI controller, and possibly worse.

One further line of exploration. (Optional) What if there is a significant error in estimating the process parameters. Repeat tests like were made above, but with various combinations of process parameters.

K_p	from	1.25 to 1.75
τ	from	1.0 to 3.0
T_d	from	5.0 to 7.0

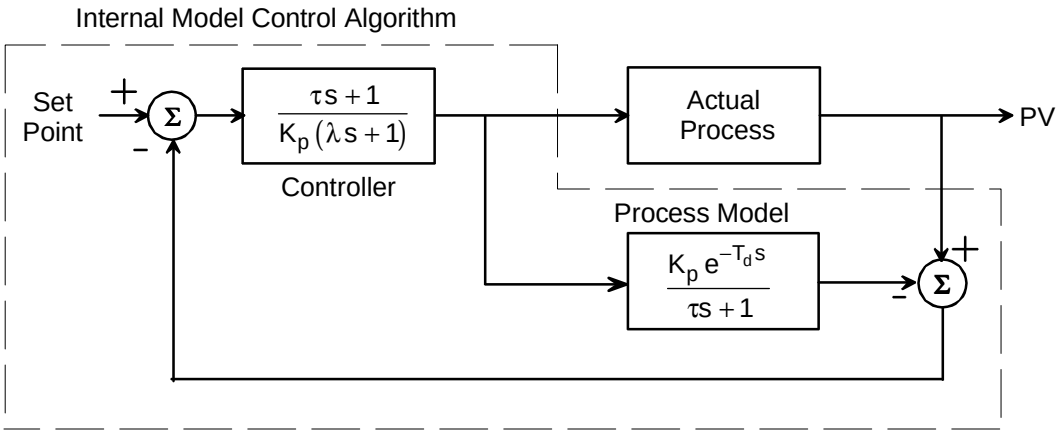
REMEMBER TO CHANGE BOTH THE PROCESS MODEL IN "SET UP," AS WELL AS TO RECALCULATE AND REENTER THE TUNING PARAMETERS.

An observation you might make is that the control technique is more sensitive to errors in the dead time estimation than to other errors. For this reason, the technique is often applied to processes where the dead time is known with a fair degree of certainty, such as where the dead time is due to the physical movement or flow rate of a variable which can be measured.

3.0 INTERNAL MODEL CONTROL

With internal model control (IMC), the objective is the same as that of the Smith Predictor controller. We want to approximate the process with a simple model (such as a first order lag plus dead time); we also want to specify the form of the closed loop PV response to a step set point change by a single tuning parameter, the time constant of an exponential rise to set point.

If the process is modeled by a first order lag plus dead time approximating model, then the controller itself becomes a lead-lag plus dead time.



Since the basic premise of the Smith Predictor and IMC are identical, we can expect the behavior to be identical, when applied to the same process.

3.1 Set Up

Select **Control | Select Control Strategy | Internal Model Control**.

Select **Process | Select Model**. Highlight "CLASS.mdl" (or whatever name you used when you saved the process in Section 2.2) and press **Open**.

The following parameters were estimated in Section 2.2:

Process Gain	K_p	=	1.5 .
First-order-lag	τ	=	2.0 mins .
Dead time	T_d	=	6.0 mins .

Press **TUNE**, **then Setup** and enter these values.

Suppose you want the PV to rise to SP with a time constant of 3.0 minutes. Hence

λ = 3.0 minutes.

Return to the **Tuning** tab and enter this single tuning value. Press **Clear**.

3.2 Closed Loop Response

Put the controller in Automatic and increase the set point by 10% of measurement span.

Is the response the same as for the Smith Predictor, using the same process and the same tuning criterion? Yes .

Since the set point behavior of the IMC is the same as that of the Smith Predictor for the same process model, we can assume that it will also be the same for load. (You may wish to confirm this by repeating the load changes made in Section 2 using the IMC controller rather than the Smith Predictor.)

The IMC controller lends itself to more complex process models. The theoretical details of IMC are covered in numerous text books.

4.0 DAHLIN'S ALGORITHM

4.1 Set Up

Dahlin's algorithm is similar in objective to the previous three model-based control techniques in that it uses an explicit process model approximation as part of the control strategy. It differs in one important aspect, however, in that it is a sampled data algorithm, rather than a continuous algorithm. Hence parameters for a discrete control algorithm must be calculated, using "z-transform" technology. This is done by the set up procedure, however, so that the only user requirements are to furnish the proper parameters, including sampling period, for the set up procedure.

Select **Control | Select Strategy | Dahlin's Algorithm**.

Select **Process | Select Model**. Choose the FOLPDT process and press **Open**.

Either repeat the process test made in Section 2.1 to determine parameters for an approximating process model, or simply reuse the data from that test:

Process Gain	K_p	1.5 .
First-order-lag	τ	5.0 minutes .
Dead time	T_d	6.0 minutes .

We will use a control sample period of 2 minutes.

Control sample period	Delta T	2.0 minutes .

At this time, we must have the dead time to be an integer multiple of sample periods; i.e., the sample period must divide exactly into the dead time. If this is not the case, then either re-estimate the dead time, or choose another sample period.

Press **TUNE** then **SetUp** and enter these values.

We will require that the closed loop response have a 3.0 minute time constant.

Closed loop time constant required:	λ	3.0 minutes .

Return to the **Tuning** and enter the single parameter tuning constant.

After completion of entry or a change to a tuning parameter or a set up parameter, the program automatically calculates coefficients for the discrete control algorithm. To view these coefficients, return to the **Setup** tab and press **Parameters**.

Record the first of the parameter values:

C_0 = 0.9840 .

Press **OK** and **Clear** to remove the Tuning dialog box.

You have now completed the controller set up.

4.2 Closed Loop Set Point Response

Put the controller in AUTO and increase the set point by 10% (of full scale).

Was any change made to the process variable in less time that the process dead time?
No .

Once the dead time expired, did the process variable respond with the requested 3 minute time constant? Yes .

Was the controller output response as predicted? Yes .

Access the Tuning Display and change λ to 1.0 minutes. (This is asking for an even faster closed loop response.)

Press **TUNE | Setup | Parameters**. Record the first parameter:

C_0 = 1.7487 .

Was there a change made to the coefficients? Yes .

Again increase the set point by 10%. We should expect that the controller output will overdrive even more, then cut back to its final value. Is this what happens?
Yes .

This has demonstrated that Dahlin's algorithm, except for using sampled data theory, utilizes the same principles as the Smith Predictor and IMC controllers. Hence the response is similar.

If desired, you can call in the "Class" process model (saved in Section 2.2) and repeat the experiments made there, using Dahlin's algorithm