Balance Perfection with Sufficiency in Modeling for Control

R. Russell Rhinehart – 2025-11-03

Question:

What is the best practice regarding the speed of execution of simulation models for control and optimization? Should they all run in real time? Or faster? What if the simulated control and supporting algorithms take longer to run than the desired real-time control interval?

Perspective:

There are several models in control algorithms. These include data reconciliation, model adjustment, inferential measurement, data imputation, forecasting, and set-point optimization. And they may be of many mathematical forms. These include mechanistic first-principles, traditional linear or nonlinear empirical regression or AI Big Data forms, and steady-state or dynamic versions of either. It is not just the central control calculation that needs to be investigated.

Also realize that no model is perfectly true to the process. Even the best attempt at rigor will still have modeling error. These sources of error include thermodynamic relations, reaction kinetic equations, equations of state, fluid flow pressure loss factors, uncertainty about impurities in the feed or those generated within the process, pump characteristics for that particular pump (which is different from its adjacent 'twin" and change in time), and the presence of inert gas from the start-up period. There are many others. And the process coefficients change in time due to heat exchanger fouling, catalyst deactivation, ambient heat losses, burner feed air properties, and more. Even if you could get, you will not be able to keep, a perfect model.

And realize that even if a model could be perfect, the data you'll get from the plant will have calibration error, noise, and numerical truncation from several transmission devices along the way from sensor to control computer. The control calculation will have a slight case of GIGO.

And even if the control calculations are in double precision, data transmission devices may only have 12-bit processors calibrated in the mid 60% of their range, as common in control signal transmission. So, there is a great loss of precision in the actual control action. And it is further corrupted by calibration errors in its transmission sequence; and possibly stiction, deadband, or other such effects in the final control element.

No matter how hard you try, models you'll use in the control algorithms will not match the real process. And a perfect control algorithm would still be embedded in a host of

imperfect data processing devices and their algorithms. A good controller uses feedback correction to compensate for process-model mismatch.

Since, in reality, the process cannot be perfectly known, any model of it will be imperfect. As hard as you might try, the controller model will not exactly match the process. This means that you would be cheating if the process model in your process simulator was the same as the model you use in the control algorithms. Don't cheat, use a simpler model in the various control algorithms. It will also run faster. But ensure that the reduced controller model reasonably captures the simulated process behavior.

In my experience people tend to choose perfection over sufficiency. But, this violates the K.I.S.S principle. The choice to seek perfection may be a leftover academic-science perspective from seeking A+ grades in school, which needs to be tempered. Or, it may be direction from your supervisor who fears the consequences to his career if his employee admits to a modeling imperfection in a presentation to upper management, which causes them to lose confidence in the work. Models cannot be perfect, and do not have to be perfect, but they do need to be adequate for their application.

Here are some examples that may help modelers cross the perfection gap: 1) Feedback on the process-model-mismatch corrects the model removing offset. 2) Supervisory optimization (RTO) uses steady-state models. 3) PID and feedforward are well accepted and effective, but are based on linear and FOPDT models. 4) Inferential models are simple relations that are regression-fit to imperfect process measurements.

Answer:

All this leads to two perspectives which mimic reality and both simplify and speed-up the control simulation:

- Balance perfection with sufficiency in your models.
- Use the semi-rigorous (but adequate) model in the process simulator, but simpler models in the several control procedures.

The bulk of my model work has been with the exploration of model-based process control and optimization. A convenience is, if the simulated process runs slower than real time, then triggering the model-based controller and associated process data analysis routines in the control loop (such as data reconciliation) also execute on the slower simulated time.

If simulation is slower than real time, monitor the duration of each simulated event, and compare it to real time to see where the simulation needs to be sped up. The speed-up solution could be faster processors, accepting longer control intervals, or reducing the computational complexity arising from seeking perfection in the models in the control and data processes.

Solve speed issues in simulation, by using control algorithms with these approaches:

- Try less frequent control triggering. A heuristic for well-behaved processes is to have 10 control actions within the time-constant of its FOPDT model. Excessive control frequency does not make control measurably better. The control interval could be much longer than the delta-t for the process simulator.
- Within the control interval, immediately after the signal is sent to the process, start associated data processing (reconciliation, model adjustment) on the existing data, do not wait for the new data. Although the simulation might work sequentially, in an application, control calculations can run in parallel with the process.
- Remove all minor and inconsequential effects (ambient losses, heat transfer through a pipe) from the control models. If some phenomenon only has a 1% impact on the controlled variable, it will be imperceptibly buried in the process noise. Modeling it will be inconsequential.
- Convert all very fast dynamic events in the control models (fluid acceleration, valve actuator, mixing in small tanks) to steady state models. Only use dynamic models for the slower elements. This is termed a pseudo-steady-state model.
- Change rigorous Differential Equation solvers to simpler ones (such as Runge-Kutta to Euler's).
- Enlarge convergence tolerance on nonlinear solvers (root finding) until you can notice an impact on the simulated variable relative to the process noise. There is no sense in seeking extreme precision, considering data uncertainty from the process.
- Reduce the order of the model. For instance, instead of modeling all 60 trays in a column, use 30 with a higher tray efficiency to give similar results. Instead of 120 gains in an MPC model, use a vector length of 30 in the dynamic matrix.