Handling of Computationally Demanding Digital Controllers in Power System Dynamic Simulations

Mehran Jafari*, Gautier Bureau†, Marco Chiaramello†, Adrien Guironnet†, Patrick Panciatici†, and Petros Aristidou*
*Dept. of Electrical Eng., Computer Eng., & Informatics, Cyprus University of Technology, Limassol, Cyprus
†Réseau de Transport d' Électricité (RTE), France
Corresponding email: mm.jafari@edu.cut.ac.cy

***Abstract*—Nowadays Smart grids utilize digital controllers to guarantee the reliable and efficient operation of their components. However, the numerical simulation of these types of controllers is challenging due to the numerous discrete time events introduced. The interpolation-based method (IBM) can effectively simulate smart digital controllers alongside the differential-algebraic equations of the system under control. Compared to the traditional step-reduction method (SRM) that reduces the time step for each time event during the simulation, IBM allows the use of variable time steps and solves the system while ensuring simulation accuracy. However, in the case of computationally demanding controllers, the computational performance of IBM suffers significantly. In this paper, a modified version of IBM is introduced, which can handle computationally demanding controllers in dynamic simulation of power systems. The performance of the proposed method is assessed and compared to SRM and IBM by simulating a test system with different controllers.**

***Index Terms*—Digital controllers, time-domain simulations, interpolation.**

I. Introduction

The safe and reliable operation of smart grids is highly dependent on the smart digital controllers that control them. However, digital controllers are modeled using discrete systems, and their simulation alongside the continuous physical power system model is challenging. The dynamics of such a system are usually modeled by *large-scale, hybrid, stiff, differential-algebraic equations (DAEs)* [1] while the controller is modeled with difference equations [2]. Moreover, modern digital controllers are nonequation-based (e.g. fuzzy logic-based or Machine-Learning-based controllers), which introduces more challenges to their simulation due to their black-box modeling approach.

The first challenge of simulating digital controllers is the discontinuities (events) introduced into the simulation by each of their sample times [3]. Fig. 1 shows the discrete events introduced to the simulation of a system by two digital controllers. Each vertical line indicates a sampling action of a digital controller. These events force traditional solvers to reduce the time step taken h_n to $h_{n,1}$, landing at the first event time $t_{n,1}$ to ensure precision [4]. This process must be followed for all the events through the simulation. While integrating the system's equation using a variable-time-step approach, one can see that the size of the time steps is limited to the distance between consecutive controller samplings. In the case of many digital controllers, this leads to very slow simulations.

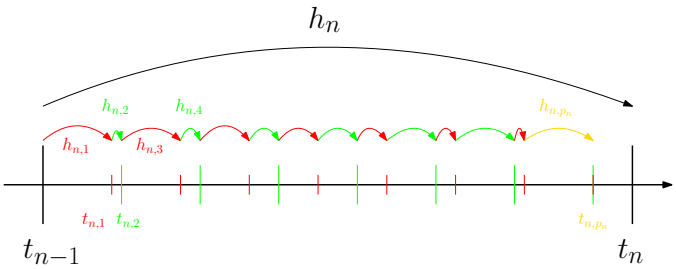

Fig. 1: Discrete events in a simulation with two digital controllers with different sampling rates indicated by red and green ink. The yellow ink indicates their overlapping.

Another challenge is the simulation of smart digital controllers that are not defined by DAEs. Thus, the approach used for the simulation must be able to handle black-box-modeled controllers, such as optimization-based or machine-learning-based controllers. In this case, methods such as the analog treatment method that require the equations of the controller cannot be applied [5].

Moreover, the calculation of a controller output may require a separate simulation or be computationally intensive, which may limit methods such as the interpolation-based method (IBM) that rely on calling the controller many times more compared to other methods [6]. In this paper, this challenge is first demonstrated by analyzing the impact of a computationally demanding controller on IBM performance. Then, two modifications to IBM are proposed for increasing its performance while handling systems with computationally demanding controllers. The modifications aim to reduce the size of the system to be solved and the number of controller calls in each time step.

The remainder of the paper is organized as follows. Section II briefly introduces the SRM and IBM methods for treating discrete events in a system. Then, the light version of IBM is proposed in Section III. A case study in Section IV is used to demonstrate the precision and performance of the proposed method compared to SRM and IBM. Finally, the conclusions are drawn in Section V.

II. System Modeling and Discrete Event Handling

A. System modeling

To introduce the methodologies let's assume a continuous system modeled with a set of DAE with initial values:

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

where $\boldsymbol{y}(t)$ is the state variable vector (both differential and algebraic), and $e(t)$ is the controller output going to the system.

The digital controller is modeled as a black box and can be defined as:

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

where ζ is the function of the controller to be called with inputs e_{k-1} and $\boldsymbol{y}(kT)$ that denotes the controller output at previous sampling action and feedback at k-th sampling coming from the system, respectively.

After discretizing the DAEs with an integration method and considering a time step with size equal to $h_n = t_n - t_{n-1}$ is taken, a Newton method [7] can be used to find the solution of the problem at the end of the time step $\boldsymbol{y}(t_n) = \boldsymbol{y}_n$. The residual function for the Newton method can be defined as:

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

However, the continuous system of (3) is interrupted by the sampling events of the digital controllers that need to be appropriately handled.

B. Step Reduction Method (SRM)

The SRM handles the time events that come from the digital controller sampling by reducing the time step h_n to $h_{n,1}$ to land on the first controller sampling time (event) found in the time step [3]. Then, by calculating the controller action $e(t) = e_{k,1}$, the solution $y_{t,1}$ can be found at time $t_{n,1}$. By repeating this process for other controller sampling actions, the simulation proceeds accurately.

It should be noted that SRM is the most accurate method since it handles each discontinuity separately. However, SRM is the slowest method since it constantly reduces the step size to land on each event (see Fig. 1).

C. Interpolation-Based Method (IBM)

Alternatively, the IBM, proposed in [6], can be used to solve the same problem without reducing the time step. This is possible by using an interpolant polynomial to estimate the feedback of the system at each sampling time $t_{n,g}$ during the time step h_n, without the need to reduce the time step.

For this method, the residual function solved with a Newton method for the time step $h_n = t_n - t_{n-1}$ is defined as:

$$\boldsymbol{g}(\boldsymbol{y}_n, e_{n,p_n}) = 0 \tag{4}$$

where the last controller output e_{n,p_n} within the time-step (see Fig. 1) is required. Based on (2), each controller output $e_{n,g}$ within the time step can be formulated as:

$$e_{n,g} = \zeta(e_{n,g-1}, \boldsymbol{x}_{n,g}, t_{n,g}) \tag{5}$$

where $\boldsymbol{x}_{n,g}$ is an estimation of the feedback states $\boldsymbol{y}_{n,g}$ using an interpolation polynomial $\boldsymbol{w}_n^{(m)}$ for each Newton iteration m as follows:

$$\boldsymbol{x}_{n,g}^{(m)} = \boldsymbol{w}_n^{(m)}(\dot{\boldsymbol{y}}_{\boldsymbol{n}}^{(\boldsymbol{m})}, \boldsymbol{y}_{n-1}, \boldsymbol{y}_n^{(m)}, t_{n,g}), \quad \forall g \in [1, p_n] \tag{6}$$

By combining (1) and (5), a new state variable vector is defined:

$$\boldsymbol{z}_n = \begin{bmatrix} \boldsymbol{z}_{n,1} \\ \boldsymbol{z}_{n,2} \end{bmatrix} \tag{7}$$

with:

$$\boldsymbol{z}_{n,1} = \boldsymbol{y}_n \tag{8}$$

$$\boldsymbol{z}_{n,2} = \begin{bmatrix} e_{n,1} & e_{n,2} & \dots & e_{n,g} & \dots & e_{n,p_n} \end{bmatrix}^T \tag{9}$$

where $\boldsymbol{z}_{n,1}$ is the vector of system state variables and $\boldsymbol{z}_{n,2}$ is the vector of controller outputs. Consequently, the corresponding residual vector is defined as:

$$\tilde{\boldsymbol{g}} = \begin{bmatrix} \tilde{\boldsymbol{g}}_1 \\ \tilde{\boldsymbol{g}}_2 \end{bmatrix} \tag{10}$$

with:

$$\tilde{\boldsymbol{g}}_1(\boldsymbol{z}_n) = \boldsymbol{g}(\boldsymbol{y}_n, e_{n,p_n}) \tag{11}$$

$$\tilde{\boldsymbol{g}}_2(\boldsymbol{z}_n) = \begin{bmatrix} e_{n,1} - \zeta(e_{n,0}, \boldsymbol{x}_{n,1}) \\ \dots \\ e_{n,g} - \zeta(e_{n,g-1}, \boldsymbol{x}_{n,g}) \\ \dots \\ e_{n,p_n} - \zeta(e_{n,p_{n-1}}, \boldsymbol{x}_{n,p_n}) \end{bmatrix} \tag{12}$$

where $\tilde{\boldsymbol{g}}_1$ is the vector of residuals of the continuous system and $\tilde{\boldsymbol{g}}_2$ the vector of residuals formed for the digital controller outputs.

The combined system can be solved with a Newton method using:

$$\boldsymbol{J}_n^{(m)}(\boldsymbol{z}_n^{(m+1)} - \boldsymbol{z}_n^{(m)}) = -\tilde{\boldsymbol{g}}(\boldsymbol{z}_n^{(m)}) \tag{13}$$

To accelerate the simulation process, a simplified (dishonest) Jacobian matrix $\boldsymbol{J}_n^{(m)}$ can be used [5]:

$$\boldsymbol{J}_n^{(m)} = \begin{bmatrix} \frac{\partial \tilde{\boldsymbol{g}}_1}{\partial \boldsymbol{z}_{n,1}} & \frac{\partial \tilde{\boldsymbol{g}}_1}{\partial \boldsymbol{z}_{n,2}} \approx \mathbf{0} \\ \frac{\partial \tilde{\boldsymbol{g}}_2}{\partial \boldsymbol{z}_{n,1}} \approx \mathbf{0} & \frac{\partial \tilde{\boldsymbol{g}}_2}{\partial \boldsymbol{z}_{n,2}} \approx \boldsymbol{I}_{p_n} \end{bmatrix} \tag{14}$$

In this way, IBM can solve the entire system without reducing the time step, leading to an increase in performance while maintaining almost the same level of precision as SRM [8].

Let us assume the controller call cost is a, and solving each Newton iteration cost is b. IBM calls the controller function ζ once per sampling per Newton iteration, therefore, the total controller cost for IBM is equal to $cc_{IBM} = a \times T \times m$, while SRM calls the controller once each sampling time, so $cc_{SRM} = a \times T$. If the controller call cost a is computationally light compared to the cost of solving each Newton iteration ($a \ll b$), the total cost of the simulation is $tc \cong b$. In this case, IBM has better performance than SRM since it takes large time steps and the solver solves the system many

times fewer than SRM. However, if the controller call cost is large and not negligible compared to b, then cc is no longer negligible and $tc = b + cc$. Now, since IBM calls the controller m times more than SRM, then $cc_{SRM} \ll cc_{IBM}$, therefore, $tc_{IBM} > tc_{SRM}$. In other words, simulating a computationally demanding controller with IBM may lead to slower simulation compared to SRM if the added cost of the extra controller calls is significant compared to the computations saved by not reducing the time step.

In addition, the size of the system to be solved increases per sampling found in each time step for IBM. For example, the size of the Jacobian for a system with i equations and j controller equations each having r samples in a time step is equal to $(i + (j \times r)) \times (i + (j \times r))$ while it remains $i \times i$ for SRM.

III. Light interpolation-based method

In this section, the light interpolation-based method (LIBM) is proposed, suggesting two modifications to the original IBM in order to reduce the computational cost arising from many calls to the controllers.

A. Controller calls

The first modification restricts calling the controller evaluation functions only in the first Newton iteration m. Therefore, the controller call cost in IBM becomes $cc_{IBM} = a \times T$, similar to SRM. In other words, the second term of (12) which is the controller function ζ remains constant for the rest of the iterations in each time step. This reduces the number of controller calls and the use of an interpolation polynomial by $m - 1$ times per controller per sampling in a time step.

The drawback of this modification is that the Jacobian and the controller mismatches are accurate only for the first Newton iteration. Therefore, minor inaccuracies are expected to occur.

B. Size of the Jacobian

The second modification aims to reduce the size of the system to be solved by including only the last controller output in each time step to form the Jacobian. Therefore, the Jacobian size remains constant and equal to $(a + b) \times (a + b)$ for the time step (assuming that the controller has at least 1 sample in the time step taken). Therefore, less computation is needed to solve the system in every Newton iteration.

Using the modifications suggested, the number of calls to the controllers and the size of the system to be solved decreases, leading to an increase in performance.

In the next section, the performance increase and the impact of the modifications on the accuracy are demonstrated.

IV. Simulation Results

To compare the methods introduced in the previous section, a 3-bus network illustrated in Fig. 2 is considered. The synchronous generator is assumed to be under the control of a digital governor [9] and a digital exciter [10]. For the governor, an equation-based controller with its continuous block diagram

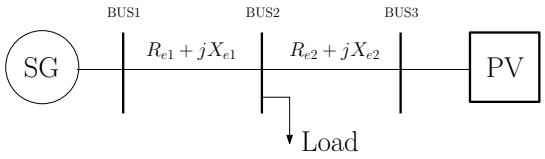

Fig. 2: The schematic of the 3-bus network

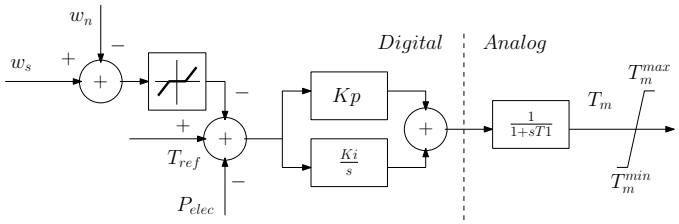

Fig. 3: The schematic of the governor used to control the synchronous generator

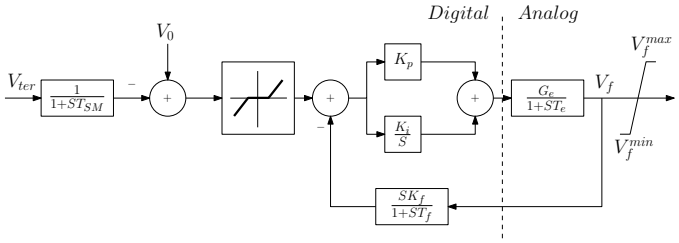

Fig. 4: The schematic of the EQAVR used to control the synchronous generator

TABLE I: Average controller calling time for EQAVR, FAVR, and MLAVR

Controller	Average call time (s)
EQAVR	2.78e-06
FAVR	1.72e-04
MLAVR	3.7e-03

shown in Fig. 3 is considered. For the exciter, however, three different controllers are simulated as follows:

1) An equation-based AVR (EQAVR) with its continuous block diagram presented in Fig. 4.
2) An AVR based on fuzzy logic (FAVR).
3) A machine learning-based AVR (MLAVR) trained on the basis of the results obtained from the simulation of EQAVR.

The average call time of the controller for each of these controllers is listed in Table I. Each controller is used for simulation with all three methods and the results in terms of accuracy and performance are compared.

For all simulations, the equations are integrated using a variable time-step predictor-corrector method consisting of a pair of second-order Adams-Bashforth and Adams-Moulton methods [11]. 18 bits are used for the quantization of the digital controller inputs and outputs. The minimum and maximum time step sizes allowed are equal to 1ms and 1s, respectively.

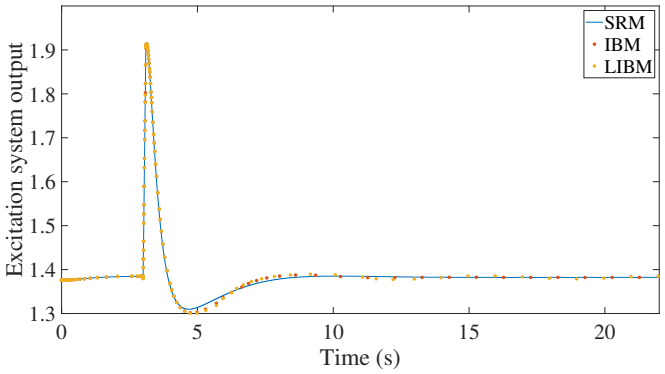

Fig. 5: Output of the EQAVR using all three methods

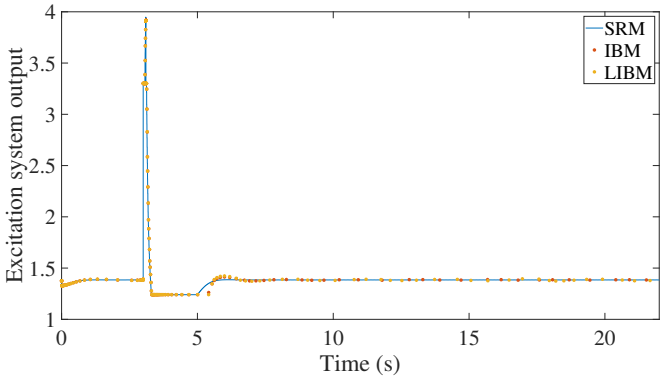

Fig. 6: Output of the FAVR using all three methods

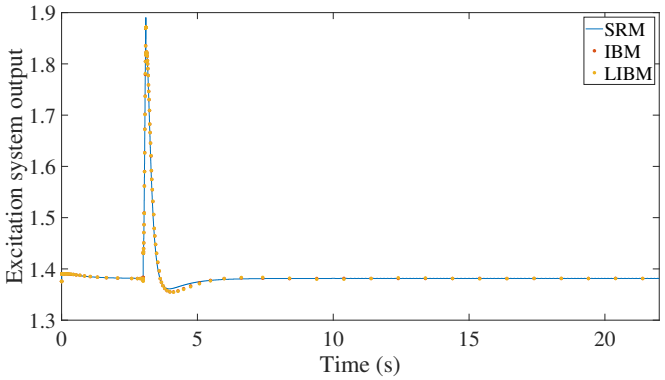

Fig. 7: Output of the MLAVR using all three methods

In addition, the increasing and decreasing rates for altering the step size are equal to 1.25 and 0.5, respectively. Finally, 20 ms and 4 ms are considered for the sampling time of the governor and all AVRs, respectively.

A short circuit, applied to bus 2 for 100 ms, is simulated for all three methods and all three variations of the AVR controller. The controller output for EQAVR, FAVR and MLAVR, simulated with all three methods, is illustrated in Figs. 5, 6, and 7, respectively.

The performance results in terms of the average of 10 runs for each simulation are listed in Table II. The first row of

TABLE II: Average runtime of the simulation of each controller and method in seconds.

	EQAVR	FAVR	MLAVR
SRM	23.90	24.80	42.91
IBM	0.48	4.00	56.93
LIBM	0.36	2.34	34.46

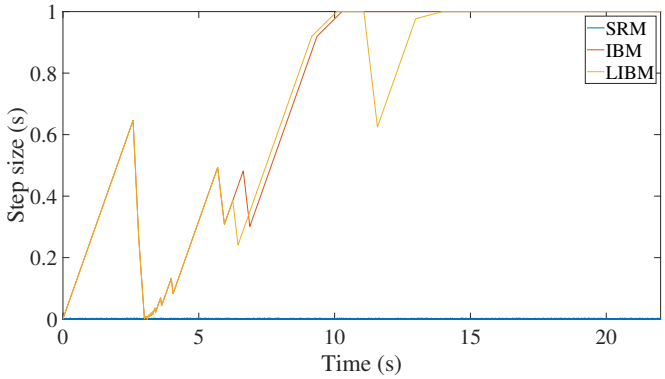

Fig. 8: Step size results for the simulation of EQAVR using all three methods

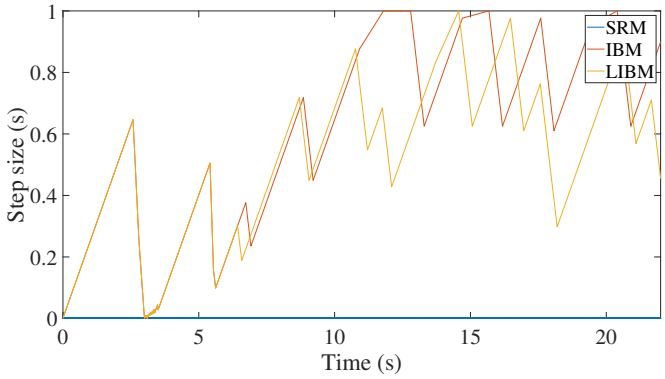

Fig. 9: Step size results for the simulation of FAVR using all three methods

the table shows that the runtime for MLAVR simulation using SRM is almost twice as that of the simulation of EQAVR and FAVR using the same method. However, this performance drop is more significant for IBM and LIBM, since controller calling is the heaviest computation for these methods. This can be deduced from Figs. 8, 9, and 10 that show the step size results for the simulation of EQAVR, FAVR, and MLAVR, respectively, using all three methods. In other words, although IBM still manages to take the maximum step size allowed for most of the simulation time of MLAVR, the run time decreases significantly due to the long time and numerous controller calls.

The main challenge is that IBM, which is many times faster than SRM for the simulation of EQAVR and FAVR, becomes 14 s slower than SRM for the simulation of MLAVR. However, LIBM is still 8 s faster than SRM because of fewer controller

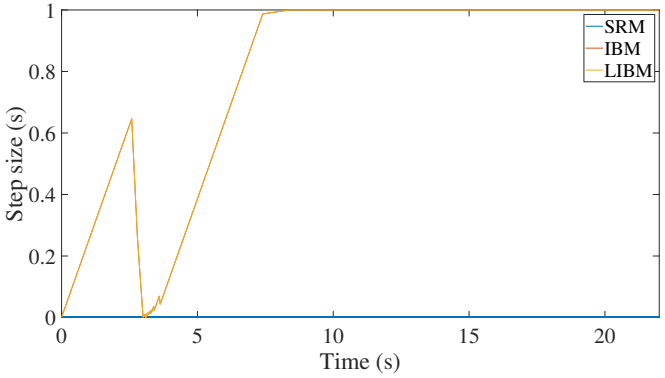

Fig. 10: Step size results for the simulation of MLAVR using all three methods

TABLE III: Number of Newton iterations for the simulation of each controller and method.

	EQAVR	FAVR	MLAVR
SRM	54604	55136	54504
IBM	551	585	520
LIBM	554	598	522

calls and solving smaller systems. It should be noted that this increase in performance is not significant for other controllers, so, in the case of normal controllers, IBM would still be the better choice ensuring better accuracy.

The number of Newton iterations required for the simulation of each controller using each method is shown in Table III. As can be seen, fewer Newton iterations are needed to solve the MLAVR system, which supports the fact that the number of controller calls leads to a performance drop. Moreover, it can be seen that using LIBM leads to an increase in the number of Newton iterations since the Jacobian updates were done once per time step.

V. Conclusion

In this paper, a modified version of IBM was proposed for the simulation of computationally heavy controllers. This method brings two modifications to the original IBM that lead to fewer calls to the controller and a solving smaller system for each time step.

Three controllers with different models and different calling times were simulated, and an increase in the performance of the simulations was shown for LIBM compared to IBM. It was also shown that the accuracy lost due to these changes does not have a profound impact on the controller output.

References

[1] F. Milano, *Power System Modelling and Scripting*, ser. Power Systems. Springer Berlin Heidelberg, 2010.

[2] 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.

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

[4] F. Zhang, M. Yeddanapudi, and P. J. Mosterman, "Zero-crossing location and detection algorithms for hybrid system simulation," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 7967–7972, 2008.

[5] M. Jafari, G. Bureau, M. Chiaramello, A. Guironnet, P. Panciatici, and P. Aristidou, "Methods for incorporating digital controllers in power system dynamic simulations," *Electric Power Systems Research*, vol. 235, p. 110827, 2024.

[6] ——, "An Interpolation-based Method for Numerical Simulation of Digital Controllers in Power System Dynamic Studies," *TechRxiv. Preprint*, 2023. [Online]. Available: 10.36227/techrxiv.23579574

[7] 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

[8] M. Jafari, G. Bureau, M. Chiaramello, A. Guironnet, P. Panciatici, and P. Aristidou, "Decoupled interpolation-based method for numerical simulation of digital controllers," in *2024 IEEE International Systems Conference (SysCon)*. IEEE, 2024, pp. 1–6.

[9] A. G. NEPLAN, "TURBINE-GOVERNOR MODELS: Standard Dynamic Turbine-Governor Systems in NEPLAN Power System Analysis Tool," Tech. Rep., 2015.

[10] ——, "EXCITER MODELS: Standard Dynamic Excitation Systems in NEPLAN Power System Analysis Tool," Tech. Rep., 2013.

[11] U. M. Ascher and L. R. Petzold, *Computer methods for ordinary differential equations and differential-algebraic equations*. Siam, 1998, vol. 61.