Towards an Open-Source Real-Time Operator-Training Platform: Analysis of Computational Efficiency

Savvas Panagi*, Thibaut Vermeulen†, Geethu Joseph‡, and Petros Aristidou*
* Dept. of Electrical & Computer Engineering, & Informatics, Cyprus University of Technology, Limassol, Cyprus
† Réseau de Transport d'Électricité (RTE), Paris, France
‡ Collaborative Research for Energy System Modelling (CRESYM), Brussels, Belgium
Emails: {savvas.panagi, petros.aristidou}@cut.ac.cy, geethu.joseph@cresym.eu, thibaut.vermeulen@rte-france.com

***Abstract*—Real-time operator-training platforms require dynamic power-system simulation that executes within strict SCADA refresh-rate constraints while preserving high modeling fidelity. This challenge is particularly critical for Transmission System Operators, who must train personnel to manage large-scale transmission networks under realistic contingency scenarios and topology changes. This paper evaluates the real-time performance of the time-domain power system simulator underpinning an operator-training environment. Using Dynaωo as the dynamic simulation engine, we quantify solver efficiency, stability, and scalability on commodity hardware under event-driven scenarios (loss of busbar section, line, and generator). A benchmarking protocol stabilizes the environment and profiles the time-domain integration window. Performance is assessed via step solving times and headroom relative to SCADA-rate synchronization. Results show faster-than-real-time execution in normal operation and for simplified substation-detail configurations, with brief spikes at disturbance instants linked to Jacobian restructuring. Increasing topology retention preserves operational fidelity but expands the number of discrete and continuous states, causing occasional real-time violations during events. We analyze hotspots and outline optimization levers, including solver tolerances, time-step control, sparse linear algebra, and event handling. The study demonstrates the feasibility of open-source dynamic simulation for operator training while identifying operational boundaries and practical paths to robust sustained real-time performance.**

***Index Terms*—Dynamic simulation, Operator training, Performance profiling, Real-time simulation**

I. Introduction

Modern power systems have evolved into complex cyber–physical infrastructures that integrate renewable generation, distributed resources, advanced control, and sophisticated protection schemes [1–3]. This transformation fundamentally alters the operational landscape for transmission system operators (TSOs), who must make real-time decisions amid intricate model interactions, cyber–physical security considerations, and emerging AI-support tools. Consequently, comprehensive and realistic training is required to prepare operators for emergency response, system restoration, and the coordination of multi-layered control strategies [4].

Real-time operator-training platforms are indispensable for developing these skills [1, 5, 6]. They must act as digital twins of the physical grid, reproducing system dynamics while operating at or faster than real time [7]. The core simulation must span steady-state operation through transient stability yet remain computationally efficient for seamless integration with Human–Machine Interface (HMI) systems and control-center workflows [8].

Integrating such simulators with Supervisory Control and Data Acquisition (SCADA) and Energy Management Systems (EMS), while maintaining synchronization with acquisition rates, presents significant challenges [5, 9]. The central trade-off is accuracy versus speed: temporal resolution should match typical 2–4 s scan rates [10] while preserving numerical precision to capture phenomena such as voltage-stability margins and frequency excursions [11].

A review of existing platforms reveals notable limitations. Many rely on static or simplified simulation modules [12, 13] that fail to capture the dynamic complexity of modern power systems [14]. Traditional simulators commonly employ reduced-order models [15], linearized approximations [16], or pre-computed scenario databases [17] that cannot represent nonlinear, time-varying behavior or support interactive HMI.

Moreover, most solutions are closed-source and tightly embedded within vendor-specific SCADA and control-center stacks [18]. These constraints limit customization and transparency and impose licensing costs that exclude small TSOs and academic institutions.

To address these limitations, this paper develops a methodology for testing and benchmarking Dynaωo, a dynamic simulator supported by the Linux Foundation Energy initiative [19, 20] under realistic training scenarios. While previous studies mainly focus on the analysis of total computational time [21, 22] and relatively small testing network sizes [23, 24], this work instead investigates computational effort at the time-step level to better align with the requirements of the targeted application. Dynaωo implements full Differential–Algebraic Equation (DAE) solvers, capturing transient

Submitted to the 24th Power Systems Computation Conference (PSCC 2026).

phenomena from electromechanical oscillations to voltage-stability margins. The platform targets standard commodity hardware and Linux-based operating systems, with optional Docker deployment.

The open-source foundation provides algorithmic transparency, unrestricted customization, and elimination of recurring license costs, while enabling collaboration between utilities and academia. Executing on multi-core x86 machines with standard memory, the platform is deployable in existing control centers. Full dynamic modeling ensures accurate representation of cyber–physical interactions, renewable integration, and protection coordination.

The primary contributions of this work are:

1) Systematic quantification of solving speed in real-time dynamic simulation, including time-step optimization, and convergence behavior under diverse scenarios and contingencies.
2) Development of methodology and benchmarks to evaluate deployment feasibility, including scalability analysis and computational-complexity assessment tools.
3) Identification and validation of algorithmic enhancements, solver-tuning methodologies, and system-level optimizations that improve computational efficiency on commodity platforms.

The remainder of this paper is organized as follows. Section II presents the operator-training platform, detailing its architecture, core components, and real-time refresh-rate constraints. Section III details the evaluation methodology, including computational-environment stabilization and the performance-evaluation framework. Section IV describes the Dynaωo solver and the French transmission test case. Section V reports experimental results, examines topology-retention impacts, and outlines profiling insights and improvement strategies. Finally, Section VI concludes and discusses directions for future work.

II. Modern Operator Training Platform

Modern power systems exhibit increasing complexity and operational interdependence. Consequently, effective operator preparation demands training environments that are both sophisticated and realistic [25]. This section details the development of a comprehensive operator-training platform with enhanced grid-simulation capabilities, building upon the foundational concepts discussed in Section I. The architecture is designed to balance computational efficiency and simulation fidelity, with particular attention to real-time constraints that mirror SCADA-system behavior.

A. Architecture and Core Components

The platform employs a multi-component architecture comprising six interconnected subsystems, Orchestrator, Game Master, SCADA Human-Machine Interface, Information Flow Model, Physical Simulator, and Automata, as shown in Fig. 1 [8, 9, 11]. Each subsystem fulfills a specialized role while sustaining seamless integration with the overall training environment. The operator-training platform is developed within

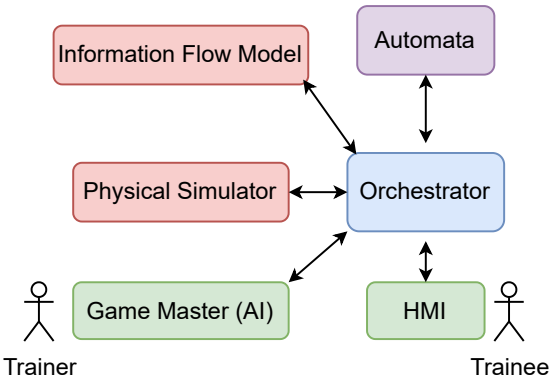

Fig. 1. Overview of modern operator training platform [26]

the ongoing TwinEU project [26], the primary objective of this paper is not the formal architectural modeling of the platform itself, but rather the systematic analysis of the dynamic simulator solving-time feasibility for real-time operation.

The **Orchestrator** serves as the central coordination hub. It manages temporal synchronization across all components and maintains consistency with real-time operational requirements [7, 27]. In practice, it enforces strict adherence to refresh-rate constraints, typically 2–4-second intervals, to match standard SCADA scan cycles [9, 10, 28]. Its primary responsibilities include orchestrating inter-subsystem data exchange and preventing temporal inconsistencies that could compromise training effectiveness.

The **Game Master (GM)** functions as the intelligent scenario controller, implementing predefined training scenarios and dynamically adapting system conditions based on trainee actions [1, 4]. It introduces realistic disturbances, equipment failures, and operational challenges at controlled intervals, thereby ensuring authentic operational complexity. Its scenario library spans routine operational tasks through complex emergency situations.

The **Human-Machine Interface** provides the primary interaction mechanism, replicating the control-center environment with high fidelity [6, 7]. It presents system status, alarm notifications, and control interfaces that mirror real-world SCADA tools, cultivating operational fluency with the interfaces encountered in practice.

The **Information Flow Model (IFM)** is an emulation of the information system. It models information flows over the communication networks, introducing stochastic delays, data-quality variations, and communication failures that operators must accommodate.

The **Physical Simulator (PS)** is the computational core, implementing a high-fidelity physical simulator of the power system. It solves hybrid, nonlinear DAE systems that capture the dynamics of generators, transmission lines, transformers, and loads [16].

The **Automata** subsystem encodes discrete-event logic for protection, control, and operations via finite-state/timed automata. Each automaton evaluates measurements, alarms, and

breaker states and issues atomic actions (trip/close, setpoint updates, topology edits, GM callbacks) with deterministic, time-stamped ordering.

B. Computational Efficiency and Refresh Rate Requirements

The platform operates under stringent real-time constraints dictated by SCADA refresh rates, typically requiring updates every 2–4 seconds to maintain operational realism [10]. As the most computationally intensive component, the physical simulator must complete its computations within this budget while accommodating the overhead introduced by other subsystems.

Higher physical-model fidelity enables richer and more realistic scenarios; however, it increases the computational burden and may compromise real-time performance. Conversely, simplified models reduce computational demands but may lack the fidelity required for comprehensive training. Therefore, model complexity must be optimized to balance system-level details against computational efficiency.

The Automata component adds further computational load via discrete-event processing, protection-system modeling, and control-logic execution. These event-driven computations must fit within the global timing budget while maintaining deterministic behavior, which is essential for reproducible training scenarios.

C. Physical Simulator

Power systems are typically modeled as index-1 DAEs, and dynamic simulation reduces to solving a DAE Initial Value Problem (IVP) over a prescribed time horizon [29]. The system equations are:

$$\mathbf{0} = \boldsymbol{F}(\dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), \boldsymbol{z}(t), \boldsymbol{p}) \tag{1}$$

$$\boldsymbol{x}(0) = \boldsymbol{x}_0, \ \ \boldsymbol{z}(0) = \boldsymbol{z}_0, \tag{2}$$

$$t \in [0, T_{horizon}] \tag{3}$$

where $\boldsymbol{x}(t)$ denotes the vector of differential–algebraic states, $\boldsymbol{z}(t)$ the vector of discrete states, and $\boldsymbol{p}$ the system parameters.

The simulator must accommodate discrete events, breaker operations, tap-changer movements, and protection actions, while maintaining numerical stability and solution continuity. Time-domain integration methods, particularly implicit schemes such as the trapezoidal rule, provide the stability required to handle the stiffness typical of power-system dynamics.

Dynamic parameter-modification capabilities enable real-time scenario adaptation, allowing the Game Master to introduce system changes without interrupting the simulation. This functionality is essential for progressive scenarios that respond to trainee actions and decisions.

The "faster-than-real-time" requirement necessitates optimized solvers that achieve solution speeds significantly exceeding the refresh rate while maintaining acceptable accuracy. This constraint drives the selection of efficient numerical algorithms, sparse matrix techniques, and optimized computational architectures capable of meeting demanding performance targets.

Together, these components constitute a comprehensive training environment that provides operators with realistic experience in power-system operation while preserving the safety and repeatability essential for effective programs. Ultimately, platform success depends on achieving an optimal balance between computational efficiency and simulation fidelity, thereby ensuring high-quality preparation for real-world operational challenges.

III. Evaluation of Physical Simulator Performance for Operator Training

This section presents a simulator- and system-agnostic procedure to determine whether the physical power-system simulator can sustain real-time operation within the platform described in Section II. The focus is exclusively on the PS component; initialization, model compilation, and warm-up occur prior to training and are outside the scope of this assessment.

A. Headroom Metric and Real-Time Criterion

The fundamental performance indicator is the *headroom per time step*, defined as

$$dt = T_{\text{sync}} - T_{\text{sol}},$$

where T_{sync} is the real-time interval allotted to each simulation step (e.g., SCADA/HMI refresh period) and T_{sol} is the actual computational time consumed by the simulator. Positive dt indicates available margin (faster-than-real-time); negative dt signals a real-time violation. Beyond non-negative headroom, sustained real-time execution typically requires a minimum margin; let $dt_{\text{min}} > 0$ denote the buffer that covers orchestrator/HMI overhead and OS jitter. The operating condition is $dt \geq dt_{\text{min}}$.

Accurate evaluation requires per-step timing instrumentation within the simulator runtime to record T_{sol} and to profile variability around discrete events. Unless explicitly required by the training logic, the simulator should avoid retaining full historical trajectories in memory; only the data streams published to the Orchestrator need to be persisted, reducing memory footprint and garbage-collection overheads. Default numerical parameters in off-the-shelf simulators are typically tuned for maximum accuracy; for operator-training purposes, parameters should be systematically re-tuned to prioritize speed and robustness while remaining within the accuracy implied by T_{sync} ("T_{sync}-level accuracy").

B. Computational Environment Settings

Ensuring consistency in simulation results requires a stable computational environment in which variations in execution time and solver performance are minimized. Because the PS executes on a dedicated host machine, all stabilization actions are applied on that specific computer. Modern CPUs dynamically adjust their frequency based on system load, power-saving mechanisms, and temperature, which can introduce inconsistencies in performance measurements across multiple

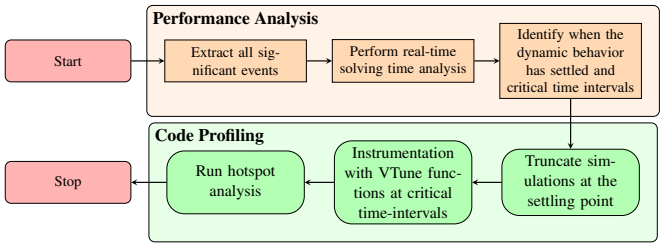

Fig. 2. Analysis Methodology

simulation runs. Therefore, a two-pronged stabilization approach is implemented.

First, CPU frequency scaling is disabled and the processor is pinned to a fixed performance state targeting approximately 85% of its maximum capacity, balancing determinism with thermal and power headroom. Second, process-priority configuration is applied to the simulation processes to ensure preferential CPU dispatch and prevent interference from background tasks. In Linux environments, this entails launching the simulator with the highest user-space priority using a negative nice value (e.g., `nice -20`/`renice -20`) and, where permitted, applying a real-time scheduling policy (`SCHED_FIFO`) via `chrt -f` with elevated priority, optionally combined with CPU affinity to reduce scheduler contention.

C. Performance Evaluation Framework

A systematic multi-step methodology, illustrated in Fig. 2, evaluates computational performance under realistic training scenarios. The procedure ensures reproducible and reliable assessment of solver behavior across diverse operating conditions by moving from raw event definition to targeted hotspot diagnosis as described below:

1) Define the training scenario set in collaboration with system-operator engineers (events, operator maneuvers, operating conditions). Where available, prepare multiple model-detail levels to trade fidelity for speed. In practice, all significant contingency events are compiled into an event catalog including, inter alia, line disconnections, generator trips, and transformer outages representative of realistic operator training scenarios. For each event, metadata are recorded to enable consistent replay and cross-run comparability.
2) Instrument and execute scenarios while recording performance statistics for steady operation and event intervals. Key indicators include dt, solver-iteration counts, residual and Jacobian evaluations, and Jacobian refactorization frequencies. Real-time constraint violations and solver slowdowns during discrete events are detected and logged, yielding time-aligned traces that expose bottlenecks and transient performance excursions.
3) Systematically tune the solver parameters to improve computational speed while keeping the solving accuracy. This process unfolds in several steps: first, perform parameter sweeps across key solver settings, including numerical tolerances, iteration limits, Jacobian update and reuse strategies, time-step adaptation controls, and event-handling heuristics. Next, for each configuration, use the recorded dt values to evaluate whether the resulting performance meets the real-time constraint $dt \geq dt_{\text{min}}$ across all scenarios. Finally, identify and select either scenario-specific optimal parameter sets or, where feasible, a robust global configuration that consistently maximizes speed without affecting the accuracy.

If the tuned parameter set cannot achieve sustained non-negative headroom across all scenarios, the following remediation levers should be considered:

1) Change T_{sync} (adjust the refresh period), if this is allowed by the overall system design.
2) Reduce accuracy (looser tolerances and simplified models).
3) Reduce system size (lower-detail models, decompose and introduce equivalents, prune non-essential dynamics).
4) Improve simulator performance (more efficient numerical methods, smarter update policies, parallel processing, optimized linear algebra).

D. Simulator Improvement Identification

To identify the modifications needed to improve simulator performance, a focused, profiler-driven diagnostic is applied:

1) For each event with $dt < dt_{\text{min}}$, estimate the stabilization time (t_{stab}) as the earliest time the system attains a new equilibrium (e.g., state and output derivatives below small thresholds over a sliding window). This yields an event-specific stabilization marker that delimits the analysis window $[t_{\text{event}}, t_{\text{stab}}]$.
2) Profile exclusively within this window to expose internal performance characteristics. Instrument the simulator with start/stop hooks around the window and enable call-graph sampling. Collect call stacks, inclusive/exclusive CPU and wall-clock times, allocation counts, isolating the dynamic segment of interest.
3) Aggregate and rank functions/modules by exclusive time and cycle consumption, with emphasis on event handling, model evaluation, linearization, sparse factorization, and I/O. The resulting hotspot list prioritizes code paths for optimization and quantifies expected impact on real-time feasibility and headroom.

IV. Case Study Description

A. Dynaωo Description

As detailed in Section I, Dynaωo [20] was selected as the PS for this work. Dynaωo is an open-source dynamic simulation environment, jointly developed by Réseau de Transport d'Électricité (RTE) and academic–industrial partners. Architected as a modular and extensible framework for time-domain studies of large-scale grids, it delivers numerical stability and high computational performance for the hybrid, nonlinear dynamics typical of modern networks. Its openness enables rigorous scrutiny of models and algorithms, reproducibility of results, and collaborative advancement of methods [20].

Within the platform of Section II, Dynaωo provides comprehensive dynamic and static component models, supports fault and contingency analysis, and exposes sufficient interfaces for instrumentation and profiling. These characteristics make it a suitable vehicle to instantiate the headroom-based performance-evaluation protocol above, while preserving the scientific transparency emphasized in Section I.

Simulation in Dynaωo relies on a combination of a set of models of network components and a solver, the choice of which depends on the type of phenomena and the time span of the study. For this work, DynaWaltz, a configuration suitable for long-term stability study, was selected.

B. RTE Power System Dynamic Model

The power-system model encompasses more than 6000 buses, 7075 branches, and 335 three-winding or four-winding synchronous generators with detailed dynamic representations, including excitation systems, automatic voltage regulators, power system stabilizers, turbines, and speed governors. The network incorporates 3039 dynamically modeled loads featuring various types of voltage-control automata (secondary voltage regulation, shunt automaton models), voltage-sensitive loads (Alpha–Beta), and other specialized load representations to capture the system's dynamic behavior accurately.

The model represents the full French network with voltage levels from 63 kV to 400 kV (excluding the eastern 63 kV network), for a total of 4009 substations. In addition, a sample of substations in neighboring countries is included to ensure consistent results near borders. Long-term dynamics are governed by 839 load tap changers (LTCs) distributed across 2138 two-winding transformers, along with current-limiting automatons and tap-changer blocking devices. Due to the size of the French transmission network, which is considered one of the largest single-design power systems, this case represents a highly demanding dynamic simulation scenario.

In this work, three levels of model detail are considered, each significantly affecting the internal topological representation. This allows investigation of how computational time varies under different configurations, which is crucial for real-time operation. In particular, increasing the substation-level detail is essential to preserve the capability of both the trainer and the trainee to perform operational actions while maintaining control over real network components. The description of each model is provided below:

- **Model 1**: Designed for fast event-based simulations with simplified topology. This configuration includes limited substation-level detail and neglects nonessential switches.
- **Model 2**: Represents a mid-fidelity model that tracks all breakers, but no disconnectors, corresponding to a minimal realistic scenario by enabling actions on powered feeders and busbar couplers.
- **Model 3**: Provides a maximum-detail configuration where all switches are tracked (breakers and disconnectors), resulting in a larger number of model parameters and potentially higher computational requirements. This setup is used for switching scenarios including all possible combinations of routing between feeders and busbar sections.

TABLE I
MODEL COMPLEXITY METRICS FOR THE THREE RTE POWER SYSTEM CONFIGURATIONS

Variable/Function	Model 1	Model 2	Model 3
Continuous	80289	210150 (+162%)	319880 (+299%)
Discrete	104650	225518 (+116%)	334848 (+220%)
Root Functions	165537	221645 (+34%)	276151 (+67%)

It should be noted that all three models have the same topological representation and dynamical models (e.g., generators, dynamic loads, etc.). The only difference is the level of detail in the substation-level representation. Table I summarizes the number of continuous and discrete variables, as well as root functions, for each configuration.

C. Scenario Description

Overall, 14 contingency scenarios were provided by the control center, including Loss of Busbar Section (LBS), Loss of Line (LL), and Loss of Generator (LG) events. In this paper, three representative scenarios are presented for each event type, with all events occurring at $t = 10\,\text{s}$.

- **Loss of Busbar Section (LBS)**: This scenario involves the outage of a busbar section, followed by the disconnection of two transformers connected to it. The pre-event loading of the transformers was 382 MW and 242 MW, respectively.
- **Loss of Line (LL)**: This scenario represents the disconnection of a transmission line carrying 75 MW under pre-event conditions.
- **Loss of Generator (LG)**: This scenario corresponds to the sudden disconnection of a generator at $t = 10\,\text{s}$, which was operating at a pre-event active power of 1546 MW.

V. RESULTS AND DISCUSSION

This section presents comprehensive experimental results demonstrating the performance characteristics of the Dynaωo simulator under various operating conditions, as well as the impact of different modeling approaches on computational efficiency for achieving real-time compatibility. All simulations were executed on a laptop equipped with an AMD Ryzen 5 4600H processor (3.0 GHz nominal frequency) and 8 GB of RAM, running Ubuntu 22.04 LTS.

To ensure reproducible performance assessment, CPU-stabilization techniques were implemented to establish a controlled computational environment. Continuous monitoring of CPU frequency confirmed stable operation at 2.8 GHz throughout all experimental runs, effectively eliminating variations caused by dynamic frequency scaling. Further details on the computational environment are provided in Section III-B.

A. Event-Driven Dynamic Performance Evaluation

This analysis evaluates solver performance under realistic training scenarios to reveal critical insights into computational

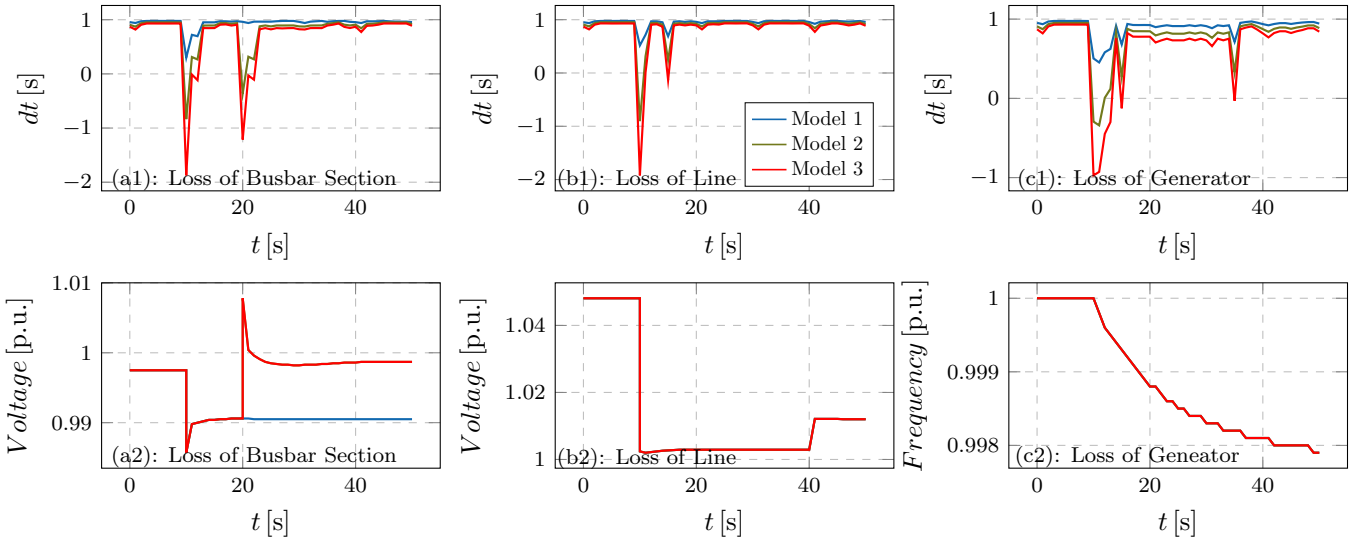

Fig. 3. Solver computational time and system dynamic response for three network model configurations under disturbance events

requirements. In all cases, the event occurs at $t = 10$ s, and the corresponding dt is recorded, as depicted in Fig. 3. Across all step sizes, the solving time remains low both before and after the event, indicating consistent solver performance. However, a distinct computational spike occurs at the moment of the event due to significant grid reconfiguration, as confirmed by log entries indicating algebraic-mode changes with Jacobian updates. Moreover, in some scenarios, smaller subsequent spikes are also observed, attributed to increased system complexity following the event or a second Jacobian restructuring.

The *Model 1* configuration is represented by the solid blue line. During the LBS event, no time-step violations ($dt < 0$) were observed. A single noticeable computational spike appears at the event instant ($t = 10$ s), yet sufficient headroom remains before reaching the synchronization limit.

The subsequent event, LL, is considered less demanding in terms of network disturbance, as the faulted element carries a lower pre-fault power. However, the voltage trajectory at the connected bus exhibits a more pronounced dip, approximately 0.05 p.u. below the previous value, due to the direct line disconnection. From a computational perspective, the solver complexity during the event remains below the synchronization threshold, ensuring that the simulation stays safely within real-time operation margins.

The last event, LG, is the most demanding scenario, corresponding to the loss of a nuclear generator unit operating at high output power. The system frequency transiently drops to 49.9 Hz, remaining within the normal operational band defined by ENTSO-E. Overall, under the simplified *Model 1* configuration, all disturbance events complete without any real-time violations, confirming that the system can operate in real-time simulators without additional performance optimization.

B. Impact of Network Model Fidelity on Computational Performance

This subsection investigates how increased model detail affects simulation performance. The *Model 2* configuration is represented by the green line, while the fully detailed *Model 3* configuration is shown in red in Fig. 3. The comparison of computational costs across all models and scenarios is summarized in Table II.

In the *Model 2* configuration, time-step violations appear (see Fig. 3-a1/b1/c1). All observed real-time violations occur during Jacobian restructuring, except for the LG scenario, where an additional violation appears at the time step immediately after the event, caused by the extended solving time required for convergence.

Among all cases, *Model 3* exhibits the highest computational burden, showing a moderate increase in solving time per step during steady-state periods but a significant rise during topology-change intervals. During the LBS event, both Model 2 and Model 3 exhibit an additional topology modification at $t = 20$ s caused by a subsequent line disconnection, which is not represented in *Model 1* due to the model simplification. This simplification of *Model 1* accelerates the simulation but compromises accuracy (see deviation of blue and red lines in Fig. 3-a2).

The LL scenario presents a similar computational profile to the LBS event, although the secondary spike is less pronounced and remains close to the real-time threshold, even under higher-detail configurations. The LG event is comparatively less demanding in computational terms, with $T_{sol} \leq 2$ s (see Fig. 3-c1); however, subsequent computationally intensive time steps result in additional dt violations.

Overall, the transition from *Model 1* to *Model 2* increases the computational cost by approximately 2.6–3.6×, while *Model 3* amplifies this to nearly 4–6×, depending on the event type. Therefore, although the solving time per step remains

TABLE II
PERFORMANCE ANALYSIS (MODEL 1/2/3) FOR DEFAULT SETTINGS

Metric	LBS	LL	LG
dt Violations	0/2/6	0/1/2	0/2/6
Jacobian Restructurings	1/2/2	1/1/1	1/1/1
$\|\|T_{sol}\|\|_\infty$ [s]	0.7/1.8/2.9	0.5/1.9/2.9	0.5/1.3/2.0
Residual Evals.	220/251/251	218/219/219	317/317/317
Jacobian Evals.	7/12/12	6/7/7	10/10/10
Nonlinear Iters.	119/149/149	117/118/118	216/216/216
Root Func. Evals.	121/127/127	118/120/120	140/140/140
Discrete Var. Evals.	82/89/89	65/67/67	107/107/107
Mode Evals.	62/64/64	48/48/48	68/68/68

TABLE III
PERFORMANCE IMPROVEMENT FROM SOLVER PARAMETER TUNING ON MODELS 2 AND 3

Scenario	**Model**	dt **Violations**		$\|\|T_{sol}\|\|_\infty$ **[s]**		**Speedup**
		Default	Tuned	Default	Tuned	
LBS	Model 2	2	0	1.84	0.89	2.07x
	Model 3	6	6	2.87	1.40	2.05x
LL	Model 2	1	0	1.9	0.95	2.00x
	Model 3	2	2	2.92	1.48	1.97x
LG	Model 2	2	1	1.34	1.32	1.02x
	Model 3	6	6	1.97	1.91	1.03x

close to the synchronization limit for *Model 2*, with occasional violations, the solving time during critical event intervals for *Model 3* makes sustained real-time operation considerably more challenging.

Comparing the solver-level parameters across models reveals distinct patterns. For the LG scenario, all solver-level parameters remain identical among models, as no event-related changes occur under different configurations. In the LBS scenario, a significant deviation is observed between Model 1 and Models 2/3 due to the second event (loss of line) that Model 1 fails to capture. In contrast, the LL scenario shows only minor variations between Model 1 and Models 2/3, which are considered negligible and attributed to the different event modeling approach caused by the absence of certain switches.

C. Solver Parameter Tuning for Real-Time Feasibility

The real-time violations observed in Models 2 and 3, particularly during topology-change events, necessitate systematic solver optimization to improve computational performance while preserving simulation accuracy. Following the methodology outlined in Section III, a targeted parameter-tuning strategy was developed focusing on the algebraic-restoration phase, which dominates the computational burden during discrete events.

Two critical parameters were identified for optimization that significantly impact the performance after a significant event. First, `fnormtolAlgJ` controls the convergence criterion for the nonlinear solver (KINSOL) during algebraic restoration after a "tough event" (discontinuity that requires Jacobian recalculation). Relaxing this tolerance allows convergence with fewer iterations while maintaining the system accuracy. Second, `msbsetAlgJ` determines how frequently the Jacobian matrix is re-evaluated and the linear solver is reconfigured during the nonlinear iteration following a "tough event".

Table III and Fig. 4 quantify the impact of these modifications across all test scenarios. Performance improvement is measured using the *Speedup* metric, defined as the ratio of default to tuned computational times over the worst solving time step; values greater than unity indicate acceleration. The results demonstrate substantial and consistent performance gains for topology-intensive events. In the LBS scenario, Model 2 achieves a 2.07× speedup while completely eliminating real-time violations (from 2 to 0), and Model 3 attains a 2.05× speedup with maximum solving time reduced from 2.87 s to 1.40 s. Similarly, the LL event exhibits approximately twofold acceleration for both models, with Model 2 again achieving zero violations.

The LG scenario, however, presents a different characteristic. Here, the speedup is minimal (1.02× for Model 2, 1.03× for Model 3), with the solving-time trajectory remaining essentially unchanged. This behavior is attributed to the fundamentally different nature of the LG event: unlike LBS and LL, which involve significant network reconfiguration and multiple topology changes, the generator loss primarily affects the electromechanical dynamics rather than the algebraic network structure. Consequently, the algebraic-restoration phase—the target of the tuning strategy—plays a less dominant role. Importantly, the baseline solution time for LG was already considerably lower than the other fault cases, rendering the limited improvement inconsequential to overall real-time feasibility.

Accuracy verification confirms that the tuning strategy preserves simulation fidelity. Voltage profiles, frequency response, and overall system dynamics remained unaffected across all scenarios, validating that the relaxed tolerances remain well within the precision requirements. Therefore, the parameter-tuning approach effectively enhances computational performance for topology-driven events without compromising the dynamic accuracy essential for operator training.

D. Profiling Analysis and Computational Bottleneck Identification

To identify the computational bottlenecks responsible for real-time violations observed in Models 2 and 3, a detailed

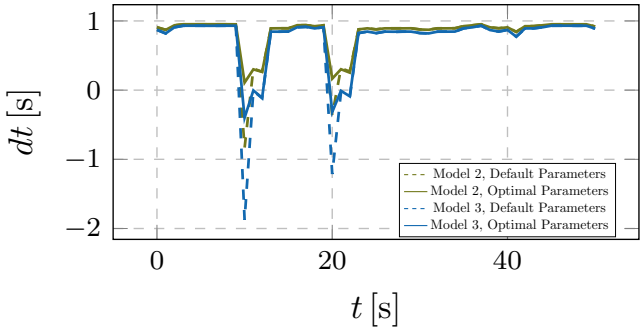

Fig. 4. Optimization of solver parameters and improvement of LBS scenario under Model 1 and Model 2 configurations.

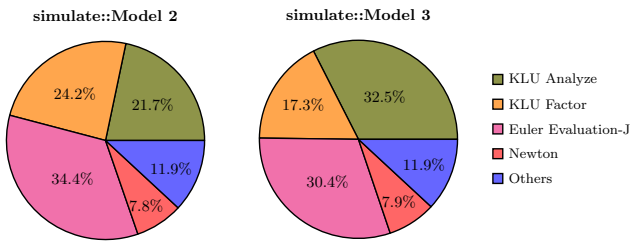

Fig. 5. Profiling analysis showing dominant computational routines within the simulation execution pipeline for LG scenario.

profiling analysis was conducted on the most demanding time step—the first step immediately following each event. This interval exhibits the highest computational burden due to Jacobian restructuring and algebraic-mode transitions, as demonstrated in the previous subsections. The analysis focuses on the `KINSOL` nonlinear solver path, which consumes approximately 91% of the total solving time and thereby represents the dominant computational workload.

Figure 5 presents the percentage cost distribution for Model 2 and Model 3 using the LG scenario as representative case. Within the `KINSOL` solver, four major computational phases were analyzed:

1) `KLU Analyze`, which establishes the Block Triangular Form (BTF) and performs fill-reducing column ordering of the sparse Jacobian structure;
2) `KLU Factor`, which computes the numeric LU factorization to construct the L and U matrices;
3) `Euler Evaluation-J`, which evaluates the Jacobian via partial derivatives from the model and network; and
4) `Newton`, which executes the iteration process including line search, norm updates, and step acceptance.

The computational load distribution reveals significant differences between the two model configurations. For Model 2, the load is distributed relatively evenly among the major phases. The `Euler Evaluation-J` stage dominates at 34.4%, followed closely by `KLU Factor` at 24.2% and `KLU Analyze` at 21.7%. The `Newton` iterations contribute 7.8%, while the remaining 11.9% encompasses miscellaneous solver operations grouped as *Others*. This balanced distribution indicates that no single phase overwhelmingly dominates the computational cost.

In contrast, Model 3 exhibits a marked redistribution of computational effort. The `KLU Analyze` phase becomes the dominant contributor at 32.5%, representing a 50% relative increase compared to Model 2. Correspondingly, `Euler Evaluation-J` decreases to 30.4% and `KLU Factor` to 17.3%, while `Newton` remains relatively stable at 7.9%. This shift toward matrix structure analysis directly reflects the increased complexity.

Examining absolute computation times reveals the precise nature of this bottleneck. The `KLU Factor` duration remains nearly identical between models, and the `Euler Evaluation-J` increases moderately by approximately 140 ms. However, the `KLU Analyze` phase requires 2.3 times longer in Model 3 than in Model 2. Further inspection attributes this increase primarily to the BTF construction. The ordering routine, by contrast, exhibits proportional and expected growth consistent with increased matrix dimensions.

The fundamental cause of this computational escalation lies in the network topology characteristics. Transitioning from Model 2 to Model 3 enlarges and complicates the Jacobian's bipartite sparsity graph through two mechanisms: increased dimensionality (more rows and columns) and the introduction of near-empty or duplicate switch rows when breakers are open. Consequently, the BTF maximum-transversal algorithm must explore longer augmenting paths and resolve more difficult pivot selections, substantially increasing its computational cost. This algorithmic bottleneck leads directly to the observed real-time violations and represents the primary barrier to achieving real-time compatibility with full-fidelity network representations.

VI. Conclusions

This work establishes the fundamental feasibility of an open-source, real-time operator-training platform based on Dynaωo, demonstrating that high-fidelity dynamic simulation can operate within SCADA-constrained refresh budgets on commodity hardware. The evaluation framework, built around a headroom metric, computational environment stabilization, and event-focused profiling, provides a rigorous and reproducible methodology for assessing real-time capability.

The results show that real-time operation for the considered disturbance scenarios and hardware configuration is achievable with the simplified substation representation (Model 1), whereas it is no longer achievable with the more detailed configuration (Model 2), which remains close to feasibility but exhibits isolated constraint violations. The full detailed model (Model 3) significantly increases computational burden, primarily due to Jacobian-structure analysis overhead, challenging sustained real-time operation. Targeted solver tuning improves performance, especially during topology changes, without compromising dynamic fidelity.

Future work will focus on the following directions, grounded in the performance analysis and tuning results of Section V:

1) **Adaptive topology-retention policies**: Develop scenario-aware and operator-action-aware rules that retain only the switching devices necessary for the current training objective, escalating detail on demand. The objective is to preserve Model 2-level controllability where possible (which, after tuning, eliminated violations for LBS/LL) while avoiding the Model 3 overhead that drives real-time violations during topology changes.
2) **Incremental Jacobian-structure updates**: Reduce the `KLU Analyze` cost, which reached 32.5% of `KINSOL` time and 2.3× longer in Model 3, by reusing permutations and warm-starting the BTF maximum-transversal

step across adjacent time steps and known post-event topologies (e.g., line or busbar outages). Caching and incremental updates of the sparsity structure are expected to directly cut the dominant analysis overhead identified in profiling.

3) **Parallelization of Jacobian evaluation and factorization**: Exploit thread-level parallelism where the profile indicates headroom: parallelize Jacobian evaluations (`Euler Evaluation-J` at 34.4% in Model 2 and 30.4% in Model 3) and investigate multi-threaded numeric factorization backends for the LU step. The goal is to shorten the critical path during event steps without increasing iteration counts.
4) **Orchestrator–simulator co-design**: Leverage slack during non-critical intervals to precompute and prewarm structures (e.g., background ordering/factorization for anticipated events), dynamically adjust solver aggressiveness while respecting T_{sync}-level accuracy, and, where permissible, apply elastic refresh policies to time-borrow across steps. This coordination targets robust headroom under the worst event spikes.

Acknowledgment

This work has received funding from the European Commission under the Horizon Europe programme, Grant Agreement No. 101136119 (TwinEU project). Additional support was provided through the TRAISIM project, which is funded by CRESYM.

The authors acknowledge the valuable contributions of Stefanos Eleftheriadis from the Sustainable Power Systems Lab for the implementation and debugging of the simulation framework.

References

[1] J. Bucciero, J. Dodge, R. Gillespie, R. Hinkel, S. Mann, S. Martin, R. Schulte, and D. Hayward, "Dispatcher training simulators: lessons learned," *IEEE Transactions on Power Systems*, vol. 6, no. 2, pp. 594–604, 1991.

[2] M. M. Adibi, "The uses of an operator training simulator for system restoration," in *Power System Restoration: Methodologies & Implementation Strategies*. IEEE, 2000.

[3] ——, "Experiences using the dispatcher training simulator as a training tool," in *Power System Restoration: Methodologies & Implementation Strategies*. IEEE, 2000.

[4] A. Bose, "Computer simulation of power systems for operator training," in *Proceedings of the 24th Intersociety Energy Conversion Engineering Conference*, 1989, pp. 165–169.

[5] R. Kuruneru and A. Bose, "Feasibility study of transient stability analysis for operator training simulators," *IEE Proceedings - Generation, Transmission and Distribution*, vol. 144, pp. 510–514, 1997.

[6] A. Bose, "Real-time power system simulation for training purposes," in *Proceedings of the Fourth IEEE Region 10 International Conference*, 1989, pp. 738–741.

[7] J. Dyer, "Phasor simulator for operator training project," National Energy Technology Laboratory, Tech. Rep., September 2016.

[8] U. Spanel and C. Roggatz, "Dutrain power system handler - the movement of an operator training simulator prototype towards an operational training system," *IFAC-PapersOnLine*, vol. 49, no. 27, pp. 170–177, 2016, iFAC Workshop on Control of Transmission and Distribution Smart Grids CTDSG 2016.

[9] M. Prais, G. Zhang, Y. Chen, A. Bose, and D. Curtice, "Operator training simulator: Algorithms and test results," *IEEE Transactions on Power Systems*, 1989.

[10] NASPI Time Synchronization Task Force, "Time synchronization in the electric power system," North American Synchrophasor Initiative, United States, Technical Report NASPI-2017-TR-001, March 2017, pNNL-26331.

[11] K. Saikawa, M. Goto, Y. Imamura, M. Takato, and T. Kanke, "Real-time simulation system of large-scale power system dynamics for a dispatcher training simulator," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-103, no. 12, pp. 3496–3501, 1984.

[12] S. Gissinger, P. Chaumes, J.-P. Antoine, A. Bihain, and M. Stubbe, "Advanced dispatcher training simulator," *IEEE Computer Applications in Power*, vol. 13, no. 2, pp. 25–30, 2000.

[13] J. Waight, K. Nodehi, M. Rafian, H. Van Meeteren, A. Bose, R. Wasley, E. Stackfleth, and E. Dobrowolski, "An advanced transportable operator training simulator," in *Proceedings of the 1991 Power Industry Computer Application Conference*, 1991, pp. 164–170.

[14] M. D. Anderson, "Power system operator training problems," *IEEE Power Engineering Review*, vol. PER-6, no. 8, pp. 23–24, 1986.

[15] Y. Ping, "A fast load flow model for a dispatcher training simulator considering frequency deviation effects," *International Journal of Electrical Power & Energy Systems*, vol. 20, no. 3, pp. 177–182, 1998.

[16] P. Aristidou, S. Lebeau, L. Loud, and T. Van Cutsem, "Prospects of a new dynamic simulation software for real-time applications on the hydro-quebec system," *CIGRE Science & Engineering*, vol. 4, no. 1, pp. 88–95, February 2016.

[17] S. Vadari, M. Montstream, and H. Ross, "An online dispatcher training simulator function for real-time analysis and training," *IEEE Transactions on Power Systems*, vol. 10, no. 4, pp. 1798–1804, 1995.

[18] X. Lai, H. Chen, A. Dong, C. Lin, W. Jin, M. Ruan, N. Saeed, and Z. Han, "Design of adaptive training control in dispatcher training simulators," in *Proceedings of the Asia Conference on Power and Electrical Engineering*. United States: IEEE, 2023, pp. 1885–1891.

[19] A. Guironnet, M. Saugier, S. Petitrenaud, F. Xavier, and P. Panciatici, "Towards an open-source solution using modelica for time-domain simulation of power systems," in *Proceedings of the 2018 IEEE Power and Energy Society Innovative Smart Grid Technologies Conference Europe*, October 2018.

[20] "Dynawo: A comprehensive power system dynamic simulator," Linux Foundation Energy, 2025, supported by the Linux Foundation Energy initiative. [Online]. Available: https://lfenergy.org/projects/dyna%CF%89o

[21] L. Razik, L. Schumacher, A. Monti, A. Guironnet, and G. Bureau, "A comparative analysis of lu decomposition methods for power system simulations," in *2019 IEEE Milan PowerTech*. IEEE, 2019, pp. 1–6.

[22] P.-M. Gibert, P. Panciatici, R. Losseau, A. Guironnet, D. Tromeur-Dervout, and J. Erhel, "Speedup of emt simulations by using an integration scheme enriched with a predictive fourier coefficients estimator," in *2018 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*. IEEE, 2018, pp. 1–6.

[23] M. Mirz, S. Vogel, G. Reinke, and A. Monti, "Dpsim—a dynamic phasor real-time simulator for power systems," *SoftwareX*, vol. 10, p. 100253, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352711018302760

[24] H. Haugdal, K. Uhlen, and H. Jóhannsson, "An open source power system simulator in python for efficient prototyping of wampac applications," in *2021 IEEE Madrid PowerTech*. IEEE, 2021, pp. 1–6.

[25] TwinEU Consortium, "TwinEU: Developing a concept of pan-European digital twin of the electricity system," https://twineu.net/, 2023, EU Horizon Europe Project.

[26] F. Sabot, S. Ben Mariem, G. Dekeyne, L. Duchesne, A. Bahmanyar, D. Ernst, O. Bretteville, T. Vermeulen, D. Herve, L. Saludjian, and G. Joseph, "Toward a cyber-physical digital twin for operator training: Real-time co-simulation of the french grid," 2025.

[27] R. Podmore, J. Giri, M. Gorenberg, J. Britton, and N. Peterson, "An advanced dispatcher training simulator," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-101, no. 1, pp. 17–25, 1982.

[28] K. Sato, Z. Yamazaki, T. Haba, N. Fukushima, K. Masegi, and H. Hayashi, "Dynamic simulation of a power system network for dispatcher training," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-101, no. 10, pp. 3742–3750, 1982.

[29] 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, September 2016.