Methods for Incorporating Digital Controllers in Power System Dynamic Simulations

Mehran Jafari, Petros Aristidou
Dept. of Electrical Eng., Computer Eng., & Informatics
Cyprus University of Technology, Limassol, Cyprus
mm.jafari@edu.cut.ac.cy

Gautier Bureau, Marco Chiaramello
Adrien Guironnet, Patrick Panciatici
Réseau de Transport d'Électricité (RTE)
Paris, France

***Abstract*—Digital controllers are involved in every aspect of today's power systems, from local device-level to wide-area supervisory applications. However, the existing dynamic simulation tools are still struggling to capture an accurate response of such systems with adequate computational performance. The reason is that digital controllers introduce numerous discontinuities to the simulation and handling them is computationally expensive, thus forcing the users to compromise between accuracy and simulation performance. This paper investigates different traditional approaches for incorporating digital controllers in power system dynamic simulations and compares them with new interpolation-based approaches in terms of simplicity, accuracy, and performance.**

***Index Terms*—Digital controller, power system dynamics, interpolation, Jacobian.**

I. Introduction

The number of modern digital components has substantially increased in power systems in the last few decades. Controllers are one of the components that are changing rapidly, either with the introduction of cutting-edge digital technologies or by replacing the old analog ones with digital equivalents. The discrete nature of digital controllers introduces discontinuities in the set of differential-algebraic equations (DAEs) describing the dynamics of power systems. These discontinuities can be categorized as state or time events [1].

State events are referred to as discontinuities happening due to a change in the equations of the system at unknown times, e.g. when a controller limit is reached and one or more equations must be modified. On the other hand, time events happen at specific times during the simulation, e.g. every time a digital controller samples and acts. While the simulation of systems containing state events requires detecting and locating the discontinuity, time events are exempt from this process as the exact time of the event is known beforehand.

The accurate treatment of both types of events requires stopping the simulation, reducing the time step to "land" on the event's time, updating the equations and variables, calculating a new set of initial values, and then resuming the simulation [1]. The solution of this hybrid DAE system [2] requires formulating the Jacobian of the simulated system and updating it after each discrete event. In this approach, the overall computation burden is heavy since the time steps are limited to the difference between the sampling times of the various digital controllers [3].

Currently, the most used approach to handle digital controllers in power system dynamic simulations is to approximate them with their analog equivalents [4]. In this way, all the time events are ignored, and the controller equations are integrated alongside the system DAEs, leading to an increased system Jacobian that incorporates the controller equations. The benefit of this approach is the ability to use variable time-step simulation algorithms, since the controller equations are continuous without discrete time-events. However, besides the inaccuracy introduced by this approximation, the approach is becoming increasingly difficult to follow since modern digital controllers involve complicated algorithms and non-equation-based models, such as model predictive controllers (MPCs) or AI-driven controllers. In addition, using the continuous equivalent controller may lead to complex, ill-defined behaviours [5].

Another approach to treat the discrete events (both state and time) introduced by digital controllers is by shifting all the discontinuities within a time-step to the end of the time-step without reducing the time-step [6]. Although this method is computationally inexpensive, it leads to inaccuracies and situations like cycling between states [7].

Finally, the interpolation-based method (IBM) proposed in [8], provides an accurate and computationally fast method to integrate the digital controllers in large time steps over multiple time events without reducing the time step. This is made possible by interpolating the state variables of the system at each digital controller sampling time and including the controller output in the DAEs model. In this way, the outputs of the controllers are estimated and corrected at each Newton iteration. This method treats the digital controller as an input-output black-box model, thus allowing to incorporate non-equation-based controllers. Including the controller response to the Jacobian of the system, however, can be challenging since the size of the Jacobian varies in each time step depending on the number of controller actions within the time step. Moreover, the digital controller output during the time step is piece-wise constant between two sampling times based on a Zero-Order-Hold (ZOH) approach.

This paper investigates different approaches for incorporating digital controllers in power system dynamic simulation and presents alternative options, making the user able to reach a compromise between simplicity, accuracy, and performance. The highlights of the paper are as follows:

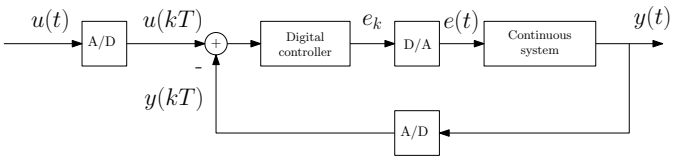

Fig. 1. A continuous system under control by a digital controller scheme

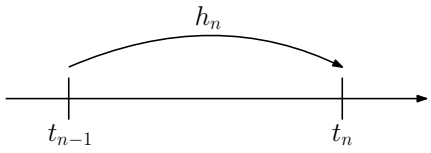

Fig. 2. Schematic of integration using analog treatment method

- Reviewing four different methods allowing the numerical simulation of power systems with digital controllers, and comparing them using case studies in terms of simplicity, accuracy, and performance.
- Comparing alternative, interpolation-based methods proposed by the authors.

The rest of the paper is organized as follows. In Section II, the different methodologies for the numerical simulation of digital controllers are briefly reviewed. A discussion of the Jacobian structure of the system for the different methods is conducted in Section III. Section IV provides numerical case studies to showcase the performance of each approach. Finally, conclusions are drawn in Section V.

II. Methodologies

To discuss different methods, a simple system controlled by a digital controller is considered, as shown in Fig. 1. Power systems are usually modeled with a set of DAEs and a dynamic simulation of the system consists of the solution of the DAE Initial Value Problem (IVP) over a specific time horizon. The system equations are as follows:

$$\begin{aligned} \mathbf{0} &= \boldsymbol{F}(\dot{\boldsymbol{y}}(t), \boldsymbol{y}(t), e(t)) \\ \boldsymbol{y}(0) &= \boldsymbol{y}_0,\ e(0) = e_0 \end{aligned} \tag{1}$$

where $\boldsymbol{y}(t)$ is the vector of differential-algebraic variables of the system, and $e(t)$ is the continuous output of the controller fed to the system.

The controller model can be generally formulated as:

$$e_k = \zeta(e_{k-1}, \boldsymbol{y}(kT)) \tag{2}$$

where ζ is the controller function based on the previous output of the controller e_{k-1} and the quantized state variables under monitor by the controller $\boldsymbol{y}(kT)$ at the k-th sampling time with period T.

For the solution of the problem DAE IVP, (1) is discretized using a numerical integration method and the discretized algebraic equations are solved for each time-step using a Newton method [9]. Considering the discrete time steps t_n, the step size is given as $h_n = t_n - t_{n-1}$, and the solution of the states at each time-step $\boldsymbol{y}(t_n) = \boldsymbol{y}_n$ can be obtained from the following residual function:

$$\boldsymbol{g}(\boldsymbol{y}(t_n), e(t_n)) = 0 \tag{3}$$

This set of equations ignores state events and only focuses on time events. It should be noted that in the case of digital controllers, any state events that take place within a digital controller are transformed to time events. For instance, if a state variable reaches a limit but the limit is monitored and controlled by a digital controller, then the changes to apply the limit will take place at the next digital control action. In the case of other state events, we assume that they are handled using a zero-crossing detection of their guard functions and a time-step reduction [2]. The rest of this section reviews four different approaches for treating the time events stemming from the digital controller sampling actions.

A. Analog treatment method

In the analog treatment method (ATM), the digital control behavior of (2) is approximated with a continuous equivalent DAE model:

$$\mathbf{0} = \boldsymbol{\zeta}'(e(t), \boldsymbol{y}(t)) \tag{4}$$

Then, the controller DAEs are solved alongside (1) for the time step t_n, as is shown in Fig. 2. In other words, the discrete nature of the digital controllers is ignored, and no special treatment is performed for the discontinuities [4]. This method is widely used since it is simple and fast. However, it introduces inaccuracies if the sampling rate of the controller is different from the time step size. Moreover, the simulation of non-equation-based controllers is not possible using ATM.

B. Step reduction method

The step reduction method (SRM) handles every time event by reducing the time step h_n to "land" on the discontinuity [1], calculating the controller's output e_k, then integrating the system's equation using $e(t) = e_k$ as an input. This means that (2) is solved prior to (1) and $e(t)$ is considered constant throughout the time-step integration process.

This method is the most accurate (closer to reality). However, it is computationally heavy since the maximum time step size is limited to the sampling rate of the controller. As shown in Fig. 3, the time step h_n is denied, and instead, the new time step with the size $h_n^{new} = t_{n,1} - t_{n-1}$ is selected, where $t_{n,1}$ denotes the first sampling of the controller within the time step h_n, and p_n is the last sampling within the same time step.

C. Simplified simulation method

Contrary to SRM, which reduces the simulation time step to match the controller sampling period, the simplified simulation method (SSM) proposed in [7] "shifts" the sampling of the controller forward to match the time step. Or, more accurately, it shifts the control action of the digital controller to coincide with the next simulation step, as it is illustrated in Fig. 4. The time-step is then repeated until no further control actions are detected within the time step. In this approach, the system

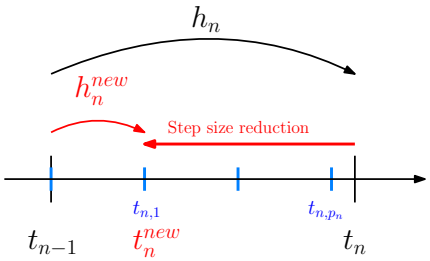

Fig. 3. Schematic of integration using step reduction method. The black vertical lines denote the simulation time steps, while the light-blue vertical lines denote the digital controller sampling period.

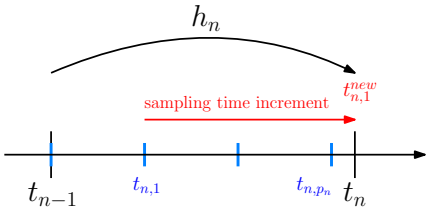

Fig. 4. Schematic of integration using simplified simulation method

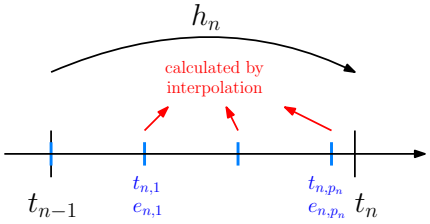

Fig. 5. Schematic of integration using interpolation-based method

equations (1) are first solved to acquire $\boldsymbol{y}_n$ and then (2) to compute e_k, before recomputing the time step.

The computational performance of this method is better than SRM but the accuracy suffers. An important drawback of this method is that in the case of multiple time events within a time step, only one is considered, and the rest are ignored.

D. Interpolation-based method

Unlike the previous methods, the Interpolation-based method (IBM) does not reduce the simulation time-step, and computes all the controller sampling actions in every time-step. It achieves this by calculating the controller outputs e_k by sampling its system state variables between t_{n-1} and t_n [8]. This approach is shown in Fig. 5.

To describe it better, let's define the residual function to be solved for time t_n denoted by $\boldsymbol{g}$ as:

$$\boldsymbol{g}(\boldsymbol{y}_n, e_{n,p_n}) = 0 \tag{5}$$

The last output of the controller in the time step e_{n,p_n} depends on the samplings before it that can be formulated as:

$$e_{n,g} = \zeta(e_{n,g-1}, x_{n,g}, t_{n,g}) \tag{6}$$

where $x_{n,g}$ denotes the interpolated value of the controller's desired system's state variable for the g-th controller sampling at time $t_{n,g}$. It should be noted that x is a subset of y that is being observed by the controller, which can be calculated using an interpolation polynomial $w_n^{(m)}$ for each Newton iteration m:

$$\boldsymbol{x}_{n,g}^{(m)} = \boldsymbol{w}_n^{(m)}(t_{n,g}), \quad \forall g \in [1, p_n] \tag{7}$$

For example, a second-order Taylor expansion or Adams family-based interpolator can be used for $\boldsymbol{w}_n$.

By adding (6) to (1) and solving them together, the controller output is corrected at each Newton iteration. To do that, an expanded state variables vector can be defined as follows:

$$\boldsymbol{z}_n = \begin{bmatrix} \boldsymbol{z}_{n,1} \\ \boldsymbol{z}_{n,2} \end{bmatrix} \tag{8}$$

with:

$$\boldsymbol{z}_{n,1} = \boldsymbol{y}_n \tag{9}$$

$$\boldsymbol{z}_{n,2} = \begin{bmatrix} e_{n,1} & e_{n,2} & \dots & e_{n,g} & \dots & e_{n,p_n} \end{bmatrix}^T \tag{10}$$

where $\boldsymbol{z}_{n,1}$ contains the continuous system state variables while $\boldsymbol{z}_{n,2}$ has the controller's outputs for each sampling time $e_{n,g}$.

Therefore, the new residual vector corresponding to the new state variables vector is defined as follows:

$$\tilde{\boldsymbol{g}} = \begin{bmatrix} \tilde{\boldsymbol{g}}_1 \\ \tilde{\boldsymbol{g}}_2 \end{bmatrix} \tag{11}$$

with:

$$\tilde{\boldsymbol{g}}_1(\boldsymbol{z}_n) = \boldsymbol{g}(\boldsymbol{y}_n, e_{n,p_n}) \tag{12}$$

$$\tilde{\boldsymbol{g}}_2(\boldsymbol{z}_n) = \begin{bmatrix} e_{n,1} - \zeta(e_{n,0}, \boldsymbol{x}_{n,1}) \\ \dots \\ e_{n,g} - \zeta(e_{n,g-1}, \boldsymbol{x}_{n,g}) \\ \dots \\ e_{n,p_n} - \zeta(e_{n,p_{n-1}}, \boldsymbol{x}_{n,p_n}) \end{bmatrix} \tag{13}$$

where $\tilde{\boldsymbol{g}}_1$ contains the continuous system residuals while $\tilde{\boldsymbol{g}}_2$ has the controller's outputs residuals for each sampling time.

Finally, the Newton iteration to be solved is:

$$\boldsymbol{J}_n^{(m)}(\boldsymbol{z}_n^{(m+1)} - \boldsymbol{z}_n^{(m)}) = -\tilde{\boldsymbol{g}}(\boldsymbol{z}_n^{(m)}) \tag{14}$$

where $\boldsymbol{J}_n^{(m)}$ is the Jacobian matrix of the extended system at the m-th Newton iteration, given by:

$$\boldsymbol{J}_n^{(m)} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} \frac{\partial \tilde{\boldsymbol{g}}_1}{\partial \boldsymbol{z}_{n,1}} & \frac{\partial \tilde{\boldsymbol{g}}_1}{\partial \boldsymbol{z}_{n,2}} \\ \frac{\partial \tilde{\boldsymbol{g}}_2}{\partial \boldsymbol{z}_{n,1}} & \frac{\partial \tilde{\boldsymbol{g}}_2}{\partial \boldsymbol{z}_{n,2}} \end{bmatrix} \tag{15}$$

Sub-matrices A, B, C, and D, correspond respectively to the controlled system Jacobian, the impact of the controllers on the controlled system, the impact of the system on the controllers, and the controller system Jacobian.

III. Jacobian matrix formulations

Let's assume that the number of continuous system equations of (1) is a and the digital controller equations b.

In the case of the ATM, the size of the Jacobian matrix formed to solve the problem is equal to $(a+b)\times(a+b)$ since all the equations are solved together at the same time-step. The size of the Jacobian matrix is constant throughout the simulation.

For both SRM and SSM, the size of the Jacobian matrix depends only on the number of continuous system equations and is equal to $a \times a$ since the controller outputs are computed before/after the solution of the DAEs for the specific time step and remains constant for all the time steps.

Contrary to the previous methods that have a constant size of Jacobian matrix for the whole simulation, the size of Jacobian matrix for IBM varies for each time step depending on the size of the simulation time step h_n and the number of digital controller samplings p_n falling within it. For example, assuming only one digital controller with one equation ($b=1$), the Jacobian matrix is formulated as $(a+p_n)\times(a+p_n)$.

The increasing size of the Jacobian matrix for each controller action within a time step can lead to simulations that have a Jacobian matrix many times larger than the system under simulation. For example, let's consider a small system modeled with ten DAEs ($a=10$) and 4 controllers ($N_C=4$). Within a certain time step, each has 5 control actions leading to a total of $4\times5=20$ interpolation points. Thus, the size of the Jacobian matrix is equal to 30×30, which is 9 times larger than the controlled system. Moreover, since the number of control actions p_n can change between each time step, the Jacobian matrix needs to be frequently recomputed. To achieve a reasonable performance versus accuracy, a "dishonest" Jacobian approach can be used by simplifying the sub-matrices B, C, or D in (15). A dishonest Newton method maintains the computation of the residuals accurate but uses an approximate Jacobian matrix. This way, the overall computation time decreases while the convergence rate becomes slower [10]. Some examples are shown below and compared in Section IV.

It should be noted that the sub-matrix D is approximated by an identity matrix for all the variations due to the unknown nature of the controllers. In other words, based on (13), D is a bidiagonal band matrix with non-zero elements on the main diagonal and the one below that. However, if we consider the controllers as input-output black-box models, the automatic computation of the lower diagonal is complicated or even impossible. Setting the lower diagonal to zero leads to an identity matrix approximation for D.

A. IBM-A

The most simplified one can be defined as the following:

$$\boldsymbol{J}_n^{(m)} = \begin{bmatrix} \frac{\partial \tilde{\boldsymbol{g}}_1}{\partial \boldsymbol{z}_{n,1}} & \frac{\partial \tilde{\boldsymbol{g}}_1}{\partial \boldsymbol{z}_{n,2}} \approx 0 \\ \frac{\partial \tilde{\boldsymbol{g}}_2}{\partial \boldsymbol{z}_{n,1}} \approx \boldsymbol{0} & \frac{\partial \tilde{\boldsymbol{g}}_2}{\partial \boldsymbol{z}_{n,2}} \approx \boldsymbol{I}_{p_n} \end{bmatrix} \tag{16}$$

where the sub-matrices B and C are replaced by zeros. In other words, the impact of controllers on the system and the impact of the system on the controller is neglected within a Newton iteration. This allows to decouple the two systems and exchange at each Newton method. This, however, might lead to more Newton iterations before convergence.

B. IBM-AB

This approximation maintains sub-matrix C to zero but considers the sub-matrix B, which includes the impact of controller outputs on the system's equations:

$$\boldsymbol{J}_n^{(m)} = \begin{bmatrix} \frac{\partial \tilde{\boldsymbol{g}}_1}{\partial \boldsymbol{z}_{n,1}} & \frac{\partial \tilde{\boldsymbol{g}}_1}{\partial \boldsymbol{z}_{n,2}} \approx \frac{\partial \tilde{\boldsymbol{g}}_1}{\partial e_{n,p_n}} \\ \frac{\partial \tilde{\boldsymbol{g}}_2}{\partial \boldsymbol{z}_{n,1}} \approx \boldsymbol{0} & \frac{\partial \tilde{\boldsymbol{g}}_2}{\partial \boldsymbol{z}_{n,2}} \approx \boldsymbol{I}_{p_n} \end{bmatrix} \tag{17}$$

It is noticeable that in each time step, only the last output of the controller feeds back to the system. Therefore, B is a sparse matrix that has only one non-zero element per input that comes from a controller. This approximation leads to an upper block triangular Jacobian matrix.

C. IBM-AC

This approximation maintains sub-matrix B to zero but considers the sub-matrix C, which has the impact of the system state variables on the controller equations:

$$\boldsymbol{J}_n^{(m)} = \begin{bmatrix} \frac{\partial \tilde{\boldsymbol{g}}_1}{\partial \boldsymbol{z}_{n,1}} & \frac{\partial \tilde{\boldsymbol{g}}_1}{\partial \boldsymbol{z}_{n,2}} \approx 0 \\ \frac{\partial \tilde{\boldsymbol{g}}_2}{\partial \boldsymbol{z}_{n,1}} \approx \frac{\partial \tilde{\boldsymbol{g}}_2}{\partial \boldsymbol{y}_n} & \frac{\partial \tilde{\boldsymbol{g}}_2}{\partial \boldsymbol{z}_{n,2}} \approx \boldsymbol{I}_{p_n} \end{bmatrix} \tag{18}$$

This sub-matrix is also a sparse matrix with non-zero elements for each sampling of the controller per system variable that is monitored. This approximation leads to a lower block triangular Jacobian matrix.

D. IBM-ABC

This variation has both sub-matrices B and C included simultaneously.

IV. Numerical results

In this section, two case studies are considered to benchmark the methods reviewed and the IBM Jacobian approximation variations in terms of simplicity, performance, and accuracy. A variable-step predictor-corrector integration method utilizing a pair of second-order Adams-Bashford and Adams-Moulton is used for solving all the case studies [11], and for the IBM, a second-order Taylor expansion polynomial is used for the interpolation. The same set of system and controller DAEs is solved for all methods except ATM (due to the use of analog equivalents). In addition, a uniform quantization method with 16 bits is considered for A/D and D/A blocks (see Fig. 1). The increasing rate and decreasing rate for adjusting the time steps are equal to 1.25 and 0.5, respectively. Finally, the minimum and maximum values for the time steps are considered 1 ms and 1 s, respectively.

Furthermore, Euclidean distance is used to compare the accuracy of different trajectory results of different methods, as formulated [12]:

$$d(\alpha,\beta) = \sqrt{\sum_{n=1}^{N}(\alpha_i - \beta_i)^2} \tag{19}$$

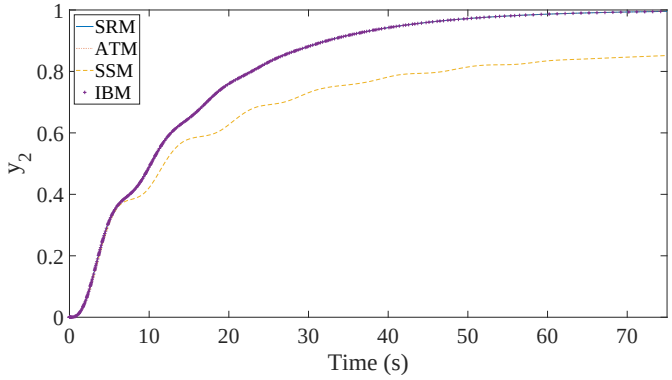

Fig. 6. Simulation results for the methods SRM, ATM, SSM, and IBM-AB for the stable system controlled by the integral controller

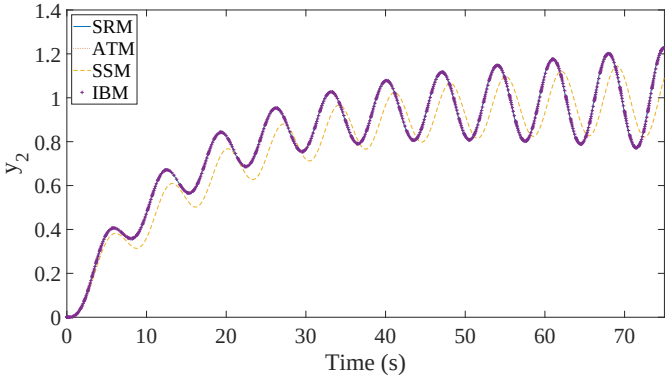

Fig. 7. Simulation results for the methods SRM, ATM, SSM, and IBM-AB for the unstable system controlled by the integral controller

where d denotes the distance between time series of trajectories α and β, and N is the total number of time steps. It should be noted that due to the variable time-step approach, the time-series trajectories might not align on the same points of time. Thus, a first-order linear interpolation is first used to estimate the solution in points between them. All solvers for all methods are implemented in MATLAB 2021 [13].

A. *Two-state system with integral controller*

For the first test system, an integral controller is monitoring a simple continuous system (see Fig. 1) containing two differential equations defined as follows:

$$\begin{bmatrix} \dot{y_1} \\ \dot{y_2} \end{bmatrix} = \begin{bmatrix} a & b \\ -b & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} + \begin{bmatrix} -b \\ 0 \end{bmatrix} e(t) \tag{20}$$

The integral controller is modelled as follows:

$$e_k = e_{k-1} + K_I T(u(kT) - y_2(kT)) \tag{21}$$

where u is the setpoint, the integral gain K_I is set to 0.07, and sampling time T is set to 0.1.

The output of the system y_2 is illustrated for a stable and unstable scenarios (different b values) in Figs. 6 and 7, respectively. It should be pointed out that only IBM-AB is used for this case study. As can be seen, the response of SRM, ATM, and IBM-AB is almost identical, with SSM being the most inaccurate.

Figures 8 and 9 show the time steps taken for the stable and unstable systems, respectively. It can be seen that SRM limits the time steps to the difference between controller sampling times while other methods are able to increase time step size with no limitation. In addition, it is noticeable that ATM has the fastest time step increase. Furthermore, the number of Newton iterations for solving the stable and unstable system and for all the methods is listed in Table I.

B. *Kundur system*

In this section, the Kundur test system [14], illustrated in Fig. 10, is used to better showcase the difference between different methods and test the accuracy and performance of

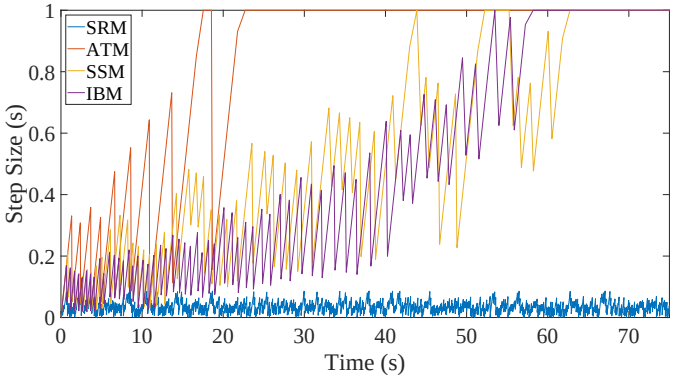

Fig. 8. Step size results for the methods SRM, ATM, SSM, and IBM-AB for the stable system with the Integral controller

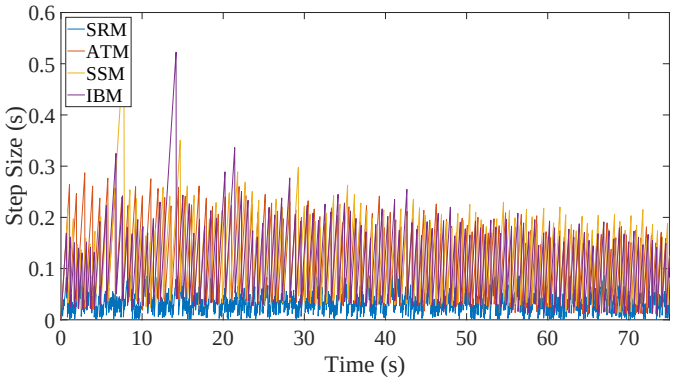

Fig. 9. Step size results for the methods SRM, ATM, SSM, and IBM-AB for the unstable system with the Integral controller

different IBM variations. The test system consists of 11 buses, 4 synchronous generators, and 2 loads. For each machine, one digital exciter and one digital governor are considered as shown in Figs. 11 and 12, respectively [15], [16]. While the controllers are represented using the s-domain block diagrams, their discrete equivalents are used for the simulation of digital controllers as explained in [4]. The difference equations of the

TABLE I
PERFORMANCE COMPARISON BETWEEN SRM, ATM, SSM, AND IBM-AB FOR A SYSTEM CONTAINING ONE INTEGRAL CONTROLLER IN TERMS OF THE NUMBER OF NEWTON ITERATIONS

Method	SRM	ATM	SSM	IBM-AB
Stable case	6338	336	830	1180
Unstable case	6151	2497	2561	2895

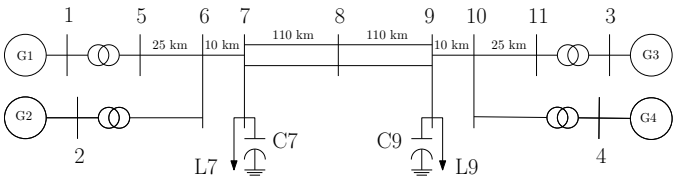

Fig. 10. Schematic of Kundur test system

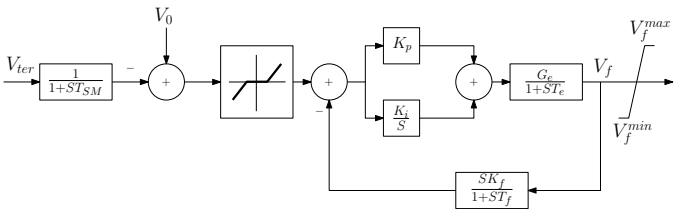

Fig. 11. Schematic of the digital exciter of Kundur test system [15]

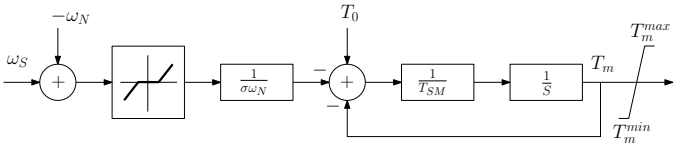

Fig. 12. Schematic of the digital governor of Kundur test system [16]

controllers are obtained using the forward Euler method. The system has 8 digital controllers in total. The sampling time T of the digital controllers is set to 210 ms, 220 ms, 230 ms, and 240 ms for the governors, and 41 ms, 42 ms, 43 ms, and 44 ms for the exciters.

A short circuit on bus 3 for 200 ms is simulated using all four methods. The simulation outputs are shown in Figs. 13, 14, 15, and 16, for the voltage of bus 1, the speed deviation of the third generator, the governor output of the third generator, and the exciter output of the third generator, respectively. An upper limit equal to 2 per unit is considered for the third exciter to assess the accuracy of the methods while facing non-linearity in the controllers, and the results are illustrated in Fig. 16.

As can be seen, ATM has the worst accuracy since it ignores the digital nature of the controller. This is also reflected in Table II which lists the Euclidean distance of the bus 1 voltage and speed deviation of the third generator with respect to SRM as the reference trajectory. It is also not the fastest method anymore which is reflected in Fig. 17 that shows the time steps taken for all four methods. Furthermore, it can be also

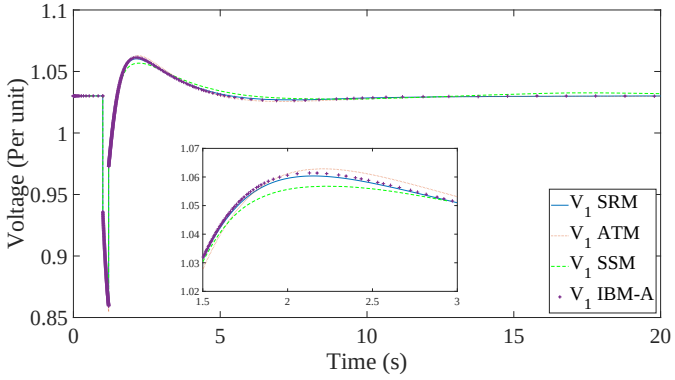

Fig. 13. Voltage of bus 1 of Kundur test system

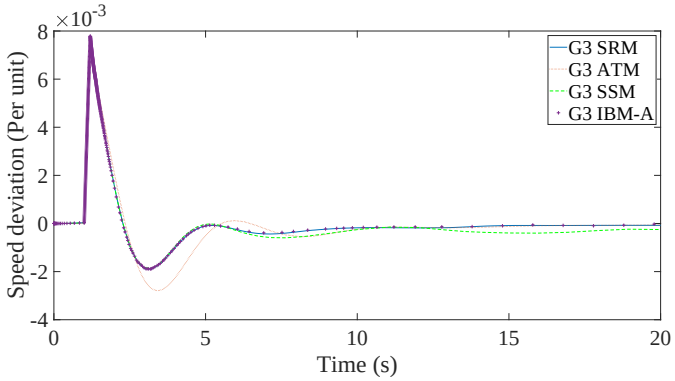

Fig. 14. Speed deviation of the generator 3 of Kundur test system

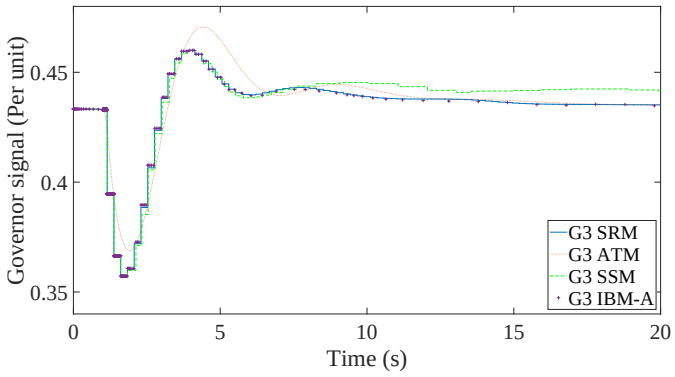

Fig. 15. Governor output of the generator 3 of Kundur test system

seen in Table III which summarizes the performance in terms of the number of Newton iterations, the number of function evaluations, and the average execution time of five repetitive simulations. The reason that ATM is not the fastest method anymore is that the size of the system to be solved is increased by the number of controllers' equations.

SSM has the fastest performance, as was expected since it doesn't change the size of the system Jacobian matrix (unlike IBM) and doesn't reduce the time steps (unlike SRM). On

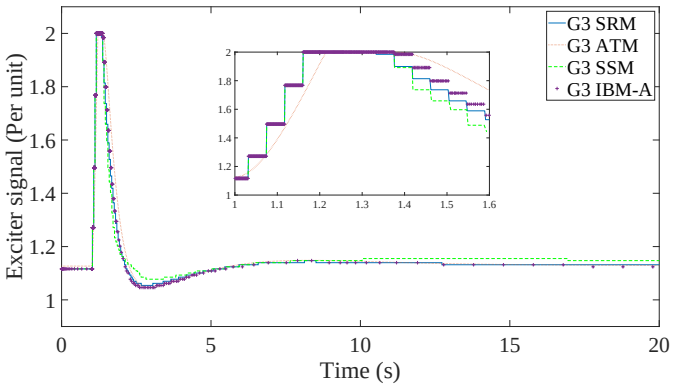

Fig. 16. Exciter output of the generator 3 of Kundur test system

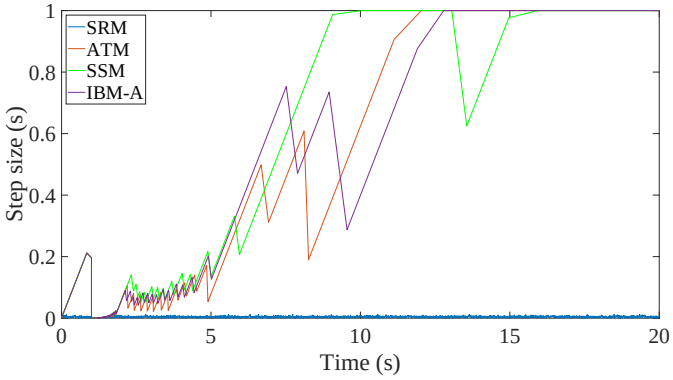

Fig. 17. Step size results for the methods SRM, ATM, SSM, and IBM for the Kundur test system

TABLE II
ACCURACY COMPARISON BETWEEN ATM, SSM, AND IBM WITH RESPECT TO SRM FOR KUNDUR SYSTEM USING EUCLIDEAN DISTANCE

Method	ATM	SSM	IBM-A
$\epsilon(V_1)$	0.1359	0.1160	0.1138
ϵ(Speed deviation 3)	0.007697	0.0009768	0.0004236

TABLE III
PERFORMANCE COMPARISON BETWEEN SRM, ATM, SSM, AND IBM-A FOR KUNDUR SYSTEM IN TERMS OF THE NUMBER OF NEWTON ITERATIONS, FUNCTION EVALUATIONS, AND RUNTIME

Method	SRM	ATM	SSM	IBM-A
N. Newton iterations	9633	1255	1160	1241
N. function evaluations	1782105	292415	214600	229585
Average runtime (s)	33.01	6.21	3.88	4.45

the other hand, IBM has the best accuracy compared to ATM and SSM, even in the most simplified version of its Jacobian matrix, while its performance is similar to SSM.

The same simulation is repeated for the IBM variation methods. The accuracy and performance results are listed in Table IV and Table V, respectively.

As the sub-matrixes are added, the number of Newton iterations decreases due to the more accurate Jacobian. However, the run time increases since they require many more function evaluations to be calculated. It can be noticed that specifically, the sub-matrix C calculation leads to a significant increase in function evaluations and consequently performance drop. Among all the variations investigated, IBM-AB seems a reliable option both in terms of accuracy and performance.

TABLE IV
ACCURACY COMPARISON BETWEEN IBM-A, IBM-AB, IBM-AC, AND IBM-ABC WITH RESPECT TO SRM FOR KUNDUR SYSTEM USING EUCLIDEAN DISTANCE

Method	IBM-A	IBM-AB	IBM-AC	IBM-ABC
$\epsilon(V_1)$	0.113806	0.113755	0.113805	0.113754

TABLE V
PERFORMANCE COMPARISON BETWEEN IBM VARIATIONS FOR KUNDUR SYSTEM IN TERMS OF THE NUMBER OF NEWTON ITERATIONS, FUNCTION EVALUATIONS, AND RUNTIME

Method	IBM-A	IBM-AB	IBM-AC	IBM-ABC
N. Newton iterations	1241	1209	1233	1206
N. function evaluations	229585	238121	393337	392652
Average runtime (s)	4.45	5.41	27.47	24.89

V. CONCLUSION

In this paper, four methods to incorporate digital controllers into the dynamic simulation of power systems are presented and compared (both qualitatively and experimentally). Two case studies were utilized to compare the methods in terms of accuracy and performance.

It was shown that for systems with many digital controllers, the analog treatment method, which is widely used today, is not always the fastest method and it lacks accuracy. Also, it was shown that the IBM has the highest accuracy relative to our reference method SRM while it has performance similar to the SSM, which is the fastest method. Furthermore, several IBM variations based on simplified Jacobian matrix formations were introduced and their accuracy and performance were compared, showing that perfecting the Jacobian may lead to a significant performance drop. However, it is shown that adding the sub-matrix B that takes into consideration the impact of the controller's output on the system's equations can be the best choice since it reduces the required number of Newton iterations.

REFERENCES

[1] D. Ellison, "Efficient automatic integration of ordinary differential equations with discontinuities," *Mathematics and Computers in Simulation*, vol. 23, no. 1, pp. 12–20, 1981.

[2] I. A. Hiskens, "Power system modeling for inverse problems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 3, pp. 539–551, 2004.

[3] M. B. Carver, "Efficient integration over discontinuities in ordinary differential equation simulations," *Mathematics and Computers in Simulation*, vol. 20, no. 3, pp. 190–196, 1978.

[4] M. Jafari, G. Bureau, M. Chiaramello, A. Guironnet, P. Panciatici, and P. Aristidou, "Modeling of Digital Controllers in Electric Power System Dynamic Simulations," in *2023 IEEE Belgrade PowerTech*, 2023.

[5] I. A. Hiskens, "Trajectory deadlock in power system models," in *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*. IEEE, 2011, pp. 2721–2724.

[6] D. Fabozzi and T. Van Cutsem, "Simplified time-domain simulation of detailed long-term dynamic models," in *2009 IEEE Power & Energy Society General Meeting*. IEEE, 2009, pp. 1–8.
[7] D. Fabozzi, A. S. Chieh, P. Panciatici, and T. Van Cutsem, "On simplified handling of state events in time-domain simulation," *Proceedings of the 17th PSCC*, 2011.
[8] M. Jafari, G. Bureau, M. Chiaramello, A. Guironnet, P. Panciatici, and P. Aristidou, "An Interpolation-based Method for Numerical Simulation of Digital Controllers in Power System Dynamic Studies," *TechRxiv. Preprint*, 2023. [Online]. Available: 10.36227/techrxiv.23579574
[9] P. Aristidou, S. Lebeau, and T. V. Cutsem, "Power system dynamic simulations using a parallel two-level schur-complement decomposition," *IEEE Transactions on Power Systems*, vol. 31, no. 5, pp. 3984–3995, Sept 2016. [Online]. Available: http://orbi.ulg.ac.be/handle/2268/189192
[10] J. S. Chai, N. Zhu, A. Bose, and D. J. Tylavsky, "Parallel Newton type methods for power system stability analysis using local and shared memory multiprocessors," *IEEE Transactions on Power Systems*, vol. 6, no. 4, pp. 1539–1545, 1991.
[11] U. M. Ascher and L. R. Petzold, *Computer methods for ordinary differential equations and differential-algebraic equations*. Siam, 1998, vol. 61.
[12] S. Karagiannopoulos, G. Valverde, P. Aristidou, and G. Hug, "Clustering Data-Driven Local Control Schemes in Active Distribution Grids," *IEEE Systems Journal*, vol. 15, no. 1, pp. 1467–1476, 2020.
[13] T. M. Inc., *MATLAB version: 9.10.0 (R2021a)*. Natick, Massachusetts, United States: The MathWorks Inc., 2021. [Online]. Available: https://www.mathworks.com
[14] P. Kundur, N. J. Balu, and M. G. Lauby, *Power system stability and control*. McGraw-hill New York, 1994, vol. 7.
[15] IEEE, "IEEE recommended practice for excitation system models for power system stability studies," *IEEE Std 421.5–2016.(Revision of IEEE Std 421.5-2005)*, pp. 1–207, 2016.
[16] P. Pourbeik, "Dynamic models for turbine-governors in power system studies," *IEEE Task Force on Turbine-Governor Modeling*, vol. 1, 2013.