Acceleration of Dynamic Process Simulation with Hierarchical Control by Steady State Leap ⋆

Shaowen Lu *

* *State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang, 110819, China, (FAX: +862423895647; e-mail: lusw@mail.neu.edu.cn)*

Abstract: The dynamic simulation of process manufacturing system with hierarchical control of multiple time scales may have the problem of scale gap, which presents a major bottleneck to the global simulation speed. This paper presents a novel approach of steady state leap (SSL) to accelerate the system simulation of such kind. The SSL technique detects the state of the system at the beginning of each simulation cycle. If the system is at a steady state, the simulator will calculate the system state using its steady state model instead of the dynamic model. Otherwise the dynamic solver routine will be applied. SSL can also be applied in parallel to achieve further acceleration. The proposed approach is capable of speeding up the simulation to a magnitude in reverse ratio of the frequency of disturbance occurrence. The performance of SSL is validated by the experiment study on a grinding manufacturing process. The issue associated with the parallel application of SSL is also discussed.

Keywords: multi-scale simulation methods; dynamic process simulation; simulation acceleration; hierarchical control; integrated control; real-time process simulation.

1. INTRODUCTION

The process industries are referred as those productions which physically or chemically transform raw materials by mixing, separating, forming or chemical reactions. Specific examples of the process industries include biochemical enterprises, cement, chemical, minerals and metals, petrochemical/refining, pharmaceuticals, power generation, pulp and paper, and water systems (Craig (2011)). In the process industries, dynamic simulations are widely used by control engineers. In contrast to steady state simulation, dynamic simulation has the ability to simulate the time varying behavior of the processing system. This is particularly useful for control engineers as it can assist them in the analysis and testing of different control strategies which result in faster commissioning and less rework (Cox et al. (2006)). Typically, dynamic simulations are used in control strategy validation, controller pre-tuning, emergency control testing, operators training and operation optimization. Today's process control system is getting more and more complex to cope with the more tightly integrated plants. In those plants, the term "process control" has become more than loop control. The advanced control technologies, such as MPC (model predictive control) (Qin and Badgwell (2003)), RTO (real-time optimization) (Darby et al. (2011)), and plant-wide control (Larsson and Skogestad (2000), have already been widely applied. Such type of control systems can deal with both the control and optimization of the whole production process. Decomposition strategies are often applied because of the complexity of the problem. This leads to a hierarchical structure of the control system which can exploit the hierarchical nature of the process system and the model structure of the individual units.

It is extremely inefficient to simulate a complex processing system with hierarchical control structure using the traditional time-stepped, sequential simulation approach. This inefficiency stems primarily from the multi-time scale nature of the hierarchical control approach. For the local regulatory controllers, the controlled plant is the process production system which is typically modeled as a continuous system, e.g. a set of differential equations which describes how the system state changes as a function of simulation time. The evolution of the system state over time is calculated by solving the model through integration. Since the system state changes continuously over time, the regulatory controller of the plant also operates continuously [1] . Consequently, the simulation of this layer (local regulatory control layer) is mostly executed in a strictly time-stepped, sequential manner, and the length of the time-step must be sufficiently small, e.g. in milliseconds or seconds, to preserve the process dynamics. Yet, for an advanced control system, such as MPC or RTO system, it is cascadingly connected to the lower layer controllers, and therefore the controlled plant of the advanced controller can be viewed as the combination of the lower layer controllers and the corresponding controlled plant of

⋆ This work is supported by the Natural Science Foundation of China under Grant 61240012, the Chinese National Fundamental Research Program under Grant 2009CB320604 and the Fundamental Research Funds for the Central Universities under grant N120408003.

[1] Strictly speaking, controller operates discretely with very small sampling intervals, which can be considered as continuous.

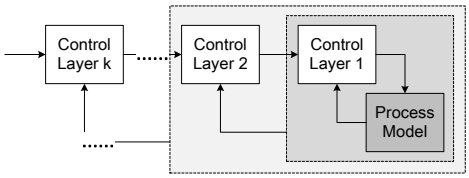

Fig. 1. Process model with cascadingly connected control layers. The inner control loop operates in the finest time scale, and outer layers cascadingly connect to inner layer.

it. With this arrangement, the advanced control system often operates in an event-driven mode on much larger time scales, e.g. hours or even days, than that of the local regulatory controllers. If the performances of the advanced control system are to be evaluated through simulation, the total simulation length must be long enough to match the time scale of the advanced control so as to make the evaluation meaningful. As a result, it will lead to impractical long simulation length if the simulation is still being run on the single uniform time scale conforming to that of the local regulatory controllers.

In view of the above problem, the objective of this work is to develop a solution of simulation acceleration. We will present an approach of steady state leap to reduce the unnecessary dynamic computation during simulation. In the following, section 2 gives the background of this topic. In section 3, the problem of simulation acceleration is specified in detail. Section 4 introduces the technique of steady state leap, and the results of experiment study are presented and discussed in section 5. Finally, section 6 concludes this work.

2. BACKGROUND

Fig. 1 shows an abstracted simulation object of the process plant and the hierarchical control. The computational bottleneck of the simulation of such a system with hierarchical control lies on the effort of solving the continuous processing plant model with fine temporal granularity. That is to say, the existence of scale gap slows down the simulation. This is because solving a high resolution entity is often considerably slower than that of the low resolution entity. Thus to maintain the consistency of the simulation, the low resolution simulation has to wait for the finish of high resolution simulation. Since each control layer operates on a different time scale, such a simulation object forms a typical multi-time scale system, which poses particular challenges to the traditional modeling and simulation techniques. There have not been universal approaches to the simulation of such kind of system, but there are attempts to tackle the issue from different aspects, and some of them are tailored to particular applications.

One of the solutions is the *multi-resolution/multi-scale modeling and simulation* (MRMS) technique. Fig. 2 illustrates the idea of MRMS. It includes multiple models and each of them is developed in a resolution/scale which matches that of the corresponding object layer. And then, these models are integrated (often by a coordinator) to form the global simulation object so that the simulation as a whole appears to have multiple resolution levels. MRMS can find its roots of concept in military applications (Davis

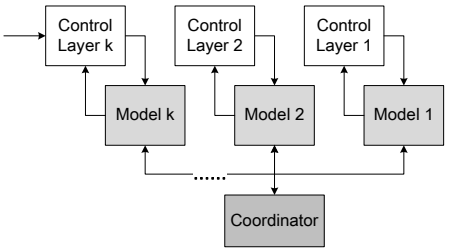

Fig. 2. Multi-resolution/multi-scale modeling approach. With this approach each control loop has its own model of the controlled plant for simulation.

(2005)) and in material (Ingram and Cameron (2004)) and chemical engineering (Maroudas (2000)) as well. But this approach may suffer the inconsistency problems (Reynolds et al. (1997)), i.e. the same entity presents inconsistently different properties at different levels caused by inadequately linking the model attributes at multiple levels.

Another approach is the use of parallel and distributed simulation (PDS) techniques (Fujimoto (2000)). However, the application of PDS is not common in continuous industrial process (Boer and Verbraeck (2006)). The reasons are multifold. First, there are no ready-to-use tools to help quickly building parallel and distributed simulation of industrial processes. Ad hoc solutions have been reported but very rare. For example a distributed simulator for training operators of a sugar plant is developed based on DCOM (Distributed Component Object Model) (Santos et al. (2008)). Second, PDS can speed up simulation when the computation load is properly decomposed and distributed onto multiple workstations running in parallel. Whereas continuous industrial processing plants are traditionally solved using the sequential modular approach (Hillestad and Hertzberg (1986)). This solving method is intrinsically sequential as the process model is decomposed into unit models which are sequentially solved in an order according to flowsheet topology. Parallelism depends very much on the topology of the flowsheet. It is not always possible to partition process flowsheet into sub-entities which can be solved in parallel.

In process modeling technology, quasi steady state approximation (QSSA) (Turanyi et al. (1993)) is commonly used for model reduction of the systems exhibiting dynamics of different time scales. This approach assumes the fast dynamics to be in quasi steady state and obtain the slow dynamics according to the corresponding quasi steady state constraints (Vora and Daoutidis (2001)) and the applicability of QSSA has been typically restricted for certain regions of initial and operating conditions. QSSA is not suitable for our problem because it is nontrivial for the analysis of higher layer controller on the model where fast dynamics corresponding to regulatory control is reduced by quasi steady state approximations.

To accelerate the simulation speed, we proposed an approach based on the following simple consideration: We define steady state as which process variables are constant over a particular period of simulation time, and define transient state otherwise. In a time-stepped simulation, system state is calculated by dynamic solver routine at the beginning of each time-step regardless of the state of the process. However, if the process is in a steady state, it

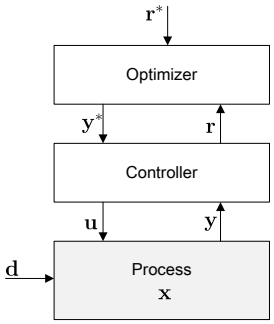

Fig. 3. Structure of the system to be simulated.

actually does not need to solve the dynamic model until the next disturbance or control input change the model state. This indicates that simulation speedup is achieved by avoiding unnecessary solving simulation model with dynamic solver during steady state. This principle will be presented in detail in the following.

3. DESCRIPTION OF PROBLEM

3.1 System Structure

In this paper, we consider that the system to be simulated includes a processing system and two layers of control. The structure of the system is illustrated in Fig. 3. To simplify the following discussion, we will call them *process*, *controller* and *optimizer* for the processing system, the lower layer control and the higher layer control respectively.

Let $\mathbf{d}$ represent the disturbances to process. Vector $\mathbf{x}$ is the state of the process. Optimizer receives $\mathbf{r}^*$, the optimization targets, and sends out a set of set-points $\mathbf{y}^*$ to controller, which implements $\mathbf{y}^*$ by controlling variables $\mathbf{u}$. Vectors $\mathbf{y}$ and $\mathbf{r}$ are the measured process variables reporting to controller and optimizer respectively. The optimizer layer and the controller layer together form a hierarchical control structure where the control loops $\mathbf{y}^* \leftrightarrows \mathbf{y}$ and $\mathbf{r}^* \leftrightarrows \mathbf{r}$ are cascadingly connected through variables $\mathbf{y}^*$ and $\mathbf{r}$.

Note that the controller layer actually includes a set of regulatory controllers. Also, for simplicity, we assume that the controller layer includes actuators and sensors so that the conversion between discrete control symbols $\mathbf{u}$ to piecewise continuous variables $\mathbf{v}$, and the conversion between process state variables $\mathbf{x}$ to sampled variables $\mathbf{y}$ will be omitted in this representation.

3.2 Objective of Simulation

Assume that the objective of simulation is to test the performance of optimizer. To be more specific, given the targets of optimizer $\mathbf{r}^*$ and process disturbances $\mathbf{d}$, the objective of the experiment is to run simulation for a length of L , and finally obtain a simulation trace which is a real valued time sequences $\mathbf{r}(t) \in \mathbb{R}$ with $t \in [0, L]$. Then, the simulation resulting sequences $\mathbf{r}(t)$ are evaluated by a function $\epsilon : \mathbb{R} \to P$, which maps $\mathbf{r}(t)$ to a set of performance indices $p \in P$.

3.3 Problem of Scale Gap

For convenience in the following discussion, define the time scale of a continuous system as the sampling interval corresponding to its Nyquist frequency, and define the time scale of a discrete system as the characteristic time between jumps. Each entity of the system shown in Fig. 3 has its corresponding time scale. Now, let h_i the time scale of the ith layer entity. In the present case, h_1 represents the time scale for the process model, h_2 for controller and h_3 for optimization. As mentioned early, for a temporal decomposed hierarchical control system the time scales of different layers are often drastically different. Therefore we assume $h_3 \gg h_2$. For a regulatory controller in the controller layer, since its objective is to stabilize the controlled local process variable and also to track given set-point, the time scale of a regulatory controller must match its controlled plant. So we let $h_1 \backsimeq h_2$ which means that the two time scales are not distinguishable.

Moreover, define k_i as the magnitude of scale difference between h_{i+1} and h_i, and $k_i = h_{i+1}/h_i$. Define N_i as the least number of simulation cycles which is required by the performance evaluation of layer i entity. Because the least necessary length of a whole simulation run, L, is determined by the time scale of the entity of the highest layer. If the hierarchical control has m layers, L can be estimated by:

$$L = N_m h_m = N_m K h_1 \tag{1}$$

where $K = \prod_{i=1}^{m-1} k_i$.

If the averaged wall-clock time for one simulation cycle of time scale h_i is t_i, then the total wall-clock time of simulation T is therefore:

$$T = N_m t_m = N_m K t_1 \tag{2}$$

It can be seen from equation (2) that if there are significant differences in the time scales of different layers, K will become dominant to determine the total wall-clock time of simulation. The problem of scale gap is that, in a simulation where the time scales of entities cross a large order of magnitude, the required time to perform such simulation may be impractically long if the lower layer model is not scalable. Our goal is to accelerate simulation. In equation (2), this is equivalent to shorten either the scale difference K or the unit time of model solving t_1. The former approach corresponds to the multi-scale modeling techniques. In this work, we focus on the latter approach, i.e. we will shorten t_1. This will be achieved by a method called *Steady State Leaping* (SSL).

4. STEADY STATE LEAP

4.1 Model Representation

In practice, a process simulation often adopts a modular based model building approach. Each unit operation model is encapsulated by a unified information structure. Fig. 4 shows the input/output interface of a unit operation model. The input/output streams represent the characteristics of the material streams, such as temperature, density, flow rate, pressure, etc. And the dashed line represents the control input signal and it connects the plant model

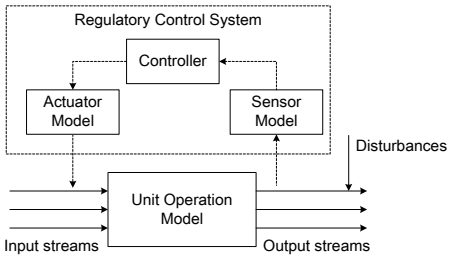

Fig. 4. Input and output of a unit operation model.

with controller. There are also boundary conditions of the model including variables related to the operation equipment, such as height, width, diameter, rotation speed, etc.

The whole plant model is assembled by connecting unit operation models, i.e. they are connected by the material streams according to the process flowsheet. There are two primary kinds of model representation: steady state model and dynamic model, corresponding to the unit operating at steady state and transient state respectively.

The dynamic model of a processing plant can be represented in the form:

$$\begin{cases} \mathbf{F}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{d}, \mathbf{u}, t) = 0 \\ \mathbf{H}(\mathbf{x}, \mathbf{y}) = 0 \\ \mathbf{G}(\mathbf{y}) \geq 0 \end{cases} \tag{3}$$

where $\mathbf{F}$ is a set of dynamic functions defining the state of unit operation model $\mathbf{x}$ to the vector of disturbance signals $\mathbf{d}$ and control signals $\mathbf{u}$. Vector $\mathbf{y}$ is the output of the model. $\mathbf{H}$ is a set of auxiliary algebraic equations derived from physical principles, and $\mathbf{G}$ is a set of inequalities that must be satisfied (Barton (1997)). If the dynamic behavior is not considered, model (3) will be reduced to its steady state form:

$$\begin{cases} \mathbf{F}(0, \mathbf{x}, \mathbf{d}, \mathbf{u}, t) = 0 \\ \mathbf{H}(\mathbf{x}, \mathbf{y}) = 0 \\ \mathbf{G}(\mathbf{y}) \geq 0 \end{cases} \tag{4}$$

where functions $\mathbf{F}$, $\mathbf{H}$ and $\mathbf{G}$ are similarly defined as before.

4.2 Algorithms

The proposed algorithms are applied under the framework of sequential modular approach (SMA), where the whole plant model that is obtained by connecting unit modules is solved using the sequential modular approach (SMA) (Hillestad and Hertzberg (1988)). The use of SMA in dynamic real-time simulation of process systems has been reported in Schopfer et al. (2004); Wang et al. (2009). This strategy successively executes the process unit models according to the connectivity of process units. The sequential modular solver first finds out if there exits recycle streams in the flowsheet, i.e. an output material stream is fed back to one of its upstream units. If no recycle is found in the process, the solver will successively execute each module and propagate the calculated results among the modules according to the connection relations. When there are recycle streams, the coordinator of the solver will tear each recycle stream to make it acyclic. A predicted feedback stream will be inserted to replace the torn stream.

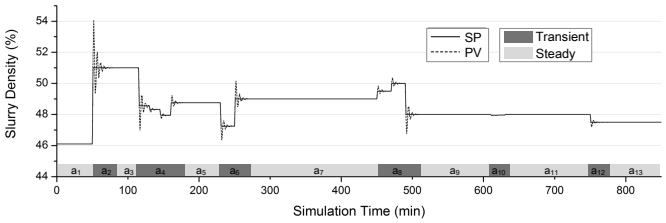

Fig. 5. Illustration of steady and transient state switching.

The error introduced by the prediction is controlled by a fix-point iterative algorithm.

With the sequential modular approach, the simulation engine calls the solver routine to sequentially calculate the unit operation models and update the state of process. In a dynamic simulation, the model is solved and the state is updated in each simulation cycle because the input, control and disturbance of the model are time-varying and the model state needs to be updated timely to keep the dynamic behavior of the system. In general, solving the model of (3) requires much more computational effort than that of (4). The idea of steady state leap is that: If it can detect that the simulation has reached a steady-state, the dynamic model should be substituted with its steady-state form to save computation effort.

To illustrate the idea, an example is presented in Fig. 5, where the density of a slurry sump is controlled by an additional water feed. As stated early, the recorded simulation trace is actually the time sequences of process variables (PVs). The figure plots a segment of one of the PVs, where X-axis represents simulation time and Y-axis represents the simulated value of the PV (short dash line), which in this case is the density of the slurry in the sump, tracking the set point (SP) given by the regulatory controller.

In Fig. 5, we can observe that the value of PV is either constant or fluctuating. Now, let's say that the simulation is at a *steady state* if all the PVs of the process model are constant over the time period of interest, and the simulation is at a *transient state* otherwise. In reality, steady state indicates the mass/energy equilibrium of the processing system. Such an equilibrium state will be broken and followed by a transient period if there are disturbances that occurred or the SPs given by controller have changed. In the former case, regulatory controller will stabilize the process when disturbances cause that PV deviates SP. The latter situation happens when a new control objective, which is derived by optimizer for instance, is to be implemented by the regulatory controllers to move the system to a different operating point. Fig. 5 illustrates the switching between steady and transient states. In the figure, transient and steady state are marked by the dark grey area and the light grey area respectively. For example, during the period a2, the process experiences a transient period after the SP changes from 46.1% to 51.0%. It is then stabilized back to steady state for a period a_3 and enters a new transient period a_5 caused by a series of adjustment of SP.

During a simulation, once the process enters a steady state, the output of model will be constant until a disturbance or new control input forces the process into a transient state. Therefore, it is actually not necessary to solve the

dynamic model at a steady state considering and it only needs to be resumed when an event indicating disturbance or a change of control input has occurred. For the problem of simulation acceleration, it is natural to consider letting the dynamic solver skipping steady state periods in expect that the total computation time can be shortened. To be more specific, if the simulation of a certain operation unit is at a steady state at time t, and it is also known that there will be no disturbance and control inputs in the horizon from t to $t + h$, where h is the length of time step, the simulator will solve model in its steady-state form of (4) instead of its dynamic form of (3). This substitution will not change final system state at the end of current time step $\mathbf{y}(t + h)$ because we know that the process is at a steady state, and $\dot{\mathbf{x}}$, $\mathbf{d}$ and $\mathbf{c}$ are all zero, there is no need to solve the differential equation $\mathbf{F}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{d}, \mathbf{u}, t) = 0$ during t to $t + h$.

The implementation of steady state leap requires that simulator can detect the change of state. The following algorithm has been used to detect state change. Indeed, the change from steady state to transient state at some simulation cycle i can only be the result of disturbance or new control input which breaks the equilibrium state. The algorithm will identify the state as steady only if the differences between all the output data at current and the last cycle are no greater than predefined calculation error threshold. This ensures that noise [2], whose magnitude should be below the threshold, will not have effects on state identification.

Algorithm 1 Simulation engine

```
1: procedure SimulationSSL   ▷ simulation engine of SSL
2:     T ← 0                 ▷ T: simulation time variable
3:     Init(o)               ▷ o: reference to simulation object
4:     while T ≤ t_max AND o.endFlag ≠ TRUE do
5:         o.state.current ← StateIdentification(o)
6:         if o.state.current =STEADY then
7:             SolverS(o)    ▷ call steady state solver
8:         else              ▷ current state is TRANSIENT
9:             SolverD(o)    ▷ call dynamic solver
10:        end if
11:        Update(o.state)   ▷ update simulation state
12:        T ← T + h
13:    end while
14: end procedure
```

The main algorithms associated with SSL are illustrated in Algorithm 1 and Algorithm 2. At the beginning of each simulation cycle, the simulation engine will first check the process state by calling function `StateIdentification`. If the process is previously at steady state, the `StateIdentification` will return `TRANSIENT` only when it finds out the event list (el.current) for the current simulation cycle is not empty indicating that there will be disturbance or control signal in this cycle which must be calculated by dynamic solver `SolverD`, which could be any solving routine which is appropriate to the equation. If the process is previously at transient state, the process will return `STEADY` only when the even list is empty and the difference of each element in the output vector between the last two simulation cycles is below the predefined calculation error threshold (`ERROR_THRESHOLD`). Note that `StateIdentification` is called at the beginning of each simulation cycle and the current data have not been calculated. It has to use the outputs of last two cycles to identify a switch of process state.

[2] In this paper only the deterministic part of the process model is considered.

Algorithm 2 State identification

```
1: function StateIdentification(o)   ▷ o: reference to simulation object
2:     cs ←IsEmpty(o.el.current)     ▷ cs: current state; el: event list
3:     err ←MaxDiff(o.d.last, o.d.last.last)   ▷ o.d: output data vector
4:     if o.state.last =STEADY then ▷ last cycle=STEADY
5:         if cs =FALSE then         ▷ event list is not empty
6:             StateIdentification ←TRANSIENT
7:         else
8:             StateIdentification ←STEADY
9:         end if
10:    else                          ▷ last cycle=TRANSIENT
11:        if cs =TRUE AND err ≤ERROR_THRESHOLD then
12:            StateIdentification ←STEADY
13:        else
14:            StateIdentification ←TRANSIENT
15:        end if
16:    end if
17: end function
```

In reality, a processing production system operates at a steady state in most time except starting-up and shutting-down. During a steady state, all state variables of the process are constant in spite of ongoing processes that strive to change them. However for the simulation, it entering a steady state requires that all the process variables are constant. From Fig. 5 we can see that if there are a large number of process variables in the model, the total length of being at steady states may be reduced, because the state of whole process is now the superposition of the state of each process variable. In other words, the more sparsely the transient state occurs, the more time can be saved by SSL.

Moreover, in some process, we can identify that some of the model variables are only dependent on the local unit model and the variables of unit models are only connected if the two models are connected by material streams. This means that, even if the whole plant model is large in its number of process variables, it may be partitioned into several sub-systems and apply the SSL technique in a distributed way. A typical algorithm of the parallel time stepped simulation engine for the execution of each local processor is illustrated in Algorithm 3. Executing SSL in parallel requires the decoupling of the model calculations for multi-processor environment. However, those topics are beyond the foci of this paper. The relevant research on parallelizing process flowsheet simulation can be found on Chimowitz and Bielinis (1987) and the SPEEDUP technology of Aspen Tech Paloschi and Zitney (1999).

Algorithm 3 Simulation engine of local virtual processor

```
1: procedure SIMULATIONSSL_LV ▷ local simulation engine
2:     lvT ← 0 ▷ T: local simulation time
3:     INIT(lo) ▷ lo: reference to local simulation object
4:     while lvT ≤ t_max AND lo.endFlag ≠ TRUE do
5:         RECV( ) ▷ receive messages from connected processors
6:         o.state.current ← STATEIDENTIFICATION(o)
7:         if lo.state.current =STEADY then
8:             SOLVERS(lo)
9:         else
10:            SOLVERD(lo)
11:        end if
12:        UPDATE(lo.state)
13:        lvT ← lvT + h
14:        SEND( ) ▷ send out local model state to connected processors
15:    end while
16: end procedure
```

5. ACCELERATION PERFORMANCE

5.1 Case Study: An Ore Milling Process

In this section, an example of two-stage grinding process system is introduced for the sake of performance evaluation. Grinding is the fine phase of comminution after coarse step of size reduction such as crushing. Its goal is to reduce the particle size of ore so that the economically valuable substance in the ore can be more efficiently separated by the subsequent process, such as flotation or magnetic separation.

In this case, the flowsheet arrangement consists of two grinding circuits. The first circuit is formed by 4 ball mills of $\phi 3200 \times 3100mm$ and 8 hydrocyclones of $\phi 500mm$, and the second circuit includes 4 ball mills of $\phi 3200 \times 3500mm$ and 24 hydrocyclone of $\phi 350mm$. First, fresh ore is being continuously fed through a vibratory feeder onto a conveyer belt, and along with mill water, it is fed to ball mill I for grinding. The ball mill I slurry is mixed with dilution water and discharged to hydrocyclone, where the coarser particles in the discharged slurry are fed back to ball mill I for re-grinding. The finer particles go to a set of 11 high frequency vibrating sieves. The coarse particles are flushed to sump, and the stream of fine particles are joined with the final product. Water is added to lower pulp density. Then, the diluted slurry is pumped into the second set of hydrocyclones by a constant pressure. Hydrocyclones separate the finer particles to form the product stream (overflow) from an underflow stream in which the coarser particles are sent to ball mill II for a second stage of grinding. Finally, the discharged slurry of ball mill II flows into sump.

The grinding process is controlled by a two layered control system. On top of the regulatory control layer, an optimizer called *optimal setting control* (OSC) is deployed. The ultimate goal of OSC is, by giving optimal setpoints of the regulatory controls, so as to improve product quality. For a ball mill grinding circuit, its product quality is measured by particle size. This is the most concerned quality index for process engineers and comminution experts. Sieve analysis can give the distribution of particle size against mesh size. But in practice, it is common to use a scalar, P_{200}, which is defined as the weight fraction of solids in the product stream passing a 200 mesh Tyler series sieve (particle size$\leq 74\mu m$).

The requirement for product particle size is often expressed as a range: $[P_{200}^{min}, P_{200}^{max}]$ and OSC wants to let the product particle size be in the required range, and also be close to P_{200}^{min} as much as possible. To realize this objective, the optimal setting controller needs to: (i) identify and choose a set of process variables which are closely related to the optimization goal and can also be controlled by the local controller; (ii) to auto-adjust the set-points of the chosen process variables and let the local controllers track the set-points so as to drive the product indices into the required ranges. The details of the OSC algorithms can be referenced in Zhou et al. (2009).

5.2 Simulation Goal: Testing the Optimal Setting Control System

The goal of simulation is to verify the optimal setting control software, and simulation is used at the stage when the coding of OSC software has finished. More specifically, we want to check by simulation if the constraints of OSC on product quality and quantity, which are measured by the particle size and density of the product stream, are not violated during the simulation experiments.

Since we are using simulation to verify the optimal setting controller, it is necessary to configure the length of each simulation run so that the simulation can cover at least a whole cycle of the processing operation. Though the grinding process operates continuously, there are two periodic cycles which can be identified. First, because of the existence of stream circuits, some parts of ground ore will circulate several times in the process until its particle size is small enough to be separated by hydrocyclones or sieve. In the two-stage grinding process, it estimates that it takes about 30 minutes for more than 95% of a certain fed ore being discharged. Second, the balls in ball mills are replenished every 4 four hours. This will cause a rather sharp change in the working condition of the grinding process, and it corresponds to a new initial condition for simulation. In summary, the minimal length of each simulation run is set to 4 hours so that both cycles can be covered.

5.3 Setup of Simulation Experiment

The simulations are performed on NEUSimMill Lu and Chai (2013), a dynamic process simulator for the test and verification of grinding process control. The objective is to evaluate the OSC algorithm and its system implementation on NEUSimMill. The simulation platform uses a hardware-in-the-loop simulation (HILS) architecture, i.e. the control system hardware is connected to process models to establish a counterpart of the real control system used in industrial processing plant. With the ability of connecting to real control systems, the control programs are therefore directly validated in a running environment of actual hardware setup in real time. Fig. 7 shows the connection scheme of the HILS hardware-in-the-loop simulation architecture which is configured for this experiment.

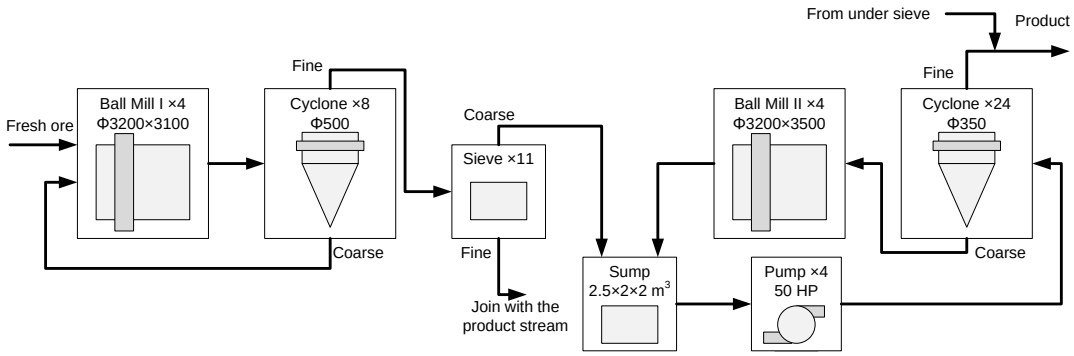

Fig. 6. Flowsheet of a two-phase ball milling process.

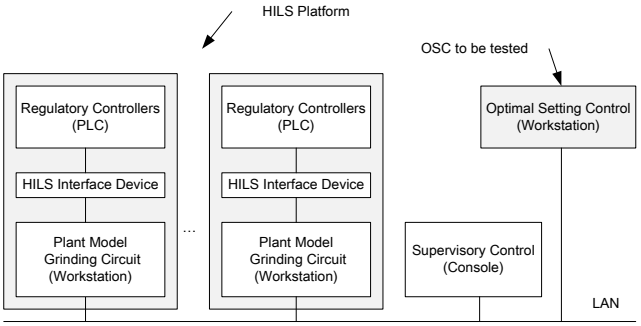

Fig. 7. The Hardware-In-the-Loop Simulation architecture of the simulation platform.

The OSC system to be tested is connected to the platform by Ethernet. The regulatory controllers are Rockwell Logix5561 PLC (Programmable Logic Controller), and the plant model workstations are Dell Precision T5500.

NEUSimMill adopts a modular based modeling approach to calculate the dynamics of the whole process. Each modular formulates the dynamic responses of a unit operation model (mills and classifiers etc.) to control or disturbances, i.e. the hardness, density and particle-size distribution of the fresh ore feed. And each module is independently implemented, and individually integrated over a common time interval; and they are only connected according to the interconnection variables of the flowsheet which represent material streams. Such a strategy allows the integration of various types of individual modules which include differential equations, and also those based on empirical input-output correlations and discrete rules etc. Unit operation models in NEUSimMill are developed based on the mass balance principle and the hybrid intelligent modeling approach. The former sets up the master structure of the model equation. And the latter approach employs heuristic inference tools such as fuzzy logic and artificial neural network in the estimation of important model parameters which often depend on working condition of the plant. is a unit operation model (mills and classifiers etc.) Due to limitation of space, the process model will not be elaborated here. Those details can be found in our work Tie et al. (2007).

5.4 Experiments and Discussion of Results

In this case, the control system is hierarchical structured with two layers: an optimal setting control layer and a regulatory control layer. There is scale gap between the OSC and regulatory control as the OSC typically operates on a time scale of minute and the regulatory control on second. Moreover, the optimal setting controller depends on the local regulatory control loops to realize its goal, and one OSC manages multiply sets of regulatory controllers of different series of grinding circuits. The simulation load is heavy because it needs to simulate all the 4 series (see Fig. 6) and their regulatory controllers in parallel.

Table 1. The operation conditions and requirements of product quality in terms of particle size

Variables	**Stage I**	**Stage II**
Ore feed rate (t/h)	74	N/A
Ball mill concentration (%)	78.2	N/A
cyclone feed Pressure (kPa)	N/A	110
stage I & II PSD requirements	$\geq$60	$\geq$70

Since OSC only starts to act when it detects that the process has been derailed from optimal operation state by disturbances, we need to add into the system artificial disturbances to activate OSC operation. In the experiment, a random disturbance in the hardness [3] of the raw ore is considered. The ore hardness of input raw ore will in general degrade the grinding efficiency, which will trigger OSC to compensate such effect by changing the setpoints of regulatory controllers. Because the effect of changing ore hardness on the process is global, which means that it will propagate throughout each of the process units, by varying ore hardness we are expecting to have a series of regulatory control adjustments.

A random disturbance in the hardness of the raw ore is configured as a Gaussian distribution of mean $\mu = 12$ and variance $\sigma^2 = 0.05$. The change in hardness is assumed to happen discretely and the length of interval between changes is configured as exponential with a mean which will vary for each experiment run. The other operation conditions are listed in Table 1.

The performance evaluation of SSL is accomplished by two different groups of experiments: (i) the whole plant model runs serially on a single workstation. (ii) The plant model is distributed on two workstations in parallel. In the latter case, the whole plant model of two-phase ball milling process is decomposed into models of grinding circuit I and grinding circuit II. And note that each workstation

[3] Ore hardness is measured by Bond Work Index (BWI).

Table 2. Comparison of the required wall-clock simulation time of different setups.

$1/\lambda$ (min)[a]	Group (i) (sec)[b]		Group (ii) (sec)[c]	
	Without SSL	With SSL	Without SSL	With SSL
30	12757	8047	7831	4776
60	12346	5305	7817	3042
90	12325	4225	7793	2585
120	12403	3802	7840	2341
150	12377	3677	7798	2291

[a] The averaged interval of disturbance in ore hardness.
[b] The first group refers to experiments where the plant model runs on a single workstation.
[c] The second group refers to those where the plant model is decomposed into two parts running on two workstations in parallel.

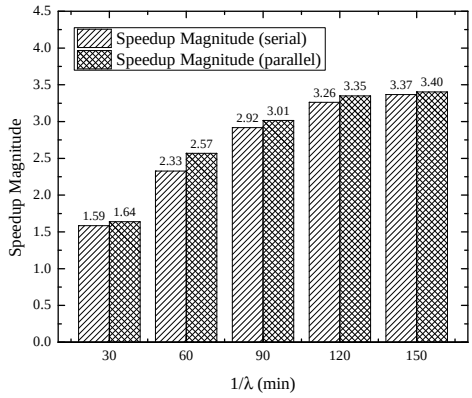

Fig. 8. Comparison of speedup magnitudes achieved by SSL in serial and parallel simulation.

corresponds to all series of the grinding circuit which are operating in parallel.

We expect that SSL can speedup simulation by reducing unnecessary computation during steady period. The experiments are design to vary the averaged interval of disturbance in ore hardness of each simulation run, and the wall-clock simulation time is recorded to check the acceleration performance of SSL under different conditions. Let $1/\lambda$ be the averaged interval of disturbance in ore hardness. We vary $1/\lambda$ because we cannot directly control that the simulation will be at a steady state; and by adjusting the frequency of disturbance occurrence, the total length of transient period can be indirectly adjusted.

The experimental results are listed in Table 2, which are obtained by varying the occurrence rate of the disturbance in the hardness of raw ore from 30 minutes to 90 minutes. Each simulation run will terminate when it reaches a predefined length representing 6 hours (360 minutes) of actual operation of the production system, and the wall-clock time the simulation run is recorded. Considering that OCS is designed to operate in minutes and the disturbance happens at most in 90 minutes, a simulation length of 6 hours is long enough to ensure multiple cycles of OCS operations.

We observe in Table 2 that varying the averaged interval of disturbance in ore hardness does not have a notable impact on the wall-cock time of simulation without SSL. This is because the simulator engine will have to solve the model in each time-step regardless the state. To evaluate the scale of speedup achieved by applying SSL, we define speedup magnitude as the ratio of wall-clock time of simulation without and with SSL. The speedup magnitude is calculated and compared in Fig. 8. We can clearly observe a direct ratio relationship between speedup magnitude and disturbance interval. This is in accordance with intuition: the less frequent the disturbance occurs, the more simulation time can be saved by SSL. Also, the relationship between speedup magnitude and disturbance interval is not linear. In fact, the speedup magnitude has not grown linearly with the slowing down of disturbance occurrence and the speedup magnitude appears to approach a saturation level. Clearly, this represents that the simulation will eventually degrade to a steady-state simulation if all the control and disturbance are zeroed.

It can also be noticed in Fig. 8 that the speedup magnitude of the parallel case is slightly greater than that of the serial case in general. Remember that the condition of applying SSL is that the whole plant model has reached a steady state because there is a single global solver to handle the whole model. In the case of parallel simulation, each local model, which is a part of the whole model, runs on a workstation and it is handled by a local solver. Considering that there is no global solver, it only needs that the local model has been at a steady state to apply SSL. In general, a local model is more likely to reach steady state because it has fewer units than the whole plant model. In the parallel example of grinding process simulation, the whole plant model has been decomposed into two local models: grinding circuit I model and grinding circuit II model. Note that the sump is the first unit operation of the grinding circuit II model. In this flowsheet, the output rate of sump is solely determined by the pump. The sump can be considered as a stream buffer, which has the effects of neutralizing the variation of flow rates of its input streams. This also makes it more likely that the grinding circuit II model is at a steady state. Since the speedup magnitude of SSL is in direct ratio to the total period of being at steady state of the model, the SSL applied to parallel simulation achieves a slightly better speedup performance.

6. CONCLUSIONS

This paper has introduced steady state leap, an approach fitting the tasks of simulating industrial process control system with hierarchical structure, which may present the problem of scale gap. The technique calculates the system state using its steady state model instead of the dynamic model only in a simulation cycle where the system has reached a steady state and no events will move the system out of the current state. The analysis has concluded that the successful application of SSL technique depends on two conditions: (i) Disturbances are known by the simulator at the beginning of each simulation cycle. (ii) The state of the model, i.e. whether it is at a steady state or transient state, can be identified by the simulation engine. The first condition is ensured by the fact that the disturbance is actually a pre-generated sequence of disturbance event, which is "known" in advance by the simulator. The second condition is handled by the process state identification algorithm. Experiments on the simulation of the optimal setting control system of a two-phase grinding process have validated the performance of SSL. First, SSL can

accelerate the whole simulation to a magnitude which depends on the occurrence frequency of disturbance. In general, the less frequently the disturbance happens, the more speedup can be achieved. Second, SSL can be applied in combinition with parallel and distributed simulation to further accelerate the simulation speed.

REFERENCES

Barton, P.I. (1997). Industrial experience with dynamic simulation. Technical report, Department of Chemical Engineering, Massachusetts Institute of Technology.

Boer, C. A., d.B.A. and Verbraeck, A. (2006). Distributed simulation in industry - a survey part 2 - experts on distributed simulation. In *Proceedings of the 2006 Winter Simulation Conference*, 1061–1068.

Chimowitz, E.H. and Bielinis, R.Z. (1987). Analysis of parallelism in modular flowsheet calculations. *AIChE Journal*, 33(6), 976–986.

Cox, R.K., Smith, J.F., and Dimitratos, Y. (2006). Can simulation technology enable a paradigm shift in process control?: Modeling for the rest of us. *Computers & Chemical Engineering*, 30(10-12), 1542–1552.

Craig, I. (2011). Control in the process industries. Technical report, IEEE Control System Society.

Darby, M.L., Nikolaou, M., Jones, J., and Nicholson, D. (2011). Rto: An overview and assessment of current practice. *Journal of Process Control*, 21(6), 874 – 884.

Davis, P.K. (2005). Introduction to multiresolution, multiperspective modeling (mrmpm) and exploratory analysis. Technical report, RAND National Defence Research Institute.

Fujimoto, R.M. (2000). *Parallel and Distributed Simulation Systems*. John Wiley & Sons.

Hillestad, M. and Hertzberg, T. (1988). Convergence and stability of the sequential modular approach to dynamic process simulation. *Computers & Chemical Engineering*, 12(5), 407–414.

Hillestad, M. and Hertzberg, T. (1986). Dynamic simulation of chemical engineering systems by the sequential modular approach. *Modeling, Identification and Control*, 7(3), 107–127.

Ingram, G.D. and Cameron, I. (2004). Challenges in multiscale modelling and its application to granulation systems. *Developments in Chemical Engineering and Mineral Processing*, 12(3-4), 293–308.

Larsson, T. and Skogestad, S. (2000). Plantwide control - a review and a new design procedure. *Modeling, Identification and Control*, 21(4), 209–240.

Lu, S. and Chai, T. (2013). The development of a hardware-in-the-loop simulation system for the control of ball mill grinding process. *IEEE Transactions on Automation Science and Engineering (submitted)*.

Maroudas, D. (2000). Multiscale modeling of hard materials: Challenges and opportunities for chemical engineering. *AIChE Journal*, 46(5), 878–882.

Paloschi, J.R. and Zitney, S.E. (1999). Parallel dynamic simulation of industrial chemical processes on distributed-memory computers. *Computers & Chemical Engineering*, 23, Supplement(0), S395 – S398.

Qin, S. and Badgwell, T.A. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7), 733 – 764.

Reynolds, Jr., P.F., Natrajan, A., and Srinivasan, S. (1997). Consistency maintenance in multiresolution simulation. *ACM Trans. Model. Comput. Simul.*, 7, 368–392.

Santos, R.A., Normey-Rico, J.E., Gómez, A.M., Arconada, L.F.A., and Moraga, C.d.P. (2008). Distributed continuous process simulation: An industrial case study. *Computers & Chemical Engineering*, 32(6), 1195–1205.

Schopfer, G., Yang, A., von Wedel, L., and Marquardt, W. (2004). Cheops: A tool-integration platform for chemical process modelling and simulation. *International Journal on Software Tools for Technology Transfer (STTT)*, 6(3), 186–202.

Tie, M., Bi, J., and Fan, Y. (2007). Hybrid intelligent modeling approach for the ball mill grinding process. In *Advances in Neural Networks - ISNN 2007*, 609–617.

Turanyi, T., Tomlin, A.S., and Pilling, M.J. (1993). On the error of the quasi-steady-state approximation. *The Journal of Physical Chemistry*, 97(1), 163–172.

Vora, N. and Daoutidis, P. (2001). Nonlinear model reduction of chemical reaction systems. *AIChE J.*, 47(10), 2320–2332.

Wang, Y., Li, L., Gui, W., and Yang, C. (2009). Realization and application of mineral processing simulator using sequential modular approach. *Computer Engineering & Application*, 45(7), 224–226.

Zhou, P., Chai, T., and Wang, H. (2009). Intelligent optimal-setting control for grinding circuits of mineral processing process. *Automation Science and Engineering, IEEE Transactions on*, 6(4), 730–743. 1545-5955.