On simplified handling of state events in time-domain simulation

Davide Fabozzi
University of Liège
Liège, Belgium
davide.fabozzi@ulg.ac.be

Angela S. Chieh
RTE - DMA
Versailles, France
angela.chieh@rte-france.com

Patrick Panciatici
RTE - DMA
Versailles, France
patrick.panciatici@rte-france.com

Thierry Van Cutsem
FNRS and University of Liège
Liège, Belgium
t.vancutsem@ulg.ac.be

Abstract - The power system models typically used in dynamic simulations involve discrete events in addition to the standard differential-algebraic equations. Those events cause the system to jump from one continuous behavior to another. Solvers have to handle those jumps. This paper focuses on simplified time simulation where large steps are used in conjunction with stiff-decay integration methods to obtain approximate solutions in short computing times. In the proposed simulation scheme, the simulation time steps are not synchronized with the system jumps, which are treated *a posteriori* in a corrective step. In this context, the paper analyzes several simple nonlinear models involving limiters, switches, minimum gates, etc. from which some precautions to be taken at the modelling and solving stages are stressed. The paper also reports on results obtained with a representative power system model.

Keywords - hybrid systems, differential-algebraic equations, stiff-decay methods, simplified simulation, long-term dynamics

1 Introduction

POWER systems are modelled by large sparse systems of nonlinear stiff differential-algebraic equations. The size of these systems can easily reach tens of thousands of variables for large scale cases. In some applications detailed simulations are not feasible because of the excessive computational effort. For instance, dynamic security assessment requires long-term simulation of numerous contingencies in short periods of time.

Simplified simulation can be used in applications where a loss of accuracy can be accepted in exchange of a faster computation. This can be achieved by:

- *model simplification*. Models are simplified in order to become smaller and/or less stiff. An example that combines reduction in size and stiffness is the quasi steady-state approximation of long-term dynamics. In this approach the short-term dynamics are replaced by their equilibrium equations [1];
- *large-step integration*. In this approach, no simplification is performed on the models, i.e. they are the same as in detailed simulation, which avoids maintaining several models of the same system. A simplified solver with appropriate numerical stability is used to "filter out" the fast dynamics [2].

Integrating with large steps may bring a significant reduction of computational effort. It is known that stiff-decay methods allow integrating with steps larger than the smallest system time constants (which of course will discard some fast dynamics) [3]. A method has stiff-decay if the error introduced by a bounded disturbance on the state variable of the $\dot{x} = -\lambda x$ equation (with $\lambda > 0$) tends to zero as the integration step tends to infinity. These methods are used in both detailed simulations [3, 4, 5] and simplified ones [2]. In fact, for a large class of continuous-time dynamics, there is virtually no convergence limit for the step size taken by a stiff-decay method: simply, an integration step much larger than the time constant would practically lead to the solution of the equilibrium equation of that dynamics. Thus, the step size is not limited by stability but only by accuracy.

By way of illustration, Fig. 1 shows the time evolution of a bus voltage obtained with respectively a detailed and a simplified solver. The former uses a second-order method with steps of 0.01 s while the latter uses a first-order stiff-decay method with steps of 0.5 s. The case refers to a long-term voltage unstable scenario of a test system used in Section 6. As can be seen, the simplified simulation overlooks some of the oscillations but matches the detailed one as soon as the large transients have died out. The large steps allow substantial reduction in computing time while rendering the overall evolution of the system. As a rule of thumb, the speed-up ratio approches the step size ratio, i.e. 50 in the above example.

Power systems are also hybrid systems, i.e. they involve variables which change at specific instants of time [6]. These systems are more problematic to handle with large steps. Relatively few attention has been paid in the literature on power system dynamic simulation to the handling of discrete parts of the model. In the general case, whether the solver uses constant or variable step size, those instants where variables change do not coincide with a simulation time step. Thus, for detailed simulation, the step size should be adjusted to make a step coincide with the upcoming event. On the other hand, as it will be shown later on, some care must be taken if the step size is not reduced, as it is the case in simplified simulation with large, fixed steps: shifting the event in time may cause variables to converge to wrong values, or even not converge, if some modelling and solving precautions are not taken.

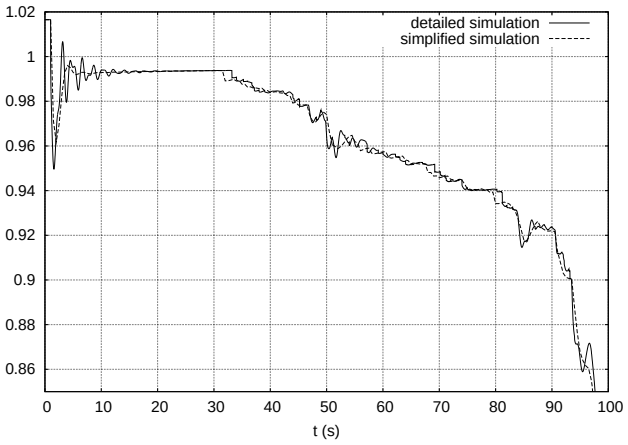

Figure 1: Detailed vs. simplified simulations; bus voltage (pu)

The objective of this paper is to investigate the problems that arise from the interaction of large step integration with the hybrid dynamics of power system models. The emphasis is on models used for stability analysis, i.e. derived under the phasor approximation [7]. The paper analyzes in some detail jumps taking place in some nonlinear models such as limiters, switches, minimum gates, etc. and proposes guidelines to handle the resulting changes in continuous-time dynamics. Some possible limitations, such as non-existence of solution of the discretized problem are also stressed. The proposed algorithm has been implemented in an industry-grade software in the context of the European FP7 project PEGASE [8], and simulations of a representative controller model are presented.

Notation

x_t denotes the values of the state x at the discrete time t

x^k denotes the value of the state x at the k-th iteration of an integration formula (discrete time omitted for clarity)

$x_t^{(j)}$ denotes an intermediate solution for the state x at discrete time t, considering the j-th candidate jump

2 Hybrid system dynamics

The dynamic model of a power system under phasor approximation takes on the form:

$$\mathbf{\Gamma}(\mathbf{z})\,\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{z}) \quad (1)$$

$$\mathbf{z}(t_k^+) = \mathbf{h}(\mathbf{x},\mathbf{z}(t_k^-)) \quad (2)$$

where $\mathbf{\Gamma}$ is a diagonal matrix with $(\mathbf{\Gamma})_{\ell\ell} = 0$ if the ℓ-th equation (1) is algebraic, and $(\mathbf{\Gamma})_{\ell\ell} = 1$ if the ℓ-th equation (1) is differential.

The continuous-time dynamics equations (1) deal with the network and various phenomena and controls, ranging from the short-term dynamics of power plants, static var compensators, induction motors, etc., to the long-term dynamics of secondary frequency control, load self restoration, etc. These equations can be either algebraic or differential. Vector $\mathbf{x}$ includes the corresponding (differential and algebraic) state variables, such as rectangular components of bus voltages, rotor angles, flux linkages, motor slips, controller state variables, etc. The components of $\mathbf{x}$ take values in $\mathbb{R}$ or in an interval of $\mathbb{R}$.

The discrete-time equations (2) capture:

1. the response of discrete controls and protections acting with various delays in shunt compensation, secondary frequency and voltage control, load tap changers, overexcitation limiters, etc. The corresponding components of $\mathbf{z}$ are shunt susceptances, generator setpoints, transformer ratios, etc.
2. the change in continuous-time equations caused by limits, switches, minimum gates, hysteresis, etc. The corresponding variables $\mathbf{z}$ act as "switches". Note that the diagonal entries of $\mathbf{\Gamma}$ may vary with $\mathbf{z}$, as suggested by (1), since a change in z may cause an equation to change from differential to algebraic and vice versa.

The components of $\mathbf{z}$ take values in a countable subset of $\mathbb{R}$.

A classical example of the second category above is the integrator with non-windup limit, shown in Fig. 2.a. Let us define z as a discrete variable with value 1 if $x < x_{max}$ and value 0 if $x = x_{max}$. This component is described by the following continuous-time equation of type (1):

$$z\,\dot{x} = z\,u + (1-z)\,(x - x_{max})$$

Thus, $\Gamma(z) = z$. The changes in z can be described by the *state transition diagram* of Fig. 2.b [9, 10].

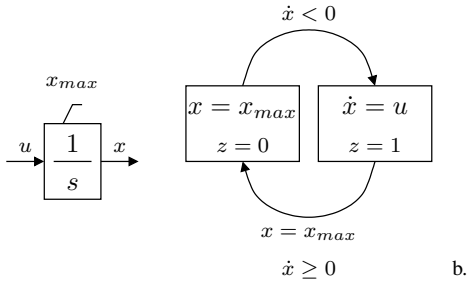

Figure 2: Integrator with non-windup limit

Continuous-time states $\mathbf{x}$ evolve over intervals of time. The algebraic states in $\mathbf{x}$ can also vary over instants, when some $\mathbf{z}$ components change. The discrete states $\mathbf{z}$ evolve over discrete instants. With the notation of Eq. (2), $\mathbf{z}$ changes from $\mathbf{z}(t_k^-)$ to $\mathbf{z}(t_k^+)$ at time instants t_k dictated by *events*. In the example of Fig. 2, the events are the transitions from $z = 0$ to $z = 1$ and vice versa.

The dynamic behavior over a continuous interval according to Eq. (1) is referred to as *flow*. The variation of some components of $\mathbf{z}$ causes the passage from one flow to another at one point in time. This passage is referred to as a *jump*.

Events are classified according to whether it is possible or not to forecast when they take place [10]. In the former case, they are called *time events*, and in the latter case, *state events*. State events take place when a *jump condition* is satisfied, as a result of system evolution. It is usually possible to specify the event condition as a zero-crossing function, i.e. the event takes place when the value of that function passes through zero. A general solver is

thus in charge of monitoring the zero-crossing functions in order to identify the *event times*.

3 Handling of state events

3.1 Computations required by detailed simulation

The most important effect of events on dynamic simulations is that they force the solver to adjust the length of continuous-time evolution intervals in order to land over the event times. This mechanism, in case of many events, leads to a reduction of the average step size (and hence an increase in the number of steps). Moreover, when Newton iterations are performed to solve both the algebraic and the algebraized differential equations (which is mandatory in case the step is large with respect to the smallest system time constant), the Jacobian of those equations with respect to the state $\mathbf{x}$ has to be updated and re-factorized when the step size h changes (the latter is involved in the algebraized differential equations).

In detailed simulation the handling of events requires the solver to perform several tasks in addition to the aforementioned step size adjustment. By definition, a jump induces a change in some components of $\mathbf{f}$, and hence a discontinuity of $\dot{\mathbf{x}}$. Furthermore, the jump may cause some algebraic variables to undergo a discontinuity, i.e. there is at least one ℓ such that $x_\ell(t_k^-) \neq x_\ell(t_k^+)$ and $(\mathbf{\Gamma})_{\ell\ell} = 0$. In this case, the solver has to determine the new values $x_\ell(t_k^+)$. In principle, the occurrence of a discontinuity requires to momentarily decrease the order of the integration formula to one. Indeed, from a strict mathematical viewpoint, it is not correct to integrate over one of the those discontinuities; in fact, all numerical integration methods are based on polynomial expansions which cannot reproduce discontinuities. The well-known Backward Euler formula $x_{t+h} = x_t + h\dot{x}_{t+h}$, however, allows to integrate over a discontinuity because it has a form which does not involve time derivatives of state variables at the point of discontinuity. Therefore, and because it involves only one past value of the differential states, this scheme can be used for the time step that immediately follows a discontinuity.

The computations typically performed by a detailed solver are as follows. Let us assume that, at time t, it is intended to take a step of length h, to arrive at time $t + h$. Let us suppose that the zero-crossing function analysis informs the solver that a state event is going to take place at $t + h' < t + h$, where h' has to be determined with proper accuracy. The solver will thus integrate the current set of Eqs. (1) with a step h', impose the event, solve the associated discontinuity, change the flow and proceed with the simulation. In practice, the accurate identification of the event time $t + h'$ may require additional steps if the zero-crossing function is strongly nonlinear.

3.2 Proposed procedure for simplified simulation

As mentioned in the Introduction, for simplified but fast simulation purposes, it is of interest to take relatively large, constant integration steps. However, overstepping the correct event times is risky and may yield wrong results, or even divergence, unless some precautions are taken when modelling and solving.

The scheme that is proposed in this paper for simplified simulation can be outlined as follows, with reference to the above example:

1. at time t, starting from the state vector $\mathbf{x}_t$, take a step of length h, to arrive at $t + h$. Let $\mathbf{x}_{t+h}^{(1)}$ be the corresponding state vector
2. at this time the jump conditions are checked
3. if it is detected that a state event has occurred in between t and $t + h$, $\mathbf{z}$ is changed accordingly, and the step from t to $t + h$ is repeated with the flows corresponding to the new $\mathbf{z}$. This yields the new value $\mathbf{x}_{t+h}^{(2)}$ of the state vector
4. steps 2 and 3 are repeated, updating the state vector, until no jump occurs any longer or a maximum number of jumps is reached. This yields the final state vector for the current step
5. then, the simulation proceeds with a new step.

In comparison to the detailed simulation, the proposed approach will show a speed-up. Firstly, the step size will not be reduced, leading eventually to a smaller number of steps. Moreover, Jacobian updates to account for the varying step size are avoided. Lastly, if there is a discontinuity associated with the event, it is not solved but taken into account when computing the states at $t + h$ [2]. This leads to saving a number of Jacobian factorizations and Newton iterations.

3.3 Implementation considerations

Newton iterations are performed to pass from $\mathbf{x}_t$ to $\mathbf{x}_{t+h}^{(j)},\ j = 1, 2, \ldots$ It is essential to iterate until reaching a solution $\mathbf{x}_{t+h}^{(j)}$ accurate for the current flow, before checking possible jump conditions (step 2). In other words, the flow must not be changed during the Newton iterations. Otherwise, incorrect jumps can be experienced, in an unpredictable manner.

When a jump takes place, the Jacobian used in Newton iterations must be updated to reflect the change in equations (1). This update can be performed locally, using a decomposed formulation as discussed in [11]. For efficiency reasons, as long as the flow does not change (i.e. in between event instants), the Jacobian should be updated as infrequently as possible. This standard technique is usually referred to as "dishonest Newton scheme" [5].

Another issue is the initialization of the Newton iterations. In dynamic simulation it is customary to adopt predictor-corrector methods. In accurate simulation, when the steps are small compared to the dynamics involved and especially when event times are well separated, high-order prediction is used in order to start from an initial point hopefully closer to the solution. Furthermore, the difference between predicted and corrected values may be used to estimate the local truncation error [3]. In simplified simulation, however, the steps may be large compared to some of the dynamics. In this case a high-order prediction may be far from the actual solution. In addition, since the step size is not adjusted to match event times,

high-order prediction might be more dangerous than helpful. Finally, local truncation error is not assessed. All this leads to adopt a zero-order prediction, i.e. merely use the value of the state at the last point in time as initial guess for the new time step.

4 Solving and modelling precautions

Attention must be paid to formulating the flow so that it is differentiable with respect to $\mathbf{x}$. If this does hold true, the dishonest Newton scheme may lead to using an outdated Jacobian resulting in divergence of Newton iterations. A non-differentiable flow has to be reformulated as a combination of differentiable flows together with appropriate conditions to jump from one flow to another. This point is illustrated in Example 1 below.

Next, since relatively large step sizes are taken, the intermediate values $\mathbf{x}_{t+h}^{(j)}$ of the procedure may significantly deviate from the actual solution. Both the jump conditions and the flows must be formulated in a way that accommodates those deviations. In particular, the intermediate values $\mathbf{x}_{t+h}^{(j)}$ may fall outside the feasibility domain defined by the limits that are enforced by the jumps. The formulation must accommodate these infeasibilities. This is illustrated in Example 2 hereafter.

Last but not least, the procedure sketched in the previous section may lead to cycling between flows. A modification of the state transition graph can help solving this problem, as illustrated in Example 3 below. The list of recommendations in this Section is probably not exhaustive.

In the time plots of this section and of the next one, the solid line refers to the exact solution, the dotted line show unsuccessful trajectories and the dashed line corresponds to the correctly implemented simplified simulation.

4.1 Example 1

Consider the simple system in Fig. 3 (which of course could be embedded in a much larger model).

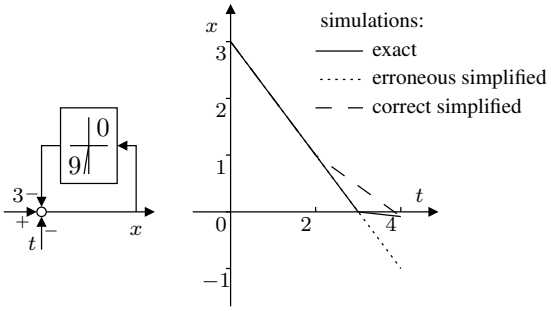

Figure 3: Example 1: block-diagram and simulation results

The corresponding flow equation is:

$$f(x,t) \stackrel{\text{def}}{=} \quad 0 = t - 3 + x + \min(0, 9\,x) \qquad (3)$$

where t is the time and x is an algebraic state. Assume the initial condition $x_0 = 3$ at $t = 0$.

Consider a simplified simulation with step size $h = 2$ s, using Newton iterations:

$$x^k = x^{k-1} - (df/dx)^{-1}\, f(x^{k-1}, t) \qquad k = 1, 2, \ldots$$

where upperscripts refer to iteration numbers. According to the dishonest Newton scheme, the Jacobian (a scalar in this case) is evaluated at $t = 0$ and kept constant. Thus, we have $df/dx = 1$. The first two simulation steps are as follows:

▷ passing from $t = 0$ to $t = 2$: the first Newton mismatch is $f(x_0, 2) = 2$ and the first Newton iteration yields: $x^1 = 3 - (1 \times 2) = 1$. Since $f(x^1, 2) = 0$, the iterations end up and the solution at $t = 2$ is $x_2 = 1$;

▷ passing from $t = 2$ to $t = 4$: the first Newton mismatch is $f(x_2, 4) = 2$ and the first Newton iteration yields: $x^1 = 1 - (1 \times 2) = -1$. The second mismatch is $f(x^1, 4) = -9$ and the second iteration gives: $x^2 = -1 - (1 \times -9) = 8$. The third mismatch is $f(x^2, 4) = 9$. Clearly, the iterations do not converge; simulation cannot proceed.

Divergence occurs because the Jacobian should be updated each time x changes sign. However, in the dishonest Newton scheme this option is not considered.

In fact, the need to refactorize the Jacobian comes from the non-differentiable nature of the $\min$ function in (3). To avoid this, the model should be rewritten in terms of two flows with a jump according to the sign of x:

$$f_A(x,t) \stackrel{\text{def}}{=} \quad 0 = t - 3 + x \quad \text{if } x > 0$$

$$f_B(x,t) \stackrel{\text{def}}{=} \quad 0 = t - 3 + 10\,x \quad \text{if } x \leq 0$$

At $t = 0$, $x_0 > 0$ and the flow is f_A, with the corresponding Jacobian $df_A/dx = 1$. The first two simulation steps of this hybrid system are as follows:

▷ passing from $t = 0$ to $t = 2$: same as above, leading to $x_2 = 1$;

▷ passing from $t = 2$ to $t = 4$: the first Newton mismatch is $f_A(x_2, 4) = 2$ and the first Newton iteration yields: $x^1 = 1 - (1 \times 2) = -1$. Since $f_A(x^1, 4) = 0$, the iterations end up;

▷ testing jump condition at $t = 4$: since $x^1 < 0$, the condition to jump to flow f_B is satisfied. The Jacobian is updated to $df_B/dx = 10$;

▷ redoing the step from $t = 2$ to $t = 4$: the first Newton mismatch is $f_B(x_2, 4) = 11$ and the first Newton iteration yields: $x^1 = 1 - (10^{-1} \times 11) = -0.1$. Since $f_B(x^1, 4) = 0$, the iterations ends up, and since the jump condition is not satisfied, the solution at $t = 4$ is $x_4 = -0.1$.

To summarize, having replaced the non-differentiable flow by two differentiable flows and a jump condition, it is possible to overstep the event that occurs at $t = 3$ and obtain the sought approximate evolution.

4.2 Example 2

Consider the system in Fig. 4 involving an integrator with a non-windup limit.

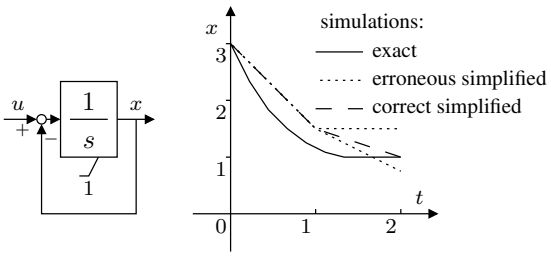

Figure 4: Example 2: block diagram and simulation results

A standard mathematical formulation of the system is:

$$f_A(x) \stackrel{\text{def}}{=} \quad \dot{x} = -x + u \qquad \text{if } x > 1 \text{ or } (x = 1 \text{ and } u - x > 0) \tag{4}$$

$$f_B(x) \stackrel{\text{def}}{=} \quad \dot{x} = 0 \qquad \text{if } x = 1 \text{ and } u - x \leq 0 \tag{5}$$

Starting from the initial condition $x_0 = 3$ at $t = 0$, we assume that u is the step function with $u = 3$ for $t < 0$ and $u = 0$ for $t \geq 0$. Since $x_0 > 1$, the initial flow is f_A.

Consider a simplified simulation with step size $h = 1$ s, using Backward Euler method (the simplest method with stiff-decay property). For the step from $t = 0$ to $t = 1$, we have:

$$x_1 = x_0 + h\dot{x_1} = x_0 - hx_1 \tag{6}$$

In this example, the emphasis is not on Newton iterations, but on the solution reached. The latter is easily obtained from (6) as:

$$x_1 = \frac{x_0}{1+h} = 1.5$$

Since no jump takes place, we proceed with the integration from $t = 1$ to $t = 2$. The solution reached is:

$$x_2 = \frac{x_1}{1+h} = 0.75$$

At this point we face the problem that x has left its domain of existence implicitly defined by (4, 5), which is the interval $[1\ +\infty[$. In principle, the simulation cannot proceed since the jump conditions have not been written for $x < 1$, which is not supposed to occur when Eqs. (4, 5) are solved accurately.

Thus, the jump conditions have to be adjusted to accommodate values of x lying temporarily outside its domain of existence. To this purpose, let us tentatively replace (5) by:

$$f_B(x) \stackrel{\text{def}}{=} \quad \dot{x} = 0 \qquad \text{if } x < 1 \text{ or } (x = 1 \text{ and } u - x \leq 0) \tag{7}$$

in which the jump condition accommodates values of x smaller than 1.

Pursuing our example, the condition for jumping from f_A to f_B is now satisfied. Hence, the step from $t = 1$ to $t = 2$ is repeated, restarting from $x_1 = 1.5$ and following flow f_B. The Backward Euler formula yields:

$$x_2 = x_1 + h\dot{x_1} = 1.5 + (1 \times 0) = 1.5$$

This is obviously wrong. Instead of bringing x back to its limit 1, the flow f_B forces x to remain at its value at $t = 1$.

In fact, the model (7) is correct if at the instant of jumping from f_A to f_B, the value of x is exactly equal to 1. Again, the formulation is not suited to integration steps not falling on the event times [1].

A more general solution to the above problem consists in writing the flows as follows:

$$f_A(x) \stackrel{\text{def}}{=} \quad \dot{x} = -x + u \qquad \text{if } x > 1 \text{ or } (x = 1 \text{ and } u - x > 0) \tag{8}$$

$$f_B(x) \stackrel{\text{def}}{=} \quad 0 = x - 1 \qquad \text{if } x < 1 \text{ or } (x = 1 \text{ and } u - x \leq 0) \tag{9}$$

in which Eq. (9) implicitly assigns x to its limit. Note that, when x reaches its limit, the equation changes from differential to algebraic, a feature that was captured in (1). Note that this technique also applies to the case where the limit is not a constant, but a function of time and/or a function of another state.

Let us illustrate how the above formulation works:
▷ passing from $t = 1$ to $t = 2$: same as above, leading to $x_2 = 0.75$;
▷ testing jump condition at $t = 2$: since $x_2 < 1$, the jump to flow f_B takes place, with the Jacobian updated to $df_B/dx = 1$;
▷ redoing the step from $t = 1$ to $t = 2$: starting back from $x_1 = 1.5$, the first Newton mismatch is $f(x_1) = 0.5$ and the first Newton iteration yields: $x^1 = 1.5 - (1 \times 0.5) = 1$. Since $f_B(x^1) = 0$, the iterations end up, and since no new jump condition is satisfied, the solution at $t = 2$ is $x_2 = 1$. As expected, the limit is satisfied.

4.3 Example 3

Consider the small system in Fig. 5 involving an integrator with limits on the rate of change of x. It can be modelled by:

$$f_A(x) \stackrel{\text{def}}{=} \quad \dot{x} = -1 \quad \text{if} \quad u - x \leq -1 \tag{10}$$

$$f_B(x) \stackrel{\text{def}}{=} \quad \dot{x} = -x + u \quad \text{if} \quad -1 < u - x < 1 \tag{11}$$

$$f_C(x) \stackrel{\text{def}}{=} \quad \dot{x} = 1 \quad \text{if} \quad u - x \geq 1 \tag{12}$$

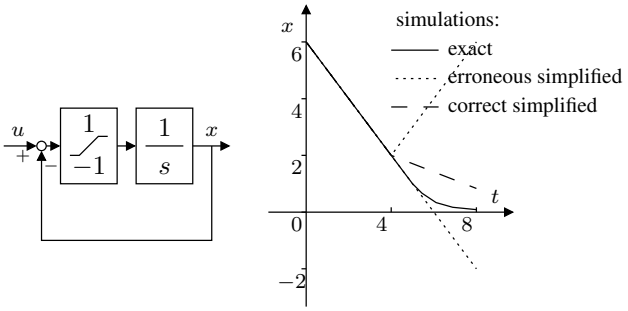

Figure 5: Example 3: block diagram and simulation results

[1] To overcome this problem one could force the state x to its limit and redo the integration step: this will adjust all other variables accordingly. This technique, however, has its own limitations, especially with varying limit implementation.

Starting from the initial condition $x_0 = 6$ at $t = 0$, we assume that u is the step function with $u = 6$ for $t < 0$ and $u = 0$ for $t \geq 0$. The initial flow is f_A.

Consider a simplified simulation with step size $h = 4$ s, still using Backward Euler method. From $t = 0$ to $t = 4$ we have:

$$x_4 = x_0 + h\dot{x_4} = 6 + (4 \times (-1)) = 2$$

No jump takes place and the simulation proceeds to $t = 8$. Using the same formula:

$$x_8^{(1)} = x_4 + h\dot{x_8} = 2 + (4 \times (-1)) = -2$$

At this point, the condition to jump from f_A to f_C is satisfied. Hence, the step is repeated with the new flow, which yields:

$$x_8^{(2)} = x_4 + h\dot{x_8} = 2 + (4 \times 1) = 6$$

for which the condition to revert to flow f_A is satisfied. Therefrom, the solutions will endlessly oscillate between the above two values:

$$x_8^{(3)} = x_8^{(1)} = -2 \quad x_8^{(4)} = x_8^{(2)} = 6 \quad \ldots$$

The large step size causes the flow to oscillate between f_A and f_C without a chance of using the correct flow, f_B.

The solution consists in preventing a direct transition from f_A to f_C, and conversely. This is conveniently represented in the state transition graph of Fig. 6. In the latter, there is no arrow between f_A and f_C; all the transitions have to go through f_B.

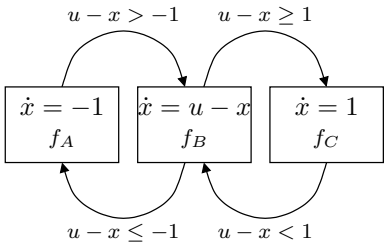

Figure 6: Example 3: state transition graph for a correct simulation

Adhering to these rules, the integration from $t = 4$ to $t = 8$ is as follows:

▷ passing from $t = 4$ to $t = 8$: same as above, leading to $x_8^{(1)} = -2$;

▷ testing jump condition at $t = 8$: since $u - x_8^{(1)} > -1$, the jump to flow f_B takes place

▷ redoing the step from $t = 4$ to $t = 8$: starting back from $x_4 = 2$, the Backward Euler formula gives:

$$x_8 = x_4 + h\dot{x_8} = x_4 - hx_8$$

whose solution is

$$x_8^{(2)} = \frac{x_4}{1+h} = 0.4$$

At this point, no new jump takes place and the simulation successfully proceeds with flow f_B.

5 When everything else fails...

5.1 Safeguards against unsolvable situations

There is no guarantee that the safeguards described in the previous section suffice to avoid numerical difficulties when overstepping an event with large steps. The problem most likely to remain unsolved, even after adjusting the flow equations and/or the jump conditions, is the cycling between flows, tackled in Example 3.

In order the simulation to proceed, a first option consists in leaving the time step with the last available solution $\mathbf{x}^{(j)}$, once the number of flow jumps reaches a maximum, and proceeding with the next step. We found in many cases that the problem disappears at the next time step. This temporary discrepancy may be acceptable in the context of simplified simulation. However, there is no guarantee that the problem will disappear.

In so far as the problem stems from severely overstepped events, another option is to reduce the step size h until the problem is no longer met. h is first reduced by a factor k: $h_{new} = h/k$ (our simulations have shown that $k = 2$ is a convenient choice). If no event cycling takes place any longer, the step is accepted; otherwise further reductions are applied until the problem is solved. Then, the original step size is restored and the simulation proceeds. The following example illustrates the benefit of step size reduction.

5.2 Example 4

Consider a simple system characterized by:

$$f_A(x) \stackrel{\text{def}}{=} \quad \dot{x} = -2 \quad \text{if} \quad x \geq 1.5 \tag{13}$$

$$f_B(x) \stackrel{\text{def}}{=} \quad \dot{x} = -1 \quad \text{if} \quad x < 1.5 \tag{14}$$

with the initial condition: $x_0 = 8$.

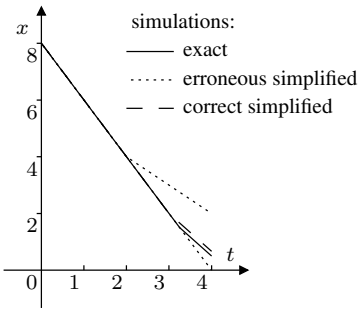

Figure 7: Example 4: simulation results

We perform a simplified simulation with $h = 2$ s. The step from $t = 0$ to $t = 2$ with the initial flow f_A yields:

$$x_2 = x_0 + h\dot{x_2} = 8 + (2 \times (-2)) = 4$$

where no jump takes place. The next step gives:

$$x_4^{(1)} = x_2 + h\dot{x_4} = 4 + (2 \times (-2)) = 0$$

where the condition for jumping from f_A to f_B is satisfied. Thus, the step is repeated, which yields now:

$$x_4^{(2)} = 4 + (2 \times (-1)) = 2$$

where the condition for jumping back to f_A is satisfied ! The simulation is trapped into a cycle from one flow to the other, a situation where the previously discussed precautions cannot help.

It can be easily shown that, with $h = 2$, the cycle problem takes place if the x computed with flow f_A falls in the interval $[-0.5\ 1.5[$. If so, the condition is satisfied for a jump to f_B, which brings x to a value for which the condition for the reverse jump is satisfied. The cycle takes place.

To avoid this problem, the step size is reduced as previously suggested. We tentatively use a halved step size, i.e. $h = 1$ s. Passing from $t = 2$ to $t = 3$ gives:

$$x_3 = x_2 + h\dot{x_3} = 4 + (1 \times (-2)) = 2$$

where no jump takes place. Thus, simulation proceeds from $t = 3$ to $t = 4$ and yields:

$$x_4^{(1)} = x_3 + h\dot{x_4} = 2 + (1 \times (-2)) = 0$$

where the condition for jumping from f_A to f_B is satisfied. A repetition of this last step with flow f_B yields:

$$x_4^{(2)} = x_3 + h\dot{x_4} = 2 + (1 \times (-1)) = 1$$

where no jump takes place. The simulation can proceed normally with flow f_B.

With the reduced step $h = 1$ s, it can be shown that the cycle problem takes place if the x computed with flow f_A falls in the interval $[0.5\ 1.5[$. This is not the case for x_3 nor for $x_4^{(1)}$, which explains why simulation is successful. In fact, the width of the interval of problematic x values has shrunk with the step size h, giving a chance to jump over that interval.

Note that in this example an increase in step size h could also avoid falling in the interval of problematic x values as can be checked, for instance, by integrating from x_2 with a step size $h = 4$ s. However, an increase in h is not desirable in terms of accuracy of solution and convergence of Newton iterations in nonlinear systems.

6 A power system example

We illustrate hereafter how the proposed algorithm deals with a typical hybrid power system model.

The case has been obtained with the Nordic32 test system, documented in [12] and whose voltage evolution at one bus has been already shown in Fig. 1. At $t = 1$ s, the transmission line $4032 - 4044$ is tripped. This causes the neighbouring voltage-controlled generator g14 to operate above its permanent field current limit. Figure 8 shows the evolution of this field current obtained by detailed simulation. The limitation of the field current, caused by the OverExcitation Limiter (OEL), is clearly seen at $t = 50$ s. The same figure shows the evolution computed by simplified simulation, using a default step size of 0.5 s. As expected, the initial oscillations are not reproduced but the overall evolution is correctly rendered.

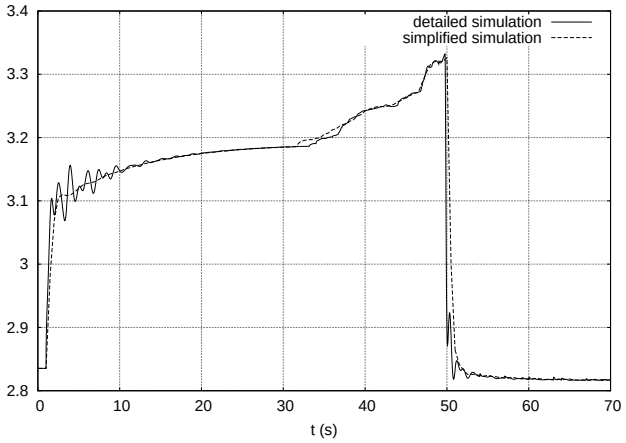

Figure 8: Evolution of i_{fd} of generator g14 (pu)

The block-diagram of the OEL is shown in Fig. 9.

In normal operating conditions the field current i_{fd} is much lower than its permanent limit i_{fd}^{lim} and the output of block 1 is -1. This keeps the integrator of block 2 at its lower limit L_1, which is negative. It results that the switch in block 3 remains in the lower position and passes the input $V^o - V$. The minimum gate receives two equal inputs, and we assume it selects w, the output of block 3.

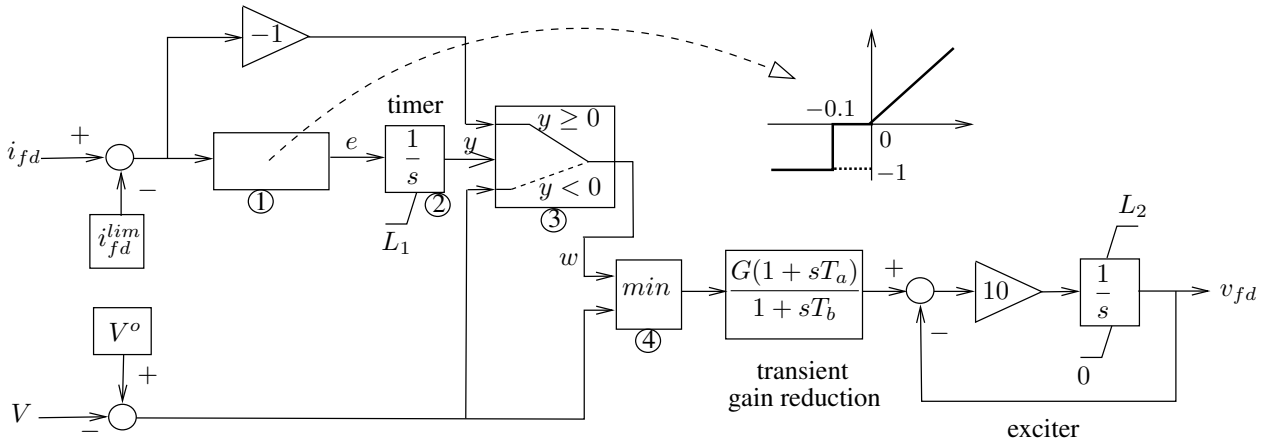

Figure 9: Block diagram of voltage regulator, field current limiter and exciter

When the machine operates with $i_{fd} > i_{fd}^{lim}$, the output y of the integrator block 2 grows, until it reaches a positive value, causing block 3 to switch. Thus, block 2 acts as a timer and provides inverse-time characteristic to the OEL. After block 3 has switched, the inputs of the minimum gate are $V^o - V > 0$ and $w = i_{fd}^{lim} - i_{fd} < 0$, respectively. Hence, the gate keeps on selecting w.

The purpose of this minimum gate is to reset the machine under voltage control whenever operating conditions improve to the point that $V^o - V$ becomes negative. However, in the following example, no such reset takes place and the minimum gate selects its w input throughout the whole simulation.

After the line outage, i_{fd} starts increasing and becomes larger than i_{fd}^{lim} at $t = 1.5$ s. The step from $t = 1$ to $t = 1.5$ is thus retaken to account for the jump from flow $e = 0$ to flow $e = i_{fd} - i_{fd}^{lim}$ in block 1. The output of block 1 becomes positive, causing a jump in block 2, corresponding to y leaving its lower limit L_1.

Variable y keeps on increasing until it becomes positive, causing the switch in block 3 to change, i.e. the machine to switch from voltage to field current control in between $t = 49.5$ and $t = 50$. A zoom on the evolution of i_{fd} and y is given in Fig. 10.

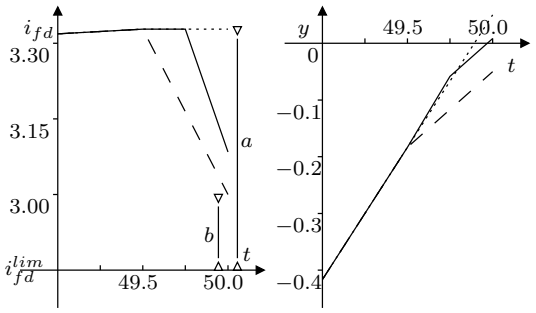

Figure 10: Detailed view of iterations when the OEL becomes active

Here is a detailed analysis of this step:

▷ at the beginning of the step, $y < 0$, block 3 is selecting its input $V^o - V$; the step converges to a positive y value at $t = 50$, triggering a jump in block 3. This is shown with dotted lines in Fig. 10;

▷ the step is restarted with block 3 passing $i_{fd}^{lim} - i_{fd}$. However, when passing through the exciter and the synchronous machine this signal has such a pronounced effect on the system that a value i_{fd} close to i_{fd}^{lim} is retrofitted to the OEL, which causes the updated value of y at $t = 50$ to get back to negative value. This is shown with dashed lines in Fig. 10. However, this triggers the reverse jump of block 3, and causes the jump handling procedure to endlessly oscillate between voltage and field control.

In fact, the case bears some similarity with Example 4 of Section 5.2. The variable y that controls block 3, when integrated with "large" derivative $i_{fd} - i_{fd}^{lim} = a$ (see Fig.10) triggers the jump to the flow with "small" derivative $i_{fd} - i_{fd}^{lim} = b$, where $b < a$ due to the above mentioned fast retroaction. Conversely, when integrated with the "small" derivative b, it triggers the jump to the flow with "large" derivative a.

As indicated in Section 5, two options are available to proceed with the simulation: (i) break the cycle and accept the last available solution, or (ii) reduce the step to have more chances to avoid cycling. The first option yields satisfactory convergence but delays the OEL action by one step. The result of second option is shown with solid line in Fig. 10. The step size has been halved after meeting the problem at $t = 49.5$. The simulation passes from $t = 49.50$ to $t = 49.75$ without jump. Then the machine correctly switches under field limit in between $t = 49.75$ and $t = 50.00$. At 49.75 s, the value y escapes the region of non-existence of solutions and can thus converge successfully.

7 Conclusion

In this paper algorithms for simplified time simulation of power system dynamics are considered. The objective is to obtain approximate time responses by "filtering out" some fast dynamics, using a relatively large time step size on the original (i.e. non simplified) model. Stiff-decay integration methods (such as backward differentiation formulae) allow using a step size larger than the fast, neglected dynamics. However, attention must be paid to the overstepping of time instants where discrete events cause the differential/algebraic model to jump from one flow to another.

In the approach detailed in this paper, no attempt is made to identify the correct event times. Instead, jump conditions are checked after the step has been solved with the previous system equations and, if jump conditions are satisfied, the step is repeated with the new prevailing flow. The procedure may be repeated if new jump conditions are satisfied.

Through simple examples, the paper emphasizes some precautions to be taken in the formulation of the jump conditions and of the new prevailing flow, as well as in the handling of transitions from one flow to another.

In spite of those precautions, it may happen that the proposed procedure leads to cycling between flows. In such a case, one option is to reduce the step size until the cycling problem is resolved, and progressively restore it to the original value.

The paper provides an illustrative power system example, in which the above procedure allows simulating the voltage unstable long-term evolution with a step size of 0.5 s, temporarily reduced to solve a flow cycling problem when an overexcitation limiter comes into action.

The procedure has been implemented in a production-grade program and is being tested on complex controller models. So far, it has been found that many discrete events can be solved by the proposed procedure without reducing the step size, while a temporary reduction of the latter allows solving the difficult cases.

Acknowledgement

This work was performed in the context of the PEGASE project [8] funded by European Community's 7th Framework Programme (grant agreement No. 211407).

REFERENCES

[1] T. Van Cutsem, C. Vournas, *Voltage Stability of Electric Power Systems*, Boston, Kluwer Academic Publishers (now Springer), 1998.

[2] D. Fabozzi, T. Van Cutsem, "Simplified time-domain simulation of detailed long-term dynamic models", Proc. IEEE PES General Meeting, Calgary (Canada), July 2009. Available at http://ieeexplore.ieee.org (DOI 10.1109/PES.2009.5275463) and http://hdl.handle.net/2268/9524

[3] U.M. Ascher, L.R. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, Society for Industrial and Applied Mathematics (SIAM), 1998

[4] J.Y. Astic, A. Bihain, M. Jerosolimski, "The mixed Adams - BDF variable step size algorithm to simulate transient and long term phenomena in power systems", *IEEE Trans. on Power Systems*, Vol. 9, No. 2, May 1994

[5] B. Dembart, A.M. Eirsman, E.G. Cate, M.A. Epton and H. Dommel, "Power System Dynamic Analysis - Phase I. Final Report EPRI EL-484", July 1977

[6] I.A. Hiskens, M.A. Pai, "Trajectory sensitivity analysis of hybrid systems", *IEEE Trans. on Circuits and Systems I*, Vol. 47, No. 2, Feb. 1994

[7] P. Kundur, *Power System Stability and Control*, Mc Graw Hill, EPRI Power System Engineering Series, 1994

[8] [Online], Available at http://www.fp7-pegase.eu/

[9] J. Lygeros, "Lecture Notes on Hybrid Systems", ENSIETA, 2-6/2, 2004

[10] F.E. Cellier, E. Kofman, *Continuous System Simulation*, Springer-Verlag, New York (USA), 2006

[11] D. Fabozzi, T. Van Cutsem, "Localization and latency concepts applied to time simulation of large power systems", Proc. 2010 IREP Symposium, Buzios (Brazil), August 2010. Available at http://ieeexplore.ieee.org (DOI 10.1109/IREP.2010.5563287) and http://hdl.handle.net/2268/65833

[12] M. Glavic, T. Van Cutsem, "Wide-Area Detection of Voltage Instability From Synchronized Phasor Measurements. Part II: Simulation Results," *IEEE Trans. Power Syst.*, Vol. 24, No. 3, pp. 1417-1425, Aug. 2009.