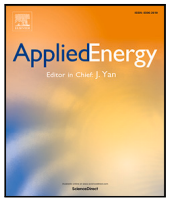
Contents lists available at ScienceDirect

Applied Energy

journal homepage: www.elsevier.com/locate/apenergy



Highlights

An approach for fast cascading failure simulation in dynamic models of power systems

Applied Energy xxx (xxxx) xxx

Sina Gharebaghi, Nilanjan Ray Chaudhuri*, Ting He, Thomas La Porta

- Fast dynamic cascading failure simulation approach using Backward Euler method (BEM).
- Hyperstability of BEM addressed through a new predictor–corrector (PC) approach.
- Adaptive Center of Inertia (COI) frame for faster convergence of Newton iterations.
- Comprehensive comparison of proposed BEM-PC approach against state-of-art.
- BEM-PC replicates cascade path and end-results with $\approx 10-35$-fold average speedup.

Contents lists available at ScienceDirect

Applied Energy

journal homepage: www.elsevier.com/locate/apenergy

An approach for fast cascading failure simulation in dynamic models of power systems☆

Sina Gharebaghi, Nilanjan Ray Chaudhuri *, Ting He, Thomas La Porta

School of Electrical Engineering and Computer Science, The Pennsylvania State University, University Park, PA 16802, USA

ARTICLE INFO

Keywords:
Dynamic model
Cascading failure
Trapezoidal method
Backward Euler method
Hyperstability
Center of Inertia (COI) frame

ABSTRACT

The ground truth for cascading failure in power system can only be obtained through a detailed dynamic model involving nonlinear differential and algebraic equations whose solution process is computationally expensive. This has prohibited adoption of such models for cascading failure simulation. To solve this, we propose a fast cascading failure simulation approach based on implicit Backward Euler method (BEM) with stiff decay property. Unfortunately, BEM suffers from hyperstability issue in case of oscillatory instability and converges to the unstable equilibrium. We propose a predictor–corrector approach to fully address the hyperstability issue in BEM. The predictor identifies oscillatory instability based on eigendecomposition of the system matrix at the post-disturbance unstable equilibrium obtained as a byproduct of BEM. The corrector uses right eigenvectors to identify the group of machines participating in the unstable mode. This helps in applying appropriate protection schemes as in ground truth. We use Trapezoidal method (TM)-based simulation as the benchmark to validate the results of the proposed approach on the IEEE 118-bus network, 2383-bus Polish grid, and IEEE 68-bus system. The proposed approach is able to track the cascade path and replicate the end results of TM-based simulation with very high accuracy while reducing the average simulation time by $\approx 10-35$ fold. The proposed approach was also compared with the partitioned method, which led to similar conclusions.

1. Introduction

Cascading failure study in highly complex dynamical systems like electric power grids is challenging as it demands long-term simulations of models involving solutions of many nonlinear differential and algebraic equations. As a result, it is very difficult to perform statistical analysis of cascading failure using such models. This has led to application of less accurate but computationally manageable quasi-steady-state (QSS) models; see for example [1] for a comprehensive source of references. The objective of this paper is to propose an approach for fast cascading failure simulation that accurately traces the cascade path and lends itself to statistical analyses.

At the outset, we clarify that our goal is to perform deterministic cascading failure analysis, which implicitly assumes that all systems act as expected during the cascade, i.e., potential mistripping of protective relaying and other malfunctions are not considered during the cascade [2,3]. This is in contrast to probabilistic approaches that consider that the evolution of the power system after an initial set of contingencies can follow multiple trajectories, see for example [4,5].

1.1. Literature on dynamic simulation of cascading failure

Unlike their QSS counterpart, the literature on dynamic models of cascading failure is relatively limited — see for example [2–9] and references therein. The papers can be broadly divided into three categories.

(1) *Review- & proposition-type papers:* For example, authors in [6] present a brief review of existing modeling techniques and simulation frameworks for cascading failure analysis, and discuss open questions related to interaction between protection systems and cascading failure. Authors in [7] propose the development of a dynamic power system simulator that has the ability to tune the present direct linear solver, nonlinear solver, and the DAE integrator. In the same line, Ref. [8] suggests a parallelized algorithm for cascade simulations. The focus is to increase the simulation speed through parallelization strategy intended for deployment on a super computer.

(2) *Papers proposing hybrid cascading failure models:* Authors in [4] developed a cascading failure simulation tool called dynamic contingency analysis tool (DCAT), which employs a hybrid approach of simulation that judges the stress of the system and switches between

☆ Financial support from NSF Grant Award ECCS 1836827 is gratefully acknowledged.
* Corresponding author.
E-mail addresses: svg5765@psu.edu (S. Gharebaghi), nuc88@engr.psu.edu (N.R. Chaudhuri), tzh58@psu.edu (T. He), tfl12@psu.edu (T.L. Porta).

https://doi.org/10.1016/j.apenergy.2022.120534
Received 1 July 2022; Received in revised form 10 December 2022; Accepted 16 December 2022

QSS and dynamic simulations. In addition to standard relay modeling, misoperations like stuck breakers are considered and corrective actions in post-transient steady-state conditions are included in the proposed model.

(3) *Papers proposing dynamic cascading failure models:* Paper [5] proposed a two-Level probabilistic risk assessment of cascading outages. Dynamic cascade events are separated into two categories, slow and fast cascade. The paper combines probabilistic simulations for the slow and the fast cascading events using different degree of details in the dynamic models.

Schafer et-al [9] proposed to include network dynamics in the model to study dynamically-induced cascading failure. They represented the synchronous generator dynamics through swing equations. However, swing dynamics might not constitute an adequate representation of the synchronous machines as exciters play an important role in electromechanical oscillations [10].

Ref. [3] proposed a detailed dynamic model for deterministic cascade propagation analysis. The method is tested with randomly selected $N-2$ contingencies. The authors conclude that the load model is very critical in evaluating the risk of cascading failures. It was also shown that the DC QSS model can reasonably approximate the cascade path in the early stages and deviates from the ground truth in later stages.

Paper [2] proposed a multi-time period two-stage stochastic mixed-integer linear optimization model to specify the optimal investment on the network to enhance system's resilience against natural disasters. The model uses dynamic simulations for cascading failure simulation, and the multi-time period restoration, modeled through a DC optimal power flow initialized by the solution of dynamic simulation.

1.2. Gaps in literature

The first category of papers [6–8] either reviewed the state-of-art or made propositions, but no cascading failure simulations were performed in these works.

Although references in the second [4] and the third category [2,3,5] have made valuable contributions, they still suffer from the computational burden faced by the simulation of dynamic models. For example, [3] effectively simulated 88 cases in Polish system out of 1200 that can be called cascades because most (1081) did not have any dependent outages (i.e., any further outages following the initial outages) leading to short simulations, while 31 diverged. Hybrid simulation [4] strategy can reduce simulation time, but may face accuracy issues as it is complicated to switch between dynamic and QSS simulations. At any rate, analysis in [4] starts with dynamic simulations — hence the bottleneck remains.

The reason behind this is the fact that dynamic simulations in these works use a similar structure and the same integration methods as in the conventional planning models. The objective of traditional planning studies is to perform $N-1$ and $N-2$ contingency simulations that normally last up to 30 s. They are computationally very expensive and not suitable for running cascading failure simulations.

1.3. Contribution of our work

We propose a fast time-domain cascading failure simulation approach based on implicit Backward Euler method (BEM) with stiff decay property, in which large time-step can be used to speed up simulations. However, one disadvantage of BEM is the hyperstability issue in case of oscillatory instability that leads to convergence to the unstable equilibrium and produces erroneous results. This has prevented using BEM in power system studies, since it is well-known that oscillatory instability is manifested in many power systems — please see [10,11] for example. The oscillatory instability stems from the lack of damping torque contribution of a generator or multiple generators [10,11] that manifests in the form of local or inter-area oscillations. Therefore, it is imperative that BEM will face hyperstability issues in such cases.

We propose a predictor–corrector (PC) approach to fully address the hyperstability issue in BEM. The predictor in PC-approach identifies possible oscillatory instability using eigendecomposition of the system matrix corresponding to the linear model obtained around the post-event unstable equilibrium, which BEM converges to. It is worthwhile to mention that the system matrix is obtained as a by-product of BEM. Next, the corrector uses right eigenvectors to identify the machine or group of machines participating in the unstable mode. This aids in applying appropriate protection schemes as in ground truth.

We also propose an adaptive center of inertia (COI) reference frame-based approach that leads to a time-invariant generator rotor angle and bus voltage angles, which ensures faster convergence of Newton iterations. Moreover, unlike the traditional COI frame-based method [10,11], our approach seamlessly works for cascading failure simulation leading to island formation. The objective of our proposed method is to trace the cascade path during simulation and reproduce the exact end result of cascade with respect to the ground truth. We use a dynamic model which applies Trapezoidal method (TM) for numerical integration as a benchmark to test our proposed model for cascade simulations. Results on the IEEE 118-bus system, IEEE 68-bus system, and the 2383-bus Polish network show high accuracy and significant speedup in simulation with multi-tier cascading failures. We also show that the proposed approach maintains a significant speedup gain compared to the partitioned approach with an explicit numerical integration method.

The following is a summary of the major contributions made in this paper–

1. Proposing a fast time-domain cascading failure simulation approach based on BEM.
2. Addressing the hyperstability issue of BEM in case of oscillatory instability through a new PC-approach.
3. Proposing a novel adaptive COI-reference frame for cascading failure simulations that seamlessly work during island formation and ensures faster convergence of Newton iterations.
4. Providing comprehensive comparisons of proposed BEM-PC method against simultaneous approach with implicit TM method, partitioned approach with Runge–Kutta (R–K) method, and AC-QSS method.

2. Dynamic simulation: Preliminaries & state-of-art

We first look into the structure of traditional dynamic simulation methods used for power system planning studies. Next, we elaborate on the challenges in using them for power system cascading failure simulation.

2.1. Dynamic simulation preliminaries

Power system's dynamic model is typically represented by a set of nonlinear differential algebraic equations (DAEs) [10]. These equations can be represented in the following compact form after augmenting them with an *implicit* discrete variable z to model relay actions when the related constraint involving function $h(\cdot)$ is violated

$$\dot{x} = f(x, V, z) \tag{1}$$

$$0 = I(x, V, z) - Y_N(z)V \tag{2}$$

$$0 \succ h(x, V, z). \tag{3}$$

Here, $x \in \mathbb{R}^n$ is the state vector consisting of individual device states, $V \in \mathbb{R}^m$ denotes the vector of real and imaginary components of bus voltages, $z \in \mathbb{Z}^p$ is a discrete variable whose elements can assume values 0 or 1 indicating status of circuit breakers operated by relays, $I \in \mathbb{R}^m$ constitutes of real and imaginary components of current injection phasors in buses, $Y_N \in \mathbb{R}^{m\times m}$ is the admittance matrix of

network in its real form (i.e., separating the real and imaginary parts of the equations), and $h : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{Z}^p \rightarrow \mathbb{R}^q$ indicates line currents should be below their ratings and bus voltages below corresponding thresholds, among others. If the inequality constraint (3) is violated, the relevant relay will determine the trip time T_{trip} and start a countdown process. When T_{trip} becomes zero, the corresponding element of z, whose nominal value is 1, also becomes 0. This changes the Y_{bus} and/or the injected current I. If the inequality constraint violation no longer holds before T_{trip} goes to zero, the countdown stops. The details of different types of relay actions have been included in Section 4.1.

Dynamic simulation in power system solves an initial value problem (IVP) on the DAEs (1), (2) with a set of known initial conditions $(x_0, V_0, z_0) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^q$. *For a cascading failure simulation, such IVPs are solved repeatedly following each event, where an event refers to a discontinuity introduced by fault, line tripping, load shedding, and so on.*

There are two philosophies for solving the IVPs in power systems literature — partitioned and simultaneous [10]. In the partitioned approach, the algebraic and the differential equations are solved sequentially, whereas the simultaneous approach uses implicit integration methods, which combines them to pose them as a set of nonlinear algebraic equations. Commercial simulation softwares use second-order Adams–Bashforth (AB2) method [12] – an explicit method – in the partitioned approach. It is well-known, see for example page 861 of [10], that production-grade stability programs use partitioned approach because of programming flexibility, simplicity, reliability, and robustness. However, it has also been mentioned that its main drawback is numerical instability of explicit methods. *Such methods are typically run at a fixed time step to avoid numerical instability issues. On the other hand, simultaneous approach with implicit integration methods can run with a variable time-step.* They are more widely explored in academic research [3], and is followed in our work.

2.2. Simultaneous solution: State-of-art

Here we briefly describe state-of-art on the simultaneous approach where perhaps the most popular implicit integration method is TM [3]. In the context of solving DAEs described in the previous section, note that z is an *implicit variable* and does not appear explicitly in the numerical integration process, except that it brings in discontinuities. To avoid clutter, going forward we will drop z from equations and will describe how discontinuities are handled later. Discretization of (1) using TM results in the following expression

$$F(x_{n+1}, V_{n+1}) = x_{n+1} - x_n - \frac{\Delta t}{2}(f(x_{n+1}, V_{n+1}) + f(x_n, V_n)) \tag{4}$$

where, Δt is the step-size of integration, subscript n corresponds to time instant t_n, and F is the mismatch function for differential equations. The mismatch function for algebraic equations is defined as follows

$$G(x_{n+1}, V_{n+1}) = Y_N V_{n+1} - I(x_{n+1}, V_{n+1}) \tag{5}$$

where, x_{n+1} and V_{n+1} are found by simultaneously solving the following nonlinear algebraic equations

$$F(x_{n+1}, V_{n+1}) = 0, \quad G(x_{n+1}, V_{n+1}) = 0. \tag{6}$$

Typically, Newton's method [13] is used for solving these equations. For the $(k+1)$th iteration of Newton's method, we have

$$\begin{bmatrix} x_{n+1}^{k+1} \\ V_{n+1}^{k+1} \end{bmatrix} = \begin{bmatrix} x_{n+1}^{k} \\ V_{n+1}^{k} \end{bmatrix} + \begin{bmatrix} \Delta x_{n+1}^{k} \\ \Delta V_{n+1}^{k} \end{bmatrix} \tag{7}$$

$$\begin{bmatrix} -F(x_{n+1}^{k}, V_{n+1}^{k}) \\ -G(x_{n+1}^{k}, V_{n+1}^{k}) \end{bmatrix} = \begin{bmatrix} \frac{\partial F}{\partial x_{n+1}} & \frac{\partial F}{\partial V_{n+1}} \\ \frac{\partial G}{\partial x_{n+1}} & \frac{\partial G}{\partial V_{n+1}} \end{bmatrix}_{n+1}^{k} \begin{bmatrix} \Delta x_{n+1}^{k} \\ \Delta V_{n+1}^{k} \end{bmatrix} \tag{8}$$

$$[J] = \begin{bmatrix} \frac{\partial F}{\partial x_{n+1}} & \frac{\partial F}{\partial V_{n+1}} \\ \frac{\partial G}{\partial x_{n+1}} & \frac{\partial G}{\partial V_{n+1}} \end{bmatrix}_{n+1}^{k} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} \tag{9}$$

where, J is the Jacobian matrix. First, Δx and ΔV are calculated using (8), which in turn are used to update x and V through (7). Newton iterations are stopped when $\|[F^T \; G^T]^T\|_\infty \leq \epsilon$, where $\epsilon \in \mathbb{R}_+$ is the tolerance for convergence.

Remarks on state-of-art:

1. *Variants of Newton iterations:* Three popular variants are full Newton's method, dishonest/very dishonest Newton's method (VDHN) and quasi-Newton's method [14].
2. *Jacobian calculation:* Both direct analytical method and difference approximation method [13] have been used.
3. *Solution of linear equation $Ax = b$:* Since the Jacobian is very sparse, solving the set of linear equations (8) in the general form $Ax = b$ takes significant advantage of this aspect during storage and computations. Both direct solution methods like sparsity-oriented triangular factorization [15] and KLU [16], and iterative solutions like Preconditioned Conjugate Gradient (PCG) [17,18], and General Minimal Residual (GMRES) method [18] have been proposed.
4. *Variable time-step:* To speed up the simulation, adaptive time step size control is used based on local truncation error (LTE) [19] that leads to large time steps when solution is not varying rapidly.
5. *Suitability for cascading failure simulation:* Even applying sparse computations for solving (8) and using a variable time-step TM-based solver, the state-of-art suffers from significant computational burden during cascading failure simulation, since they take much longer than typical $N-1$ or $N-2$ contingency simulations that last 30 s or less.
6. *Handling discrete events:* If the integration time step Δt calculated by the variable-step algorithm is more than the time remaining before T_{trip} becomes zero, then Δt is truncated to match the tripping instant. When T_{trip} becomes zero, at $t = t_n$, a discrete event occurs due to relay action, i.e., certain elements of z becomes 0. In this case, the following steps are performed – (a) network configuration is updated and the corresponding Y_{bus} is calculated; (b) V_n is updated by iteratively solving (5) for $t = t_n$. To that end, Y_N is updated if needed and the J_{22} sub-matrix of the Jacobian in (9) is used. Next, updated x_n and V_n are used as the initial guess for $t = t_{n+1}$, i.e., $\begin{bmatrix} \left(x_{n+1}^0\right)^T & \left(V_{n+1}^0\right)^T \end{bmatrix}^T = \begin{bmatrix} (x_n)^T & (V_n)^T \end{bmatrix}^T$ is used. Eq. (8) is then iteratively solved to find $\begin{bmatrix} x_{n+1}^T & V_{n+1}^T \end{bmatrix}^T$; (c) the time-step is reduced to the minimum step-size of Δt_{min} for a pre-defined period of simulation; (d) the IVPs based on the predicted initial conditions are solved using the reduced time-step.

Going forward, we define 'ground truth' as the cascading failure simulation results produced by a benchmark model that uses (a) variable-step TM [3] with $\Delta t \in [0.002, 1]$ s, $\epsilon = 10^{-4}$ that leverages an adaptive COI reference frame-based approach described in Section 4, (b) formulates the Jacobian analytically, (c) applies full Newton iterations, (d) uses sparse objects for storage and calculations, and (e) applies Matlab's [20] most comprehensive inversion routine for solving (8), see flowchart in [21]. The model consists of 4th-order synchronous generator model equipped with the same governors, static exciters, and relays as our proposed model described in the next section, except that the special protection scheme (SPS) is not functional, but measurement-based. The benchmark and the proposed models are built from the first principles in Matlab [20] and MATPOWER [22] is used for power flow solution used during initialization. Simulation is stopped if (i) speed variation of machines in a predetermined window length is below a certain threshold, and no future relay actions are anticipated, or (ii) a complete collapse is observed.

3. Proposed methodology for dynamic simulation of cascading failure

At the outset, we define what is expected out of a dynamic cascading failure simulation model –

1. The model should be able to capture the exact cascade propagation path as in ground truth.

2. The model should give exact end-result of cascade as the ground truth in terms of topology, voltage profile, frequency, and demand served.
3. The model should be computationally efficient, so that statistical analyses can be performed, which is critical for cascading failure studies.

Even though it would be ideal if the dynamic model is able to simulate the exact trajectories of state and algebraic variables of the system as the ground truth and also lends itself to statistical analyses — unfortunately, that has proven to be elusive thus far [2–9]. *We argue that if the above objectives are met at the expense of accurate tracking of trajectories of system variables, it should be sufficient for dynamic cascading simulations without compromising accuracy of statistical analyses.*

To that end, we propose the following (see, Fig. 3) –

(a) A time-domain simulation approach based on a stiff decay integration method. More specific, we apply implicit backward Euler method (BEM) [13] for the simultaneous solution process.

(b) We solve the hyperstability issue of BEM using an eigen analysis-based *predictor–corrector* method, which leverages its stiff-decay and hyperstability properties.

(c) We propose *functional* implementation of SPS against unstable interarea oscillations and generator non-first swing out-of-step protection (e.g., due to an unstable local mode).

(d) We model time-delayed overcurrent (OC), local undervoltage load shedding (UVLS), and generator first swing out-of-step relays. Note that other types of relays can also be modeled in the proposed framework.

(e) We propose an adaptive COI frame-based approach that can seamlessly work during island formation.

3.1. BEM: Absolute stability and stiff decay properties [13]

BEM is derived using a Taylor expansion centered at t_{n+1}, which is a first-order method [13]. Discretizing (1) using BEM results in the following expression

$$F(x_{n+1}, V_{n+1}) = x_{n+1} - x_n - \Delta t f(x_{n+1}, V_{n+1}) \tag{10}$$

3.1.1. Absolute stability property

First, we analyze the absolute stability property of TM and compare it with that of BEM. The standard approach for this is to consider the so-called *test equation* $\dot{x} = \lambda x$, where λ is a complex number denoting the eigenvalue of a system matrix. The *region of absolute stability* is defined as the region in the complex $\lambda\Delta t$-plane such that applying the numerical integration method for the test equation from within this region yields an approximate solution satisfying the absolute stability requirement $|x_{n+1}| \leq |x_n|$. By discretizing the test equation using TM, we have

$$x_{n+1} = \frac{2 + \lambda\Delta t}{2 - \lambda\Delta t} x_n; \quad AF = \left|\frac{2 + \lambda\Delta t}{2 - \lambda\Delta t}\right| \tag{11}$$

where, AF is called amplification factor. Therefore, the region of absolute stability of TM can be obtained by the region that is satisfying $AF \leq 1$, which is the left half of the $\lambda\Delta t$ plane. Similarly, applying BEM to the test equation results in:

$$x_{n+1} = \frac{1}{1 - \lambda\Delta t} x_n; \quad AF = \left|\frac{1}{1 - \lambda\Delta t}\right| \tag{12}$$

Therefore, the region of absolute stability of BEM is the entire left half of the $\lambda\Delta t$ plane in addition to the entire right half plane outside the unit circle centered at $(1, 0)$. As shown in Fig. 1, the regions of absolute stability in gray indicates that both TM and BEM are numerically *A-stable* [13].

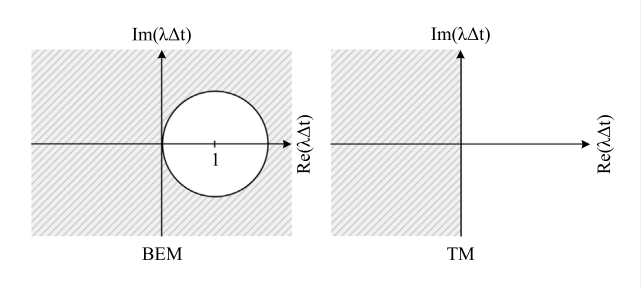


Fig. 1. Absolute stability regions of BEM (left) and TM (right) shown in gray.

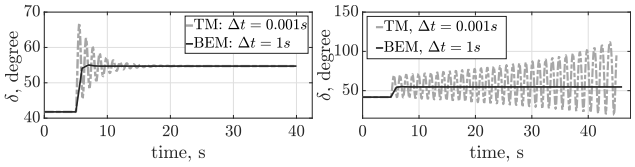


Fig. 2. Rotor angle time-domain plot in SMIB system after one line outage at $t = 5$ s; BEM vs TM in Left: Stable case, Right: Oscillatory instability case (hyperstability problem of BEM).

3.1.2. Stiff decay property

In line with our argument presented earlier, in the dynamic simulation of cascading failure, one might not be interested in detailed transient oscillatory behavior of the system as long as the expectations are met. In this regard, using large time steps would be desired. However, the integration method should be robust enough to tolerate the large steps. According to [13], when $\Re(\lambda\Delta t) \to -\infty$, for BEM we have $\frac{1}{1-\lambda\Delta t} \to 0$, however, for TM we have $\frac{2+\lambda\Delta t}{2-\lambda\Delta t} \to -1$. This property in BEM is called *stiff decay*, representing ability of BEM in taking large steps to ignore fast oscillations in the dynamic model. On the other hand, one should not expect TM to act like integral methods with stiff decay property. This is due to the fact that the fast mode components of local errors for large time steps get propagated throughout the simulation interval [13].

3.1.3. Hyperstability issue of BEM

It is clear that the stiff decay property of BEM can be used to our advantage for dynamic simulations of cascading failure as it helps us take large time steps for ignoring fast oscillations and obtaining a coarse picture of the desired trajectories. However, BEM is never used in dynamic simulations of power system due to the *hyperstability* problem [23].

When a numerical integration method solves the differential equations of an unstable system and produces a stable response, then such a problem is called Hyperstability. This can be viewed from the absolute stability region of Fig. 1 for BEM satisfying $\Re(\lambda\Delta t) > 0$ and $(\Re(\lambda\Delta t) - 1)^2 + (\Im(\lambda\Delta t))^2 > 1$. It corresponds to the right half plane outside the unit circle of the left subfigure. The practical implication of this is that BEM is not able to diagnose oscillatory instability if $\lambda\Delta t$ satisfies the above constraints.

Fig. 2 compares the performance of BEM and TM in a single machine (represented by classical model) infinite bus (SMIB) system [10] after tripping one of the double-circuit lines at $t = 5$ s. The left and the right subplots represent a stable and an unstable case, respectively — the latter is simulated by a negative damping factor. In both the scenarios, traditional model with TM is simulated with $\Delta t = 0.001$ s, and BEM uses much larger time step of 1 s. It can be seen that for the stable case, the stiff decay property allows BEM to obtain the exact final result as TM while producing a coarse trajectory. For the unstable case however, the hyperstability problem of BEM is evident, where it converges to the unstable equilibrium point. Next, we will address the hyperstability problem of BEM in detail.

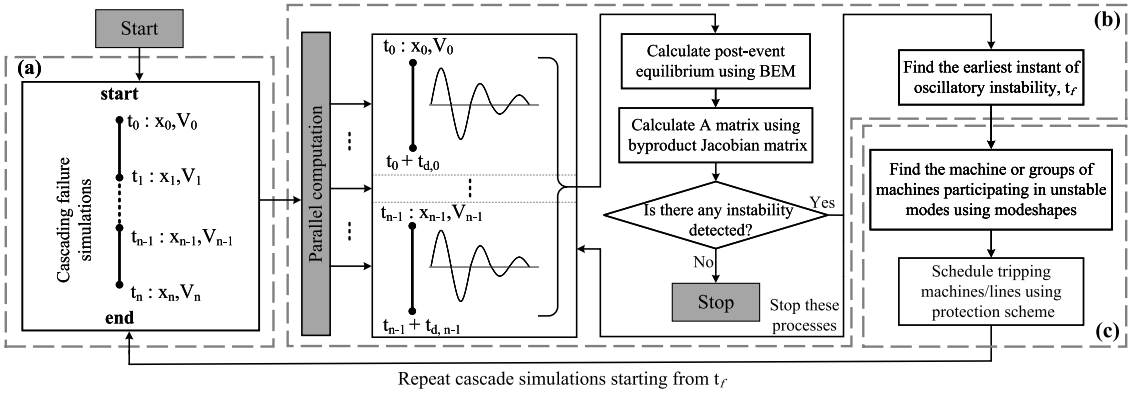


Fig. 3. Flowchart of proposed BEM with predictor–corrector (BEM-PC) approach. $t = t_i, i \in \{0, 1, \ldots, n-1\}$: instants of tiers of cascade. t_0 and t_n: instants of initial disturbance and end of simulations, respectively. Processes that can be run using parallel processors are indicated. (b): Predictor subprocess. (c): Corrector subprocess. For fair comparison with traditional methods, BEM-PC in this work is applied in a fully serial fashion.

3.2. Addressing hyperstability problem of BEM using predictor–corrector approach

We propose a predictor–corrector (PC) approach to tackle the hyperstability problem in BEM, which is shown in a flowchart in Fig. 3. In this flow chart, there are four key functions that are being performed in a *serial-parallel* process.

3.2.1. Cascading failure simulation subprocesses (a)

In this subprocess, we run the cascading failure simulation using variable-step BEM, where OC, UVLS, and generator first swing out-of-step relays are modeled. The stopping criterion for BEM is similar to TM as described earlier, i.e., simulation is stopped if (i) speed variation of machines in a predetermined window length is below a certain threshold and no future relay actions are anticipated, or (ii) a complete collapse is observed. For step-size control in BEM, we use a different method than TM,

$$\Delta t_{n+1} = \Delta t_n \frac{\tau}{\|F_x^0\|_\infty}; \quad \Delta t_k \in [\Delta t_{min},\ \Delta t_{max}] \tag{13}$$

where, $\|F_x^0\|_\infty$ shows the largest component of the first mismatch vector and τ is a hyperparameter to be tuned, see [23] for explanation. Note that immediately following each event, we run simulation with $\Delta t = 0.002$ s for a pre-determined k steps. This ensures that the non-oscillatory instability is captured. Moreover, if Newton iterations take more than r iterations for convergence at a time instant, then we decrease time-step to $\Delta t = 0.002$ s.

During the course of such a simulation, the instants of tiers (events that alter the topology of the system) of cascade are marked by time variable $t = t_i$ in Fig. 3. Corresponding to each such instant, we get the values of x and V as $[t_i : x_i,\ V_i]$. Note that this subprocess runs in a *serial* manner to solve a sequence of IVPs described in Section 2.

Following each such instant, there could be four broad types of unstable scenarios so far as voltage, angle, and frequency stability are concerned – (1) local voltage instability, (2) frequency instability, (3) non-oscillatory angle instability, and (4) local/interarea oscillatory angle instability. Except the last phenomena, BEM does not face issues in capturing the others.

As discussed earlier, due to the hyperstability issue, BEM may converge to the unstable equilibrium following the fourth category of instability. This in turn can deviate the cascade propagation path from ground truth. *Our goals are to identify the earliest tier of onset of such instability, and execute appropriate protective actions that will be taken in the ground truth.*

3.2.2. Predictor subprocess (b)

This subprocess shown in Fig. 3 runs after subprocess **(a)** ends. It constitutes running multiple simulations and calculations that are independent and thus *parallelizable*. The following steps are taken –

i. *Solving independent IVPs for short duration:* As subprocess **(a)** spits out $[t_i : x_i,\ V_i]$ data, we solve IVPs with initial values $(x_i,\ V_i)$ that can be run in the ith parallel processor using variable-step BEM. The simulation for the ith independent IVP is stopped if the speed variation of machines in a predetermined window length is below a certain threshold. As shown in Fig. 3, $t_{d,i}$ is the time elapsed since the beginning of such a simulation when this stopping criterion is met. In this period, we do not consider any event including relay actions.

ii. *Calculate system matrix for model linearized around post-event equilibrium:* Solving variable-step BEM within each parallel processor allows the trajectories to reach the post-event equilibrium – ***more critical, the post-event unstable equilibrium point due to BEM's stiff-decay and hyperstability properties***. We calculate the system matrix (A matrix) of the model linearized around this equilibrium as a *byproduct* of BEM-based simulation using the elements of the Jacobian matrix as follows

$$A = P_{11} + P_{12} P_{22}^{-1} P_{21} \tag{14}$$

where,

$$P_{11} = \tfrac{1}{\Delta t}(I - J_{11}); \quad P_{12} = -\tfrac{1}{\Delta t} J_{12}; \quad P_{21} = -J_{21}; \quad P_{22} = J_{22} \tag{15}$$

where, I denotes the identity matrix and Δt is the time step at the end of duration t_{di}.

iii. *Eigendecomposition of A matrix:* Eigendecomposition of A matrix is performed to detect oscillatory instability. For large systems, one can use selective unstable eigenvalue and corresponding right eigenvector calculations using the S-method [24] or refer to relatively recent works on this topic [25,26]. The earliest event and corresponding time instant, say t_f is identified at which the instability occurs.

3.2.3. Corrector subprocess (c)

If any oscillatory instability is detected, its origin can be found from participation factors [10]. Assuming the typical case of instability from electromechanical modes, we find the machine or groups of machines participating in these modes using their speed modeshapes calculated in subprocess **(b)**. We schedule *functional implementation* of the predetermined protective action (e.g., out-of-step generator tripping or SPS action) that will take place following $t = t_f$ in the ground truth after a designed delay.

3.2.4. Restart subprocess (a)

As shown in Fig. 3, with the knowledge of pre-determined protection action to be taken, we re-initiate the solution of IVPs at $t = t_f$ with initial states x_f, V_f and perform protection actions after a pre-defined delay and continue solving the subsequent IVPs.

The above steps will be repeated until no instability is detected in the predictor subprocess.

In summary, in absence of oscillatory instability during cascading failure, the Predictor–Corrector (PC) subprocesses do not lead to any further protection actions. In presence of oscillatory instability BEM leads to wrong results, as it fails to capture such instabilities due to hyperstability issue. In this case, PC acts to detect such instabilities and takes protection actions that should have taken place in ground truth. This corrects the cascade propagation path.

3.3. Discussion on computational efficiency of BEM-PC

In this Section, we present a brief *qualitative* discussion on the computational efficiency of BEM-PC compared to TM using *logical arguments*. To this end, first we discuss why it is difficult to analytically compare the computational complexity of BEM-PC against TM.

In both TM and BEM-PC the Newton iterations for each time instant like t_{n+1} involve inversion of the Jacobian matrix, which is sparse. The major portion of CPU time for each Newton iteration is related to Jacobian matrix inversion. For both TM and BEM-PC methods, we use '\' or '*mldivide*' command in Matlab [21] for solving (8) in the well-known form of linear equation $Ax = b$. In this case, Matlab automatically utilizes *Unsymmetric MultiFrontal PACKage with automatic reordering (UMFPACK)* [27], which can be identified using the '*spparms*' command [28]. *UMFPACK* uses unsymmetric-pattern multi-frontal method and direct sparse LU factorization. When inverting the $p \times p$ dense matrix, the order of complexity in such algorithms, e.g., recursive block LU algorithm is $\mathcal{O}(p^3)$ [29]. For the sparse matrix inversion, the complexity typically depends upon the number of nonzero entries, rather than the matrix dimension. *To the best of our knowledge, there is no exact general expression describing the order of complexity $\mathcal{O}$ for sparse matrix inversion using UMFPACK.*

In the following, we investigate the computational complexity of the subprocesses in TM and BEM-PC, and difficulties in quantitative and analytical comparison of these methods.

1. *Complexity analysis of TM:* The cascading failure simulations (similar to subprocess (a) in Fig. 3) is the only set of calculations where TM is applied. TM involves Newton iterations for each instance where the major portion of CPU time is spent in the inversion of Jacobian matrix with size $(n+m)\times(n+m)$, where n and m are the length of the state vector consisting of individual device states and the length of the vector of real and imaginary components of bus voltages, respectively. *UMFPACK* is used for Jacobian matrix inversion in each iteration of Newton's method.
2. *Complexity analysis of BEM-PC:* According to Fig. 3, in BEM-PC, the predictor–corrector (PC)-approach has to be performed in addition to the cascading failure simulations in subprocess (a). The predictor subprocess (b) is *parallelizable* and has three main tasks:
   (b1) Calculation of the post-event equilibrium,
   (b2) Calculation of $A \in \mathcal{R}^{n\times n}$ matrix from (14), and
   (b3) Eigendecomposition (i.e., calculation of eigenvalues and modeshapes) of A matrix.
   Here is the list of subprocesses in BEM-PC that require inversion of the Jacobian or the subjacobian J_{22}:
   - Subprocesses **(a), (b1)**: *UMFPACK* is used in Newton iterations for Jacobian matrix inversion of size $(n+m)\times(n+m)$.
   - Subprocess **(b2)**: During system matrix calculation, *UMFPACK* is used for inversion of the subjacobian J_{22} of size $m \times m$ in (14).

   In addition, in subprocess **(b3)**, we use the '*eig*' command in Matlab, which has a degree of complexity $\mathcal{O}(n^3)$.
3. *Comparison of complexity in TM and BEM-PC: Difference between efficiency of BEM-PC and TM comes from the ability to adopt large time step-sizes in BEM-PC method.* Although, subprocess (b) containing (b1), (b2), and (b3), and subprocess (c) runs in BEM-PC in addition to the routine cascading failure simulation (subprocess (a)); since BEM-PC is able to converge to a comparatively large step size Δt much sooner than TM, we see significant speedup using BEM-PC w.r.t. TM.
4. *Difficulties in quantitative comparison:* Here we mention some of the difficulties of quantitative comparison on the efficiency of BEM-PC vs TM.
   - Even if we assume that the CPU time of other arithmetic operations in the Newton iterations compared with that of matrix inversion is negligible, the order of complexity $\mathcal{O}$ for the sparse matrix inversion in *UMFPACK* is not quantifiable.
   - The size of the Jacobian matrix in both TM and BEM-PC and subjacobian J_{22} in BEM-PC are time variant from one island to another during cascading failure. In addition, there is no predictable pattern for this change upon different contingencies. Therefore, comparing the order of complexities ($\mathcal{O}$) of TM against BEM-PC is very difficult.
   - Time step Δt is continuously changing in both methods *that use different variable step size adaptation approaches*, with no predictable pattern either during cascading failure simulations for a generic contingency, or under different contingencies. Since Δt determines how many times Newton iterations are initiated, it is very difficult, if not impossible to quantify overall complexity.

Since it is difficult to analytically quantify the computational efficiency of BEM-PC against TM, we present a *qualitative* assessment through logical arguments as follows–

1. Subprocess (a) in Fig. 3 runs with variable time algorithm (13), which allows significantly larger step size Δt compared to that allowed in LTE-based variable step TM mentioned before. This is possible due to the *stiff-decay* property of BEM as described in 3.1. As a result, this subprocess can run much faster than TM. We have presented a statistical analysis of CPU time for running this subprocess compared to TM in Section 5.3. In addition, we have also shown variation of time-step Δt for TM and BEM in a typical case.
2. Subprocess **(b1)** can be performed extremely fast with large Δt. In subprocess **(b2)**, the A matrix calculation is a by-product and needs inversion of J_{22}, which is a *highly sparse* matrix. We leverage sparse computation for this, as described earlier. Also, there are highly efficient routines that can be used for eigendecomposition in subprocess **(b3)**. Further, note that once any unstable mode is found, A matrix does not need to be calculated for the remaining equilibria. We have demonstrated statistical analysis of CPU time needed for these individual steps in Section 5.3 *without parallelization.*
3. Subprocess (c) needs minimal computation as it is based on lookup table for enacting SPS action.
4. Clearly, the computational efficiency of BEM relies on the fact that for a typical power system, a relatively small fraction of cascade simulations will lead to oscillatory instability, and therefore needs to re-run subprocess (a).

We have performed exhaustive comparison between the proposed method and TM through statistical analysis of CPU time needed for different subprocesses within BEM-PC in Section 5.3, which support the above-mentioned arguments. In addition, a comparison with partitioned approach with fixed time-step-based explicit integration leads to similar conclusions.

Remarks:

1. At a fundamental level, the proposed approach will be able to speed up any transient stability simulation of power systems with accurate end results. However, due to short-term nature of typical transient stability simulations the speed gain would be rather limited.
2. In contrast, cascading failure simulations may run for a much longer time, lead to formation of multiple islands, and show response across different time-scales throughout the process. BEM can run with a larger integration time-step than TM during most of the simulation period, which makes it ideally suited for longer term simulations. This argument becomes even more relevant since the proposed BEM-PC approach requires additional computations due to the prediction and the correction steps.

4. Modeling and adaptive COI-frame-based approach

In this section, first we describe the models of power system components (e.g., generators and transmission lines) and relays used for protection, which determine the DAEs to be solved by both TM and BEM-PC. Next, we propose an adaptive COI-frame-based approach for faster convergence of Newton iterations in (8), which is used for both TM and BEM-PC. Therefore, adaptive COI frame is used when subprocesses (a) and (b) of BEM-PC in Fig. 3 solve the DAEs using BEM.

4.1. Component and relay action models

We consider a 4th-order synchronous generator model (states E'_q, E'_d, δ, $\Delta\omega$) with a first-order governor and static exciter models [11]. Both static constant power and dynamic loads in the form of synchronous condensers were considered. The synchronous condensers have similar models as generators, except that they do not have governors.

We have considered certain relay actions in our model. For example, undervoltage load shedding (UVLS) relays are associated with individual buses connected to static loads, measuring an average voltage magnitude in a window of length T_w^{UVLS} s. The relay trips λ fraction of load if the average voltage magnitude of bus stays below threshold V_{th} for T_{tp}^{UVLS} s. The maximum number of times the UVLS relays are allowed to shed a specific load is K_{max}^{shed}.

The overcurrent (OC) relays measure an average magnitude of current flow in the lines in a T_w^{OC} s window. The trip delay for an overloaded line is $T_{tp}^{line} = \frac{0.14}{(\frac{|I|}{I_c})^{0.02}-1}$ where, $|I|$, and I_c are average current flow in the present window and line heating limit, respectively. The window for OC relays is updated once in every second. Due to probable large amplitude oscillations in the line flows immediately following an event, OC relays use the latest pre-event trip delays till 1 s following the event and then starts updating it. In addition, generator out-of-step relay action trips a machine with non-oscillatory instability. We have considered a specific type of pre-designed SPS action on oscillatory instability involving multiple machines as described in Section 5.2. For both TM and BEM-PC, if the time remaining till the earliest scheduled trip instant by relays, Δt_{trip}^{relay} is less than Δt_{n+1} suggested by variable-step algorithms, we consider $\Delta t_{n+1} = \Delta t_{trip}^{relay}$.

Remarks on Modeling:

1. Both BEM-PC and TM solve IVPs of the same DAEs, but the former can reach the post-disturbance equilibrium faster. This is possible because BEM-PC has stiff decay property that enables it to use a variable integration time-step with a larger step-size than TM during most of the simulation period.

2. In certain cases the cascading process may include mid/long term stability issues involving aspects like boiler dynamics and long-term frequency instability. This however can easily be integrated in our proposed framework, and is not a limitation. Due to its stiff-decay property, BEM is ideally suited for longer term simulations and gives more benefit.

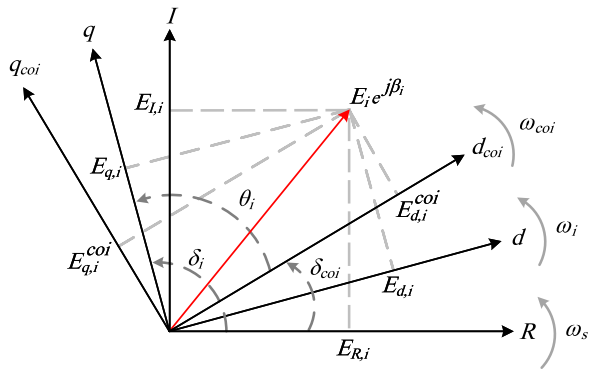


Fig. 4. Three different reference frames.

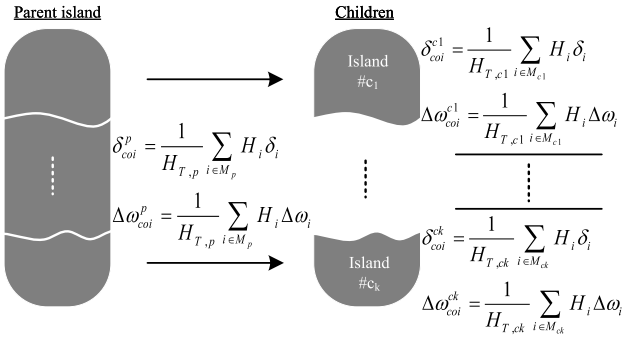


Fig. 5. Island formation during cascading failure: superscripts p for parent, ci for ith child.

4.2. Adaptive COI-frame-based approach

In both TM and BEM-PC approaches, instead of network reference frame ($R-I$ frame) rotating at synchronous speed ω_s, we project all phasors of an island on the COI frame ($d_{coi}-q_{coi}$) [10,11] rotating at $\omega_{coi} = \frac{1}{H_T}\sum_{i\in M} H_i\omega_i$, where ω_i and H_i are the rotor speed and inertia constant of the ith machine, $H_T = \sum_{i\in M} H_i$, and M is the set of indices indicating machine numbers in the corresponding island. Fig. 4 shows the terminal voltage phasor of the ith machine projected on different reference frames including the machine's own $d-q$ frame. The use of COI frame leads to rotor angle $\theta_i = \delta_i - \delta_{coi}$ (where, $\delta_{coi} = \frac{1}{H_T}\sum_{i\in M} H_i\delta_i$), which is constant in *steady state* under off-nominal frequency. This helps in efficient convergence of Newton iterations in (8), which is common knowledge. What is challenging however, is adapting this framework for a cascading scenario that leads to formation of multiple children from a parent island, see Fig. 5.

Challenge during formation of multiple islands: Let $t = t_n$ be the last instant when the parent island was intact and $t = t_{n+1}$ be the first post-islanding instant forming children as in Fig. 5. We need to use dynamic states x_n to predict x_{n+1} for both TM (4) and BEM-PC (10), and in addition V_n is required for TM. Since $M_p \neq M_{ci} \forall i$, see Fig. 5, the converged x_n values corresponding to the pre-islanding instant cannot be used. To be more specific, the challenge comes from the fact that unlike the $R-I$ frame which can be applied for any island, the COI

frames are locally applicable to individual islands (Fig. 5). To solve this problem, we propose an adaptive COI frame-based approach, which is described next.

Proposed approach: We perform the following steps to calculate $\begin{bmatrix} x_{n+1}^T & V_{n+1}^T \end{bmatrix}^T$ within any child island $\#ci$ –

Step (I): Since $R-I$ frame is universal, we calculate $\delta_n, \Delta\omega_n \in \mathbb{R}^{|M_{ci}|}$ as

$$\delta_n = \theta_n + \delta^p_{COI\_n}; \quad \Delta\omega_n = \Delta\bar{\omega}_n + \Delta\omega^p_{COI\_n} \tag{16}$$

where, $\theta_n, \Delta\bar{\omega}_n \in \mathbb{R}^{|M_{ci}|}$ are rotor angle and speed deviation vectors in island $\#ci$ w.r.t. the parent's COI frame.

Step (II): This is the step where we *adaptively* change the COI frames from parent to child for each island. To that end we update $\theta_n, \Delta\bar{\omega}_n \in \mathbb{R}^{|M_{ci}|}$ from the parent's COI frame to the COI frame of island $\#ci$ (Fig. 5) as

$$\theta^{ci}_n = \delta_n - \delta^{ci}_{COI\_n}; \quad \Delta\bar{\omega}^{ci}_n = \Delta\omega_n - \Delta\omega^{ci}_{COI\_n} \tag{17}$$

where, in (17), $\delta^{ci}_{COI\_n} = \frac{1}{H_{T,ci}}\sum_{i\in M_{ci}} H_i\delta_{i_n}$ and $\Delta\omega^{ci}_{COI\_n} = \frac{1}{H_{T,ci}}\sum_{i\in M_{ci}} H_i\Delta\omega_{i_n}$. Note that the other machine states, exciter states, and governor states are not changed. The device states along with $\delta^{ci}_{COI\_n}$ and $\Delta\omega^{ci}_{COI\_n}$ constitute the updated state vector $x_n \in \mathbb{R}^{6|M_{ci}|+2}$ for island $\#ci$.

Step (III): We update V_n within island $\#ci$ by iteratively solving (5) for $t = t_n$ with updated states x_n from Step (II). To that end, we update Y_N if needed and make use of the J_{22} sub-matrix of the Jacobian in (9).

Step (IV): In the final step, we use updated x_n and V_n as the initial guess for $t = t_{n+1}$, i.e., we use $\begin{bmatrix} \left(x^0_{n+1}\right)^T & \left(V^0_{n+1}\right)^T \end{bmatrix}^T = \begin{bmatrix} (x_n)^T & (V_n)^T \end{bmatrix}^T$. We iteratively solve (8) to find $\begin{bmatrix} x_{n+1}^T & V_{n+1}^T \end{bmatrix}^T$ in island $\#ci$.

Note that steps (II) and (III) constitute a *sequential approach* within the simultaneous solution process. For $t > t_{n+1}$, the adaptive COI-frame based approach *is identical with standard COI-based approach in the island's own COI frame until it further breaks into multiple islands.*

4.3. Comparison of proposed adaptive COI frame-based approach with network (R-I) frame-based approach

To demonstrate the advantage of the proposed adaptive COI frame-based approach and explain the reason behind it in more detail, we contrast the cascading failure simulation results of TM with adaptive COI reference frame against TM with network (R-I) reference frame. In this regard, we investigate a contingency in IEEE 118-bus system where cascade is triggered with 2 node outage at $t = 3$ s. Figs. 6 and 7 compare the time-domain cascading failure simulation results obtained using these two reference frames. In Fig. 6, left subplots, (a) and (b), illustrate dynamic simulations with R-I reference frame and right subplots, (c) and (d), are for the simulations with adaptive COI reference frame. In this figure, ω shows machine speed, and δ and θ denote rotor angles of machines in R-I frame, and adaptive COI frame, respectively. To avoid clutter in Fig. 6, we plot machine speeds and rotor angles of only two generators ($G5$ and $G53$) among all 54 machines in IEEE 118-bus system. In addition, Fig. 7 plots real and imaginary terms of voltages as well as voltage magnitudes in two selected buses in network R-I reference frame (left subplots, E_R, E_I, and $|E|_{RI}$), and adaptive COI frame (right subplots, E^{coi}_d, E^{coi}_q, and $|E|^{coi}$).

The subplot (a) in Fig. 6 shows that the system settles at a new frequency greater than the nominal frequency 60 Hz, inherently because of primary frequency response in the post-disturbance situation. This makes rotor angles in the R-I frame to increase with time (subplot (b) in Fig. 6). In addition, as it is shown in the subplots (a), and (b) of Fig. 7, real and imaginary terms of nodal voltages oscillate with time. On the other hand, simulation based on adaptive COI frame in Fig. 6 results in time-invariant steady-state for rotor angles (subplot (d) in Fig. 6), and for real and imaginary terms of bus voltages w.r.t. adaptive COI

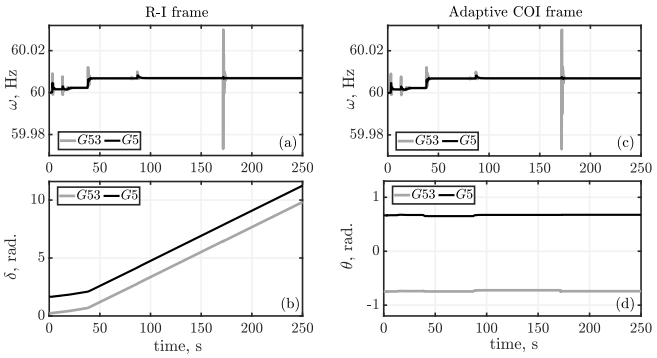


Fig. 6. Selective machine speeds and rotor angles in two different reference frames. Left: Real-Imaginary reference frame. Right: adaptive COI-reference frame. For COI frame ω is calculated as $\omega = \bar{\omega} + \omega_{coi}$, where $\bar{\omega}$ and ω_{coi} denote the machine speed w.r.t. COI frame and speed of COI reference frame, respectively.

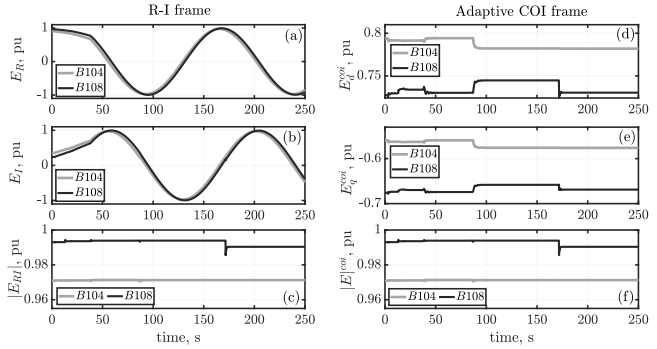


Fig. 7. Real and imaginary terms of nodal voltages and voltage magnitudes of representative buses in two different reference frames. Left: Real-Imaginary reference frame. Right: adaptive COI-reference frame.

frame (subplots (d), and (e) in Fig. 7). Fig. 7 shows that magnitudes of voltages are identical for both R-I and adaptive COI reference frames.

Now, let us explain the implication of these observations on the Newton iterations as in (8). Newton iterations for instant $n+1$ starts with an initial guess for states and algebraic variables ($x_0^{n+1} = 2x_n - x_{n-1}$, and $V_0^{n+1} = \frac{(V_n)^2}{V_{n-1}}$). *The more precise the initial guess, the higher is the possibility of faster convergence of Newton iterations. Based on Figs. 6 and 7, the variables to be calculated using Newton iterations are time-invariant under steady-state when adaptive COI-reference frame is applied as opposed to the R-I frame. This leads to a more precise initial guess for Newton's method and a faster convergence of iterations. Moreover, this time-invariance allows choice of larger integration time-steps Δt without sacrificing accuracy of the initial guess.*

To illustrate this, Fig. 8 compares the cumulative number of Newton iterations and time step Δt of main island for dynamic simulations with adaptive COI and R-I reference frames. This figure clearly demonstrates that the simulation with adaptive COI reference frame demands much less number of Newton iterations and results in using larger time step size Δt. The TM simulation with adaptive COI reference frame is 3.3 times faster than the TM simulation with R-I reference frame. Both simulations produce identical end results for cascade.

Therefore, adaptive COI reference frame improves the convergence of Newton iterations and allows larger time step sizes for simulations. Although we did the comparison using TM, the same conclusion is expected for BEM-PC.

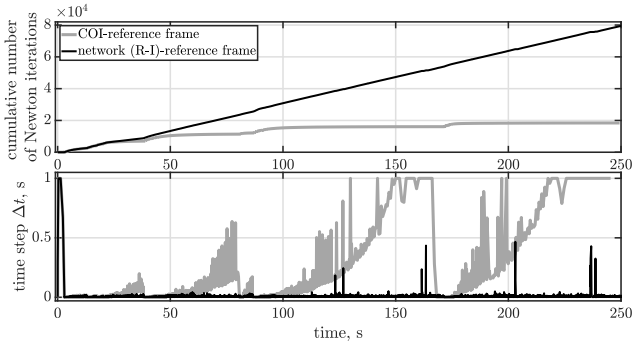


Fig. 8. Top: Cumulative number of Newton iterations over time for TM based on adaptive COI-reference frame and network (R-I)-reference frame. Bottom: Adaptation of integration time step Δt in TM method based on adaptive COI-reference frame, and network (R-I)-reference frame. The TM simulation with adaptive COI frame is 3.3 times faster than the TM simulation with R-I frame.

5. Case studies

The IEEE 118-bus system and the Polish network during winter 1999–2000 peak condition [22] are studied here to contrast our proposed approach (called BEM-PC hereafter) and the traditional approach (called TM from now). The IEEE 68-bus system is also studied, which will be introduced later. The IEEE 118-bus system consists of 118 buses, 54 machines, and 186 branches. The Polish system is a large-scale network with 2383 buses, 327 machines, and 2896 lines. We synthetically generate dynamic data, for these models. For the IEEE 118-bus and the Polish systems, cascades are triggered with 2 and 3 initial node outages, respectively, which are sufficient to create long term cascading sequences in these networks. For each system, 500 Monte-Carlo runs are performed with random selection of initial node outages. For BEM-PC we have used $\Delta t_{min} = 0.02$ s, $\Delta t_{max} = 0.4$ s, $k = 6$, $r = 7$, $\epsilon = 10^{-4}$, and maximum allowable Newton iterations is 10. For relays, $T_w^{UVLS} = 3$ s, $T_{tp}^{UVLS} = 3$ s, $\lambda = 25$ %, $v_{th} = 0.8645$ pu for IEEE 118-bus system, and 0.75 pu for Polish system, $K_{max}^{shed} = 5$, and $T_w^{OC} = 1$ s have been used.

Note that the following results implement the Predictor subprocesses **(b)** of BEM-PC in Fig. 3 in a *serial* fashion. *Hence the speedup obtained is a conservative estimate of what can be obtained with parallelization of the Predictor.* For IEEE 118-bus system, the simulations were run in AMD Ryzen 7 3800X CPU with 32 GB RAM and for Polish system 4 servers with 2.2 GHz Intel Xeon Processor, 24 CPU/server, and 128 GB RAM in PSU's ROAR facility [30] were used.

5.1. Monte-Carlo simulation

5.1.1. IEEE 118-bus system

Figs. 9 and 10 compare how often the total demand loss and line outages at the end of cascade are above a particular level for TM and BEM-PC. The analysis includes initial node outages. The top two zoomed subplots show that there are small differences between BEM-PC and TM for cases with (15%–25%) demand loss and cases with (44 – 60) line outages. Nonetheless, the results indicate a very close match between the end results of cascade.

Table 1 compares the accuracy of BEM-PC with respect to TM and shows that the average error at the end of cascade in states (connected vs disconnected) of buses, machines, and lines which are small fractions of the corresponding total numbers. Similarly, the central tendency measures of maximum errors in voltage magnitudes, angles, and frequency are very small. Although there are some outliers causing an increase in the average error values, for almost all of the cases BEM-PC is able to replicate the exact end-result of TM.

The R values in the table show path agreement measure [3] between BEM-PC and TM based on dependent branch outages, where both

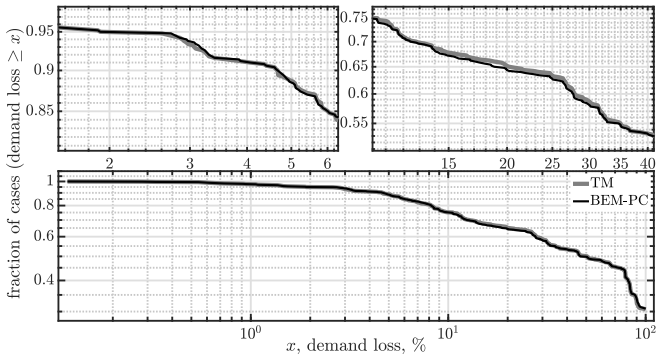


Fig. 9. Fraction of cases with % demand loss ≥ x at the end of cascade: IEEE 118-bus system.

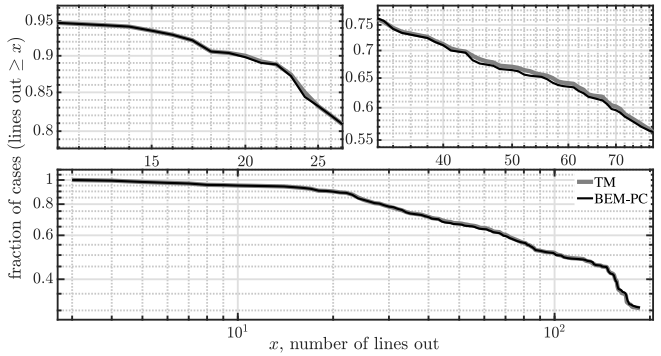


Fig. 10. Fraction of cases with line outage ≥ x at the end of cascade: IEEE 118-bus system.

Table 1
(a) End of cascade error, (b) path agreement measure, and (c) run time in TM w.r.t. BEM-PC: IEEE 118-bus system.

		Mean	Min	Max	Median
Error in State of	buses	0.6460	0	77	0
	machines	0.2940	0	36	0
	lines	0.8960	0	114	0
Maximum Error in	$\|v\|$, pu	0.0024	0	0.0741	7.9e−7
	$\angle v$, deg.	0.2874	0	14.5731	1.0e−4
	f, Hz	0.0442	0	2.3127	1.1e−6
R		0.9911	0.25	1	1
Runtime ratio		**9.9575**	**0.3403**	**60.4361**	**9.0554**

models are subject to the same set of initial outages $C = \{c_1, c_2, \ldots\}$. If contingency c_i results in the set A_i of dependent line outages in the first model and the set B_i of dependent line outages in the second model, then R is defined as follows [3],

$$R = \frac{1}{|C|}\sum_{i=1}^{|C|}\frac{|A_i \cap B_i|}{|A_i \cup B_i|} \tag{18}$$

where, $R = 1$ indicates a complete match between cascade paths from two models following all contingencies.

Based on the central tendency measures of R from Table 1 and its standard deviation being 0.05916, we conclude that the models have a high agreement in the cascade path. *In addition to the high accuracy of BEM-PC in most of Monte-Carlo runs, on average it is approximately* 10 *times faster than TM.*

5.1.2. Polish system

As before, Figs. 11 and 12 compare how often the total demand loss and line outages at the end of cascade are above a particular level for TM and BEM-PC — a very close match is observed.

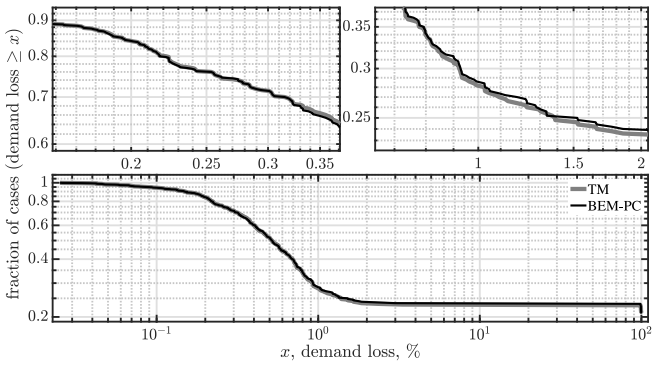


Fig. 11. Fraction of cases with % demand loss ≥ x at the end of cascade: Polish system.

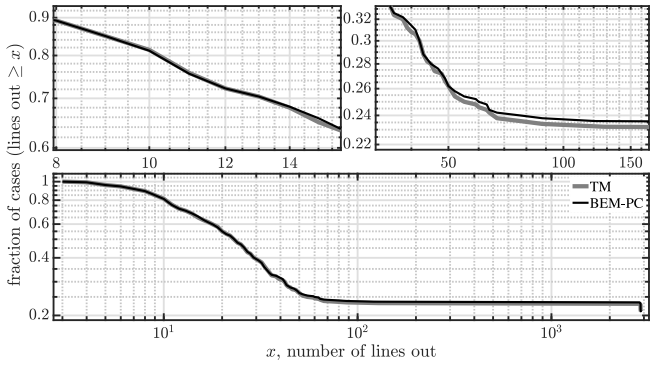


Fig. 12. Fraction of cases with line outage ≥ x at the end of cascade: Polish system.

Table 2
(a) End of cascade error, (b) path agreement measure, and (c) run time in TM w.r.t. BEM-PC: Polish System.

		Mean	Min	Max	Median
Error in	buses	0.1220	0	8	0
State of	machines	0.0620	0	3	0
	lines	0.1600	0	7	0
Maximum	$\|v\|$, pu	0.0008	0	0.0360	$2.1e{-5}$
Error in	$\angle v$, deg.	0.1441	0	10.1075	$4.0e{-4}$
	f, Hz	0.0165	0	0.2414	$8.4e{-5}$
R		0.9922	0.75	1	1
Runtime ratio		**34.6097**	**1.1593**	**430.3984**	**24.7959**

Table 2 compares various error measures at the end of cascade for BEM-PC with respect to TM. These indicate that for almost all of the cases, BEM-PC is able to accurately mimic the end results of cascade as in TM. The central tendency measures of R from Table 2 and its standard deviation equaling 0.03230 demonstrate that the models have a high degree of agreement in the cascade path. *Finally, runtime ratio indicates that on average the proposed model is 34.6 times faster than the standard model.*

Fig. 13 provides comparison between BEM-PC and TM on correlation of various end-of-cascade measures like demand loss vs number of outages and demand loss vs cascade sequence time (time between initial and final events). Out of 500 Monte-Carlo runs with 3 initial node outages, 41 cases are resilient cases and did not lead to any dependent events after initial node outages, whereas 118 cases did not converge and were considered as collapsed cases. In both panels in this figure, cases without dependent events and cases with complete collapse are disregarded. Following are the observations.

- Both plots indicate that the correlations among the two pairs of variables in BEM-PC closely match that of TM. However, there are a few cases in which these two approaches produce slightly different results.
- The density curves reveal almost identical distribution patterns for TM and BEM-PC for both sets of values on x and y axes.
- The right panel indicates a considerable number of cases have cascade sequence time more than 100 s.

For a sample case in the Polish system, time-domain plots representing the number of branch outages and demand loss against time are shown in Fig. 14. The nodes in the figure indicate the instants of outage/demand loss. The plots reveal that BEM-PC is following the exact cascade path as in the ground truth. Also, it is worth noting that the proposed simulation approach is 28 times faster than TM in this case. As described earlier, initially (at $t = 3$ s) 3 random nodes are disconnected to trigger cascade. These initial node outages are therefore not dependent outages, i.e., they do not represent the severity of cascade. In Fig. 14, the initial node outage caused disconnection of 10 lines at $t = 3$ s in addition to significant demand loss.

5.2. *Hyperstability – A challenge for BEM and performance of predictor–corrector approach*

In this section, accuracy of our proposed BEM-PC approach is tested against different cases with oscillatory instability in both IEEE 118-bus system and the Polish network. Since the 118-bus and the Polish system does not exhibit oscillatory instability under nominal setting, we create two oscillatory instability situations in the system by making damping coefficient negative in some machines, first at the beginning of cascade (case #1), and in a separate case somewhere in the middle of cascade (case #2).

The SPS action is designed to trip two unstable machines with the highest amplitudes of oscillations upon detection of oscillatory instability. In TM this takes place through *explicit* SPS action after 7.5 s and 4.5 s, respectively, in IEEE 118-bus and Polish system. However, in BEM-PC, a *functional* implementation is achieved by identifying the unstable mode and participating machines using eigendecomposition of the A matrix for post-event equilibrium as shown in Fig. 3. Then, the suitable predetermined protection action is taken by SPS. Note that other type of SPS actions can also be taken like tripping certain lines to disconnect areas oscillating against each other.

5.2.1. *IEEE 118-bus system*

We make the damping coefficients of generators $G39$ and $G51$ negative. For case #2, we introduce the negative values at the start of third tier in the middle of cascade. Fig. 15 shows rotor speeds of two representative machines in case #1 for TM, BEM-PC, and BEM without PC-approach (BEM). The top and bottom subplots show that BEM leads to only one tier of cascade due to hyperstability issue. The top panels show that BEM-PC has captured the oscillatory instability in the system and is able to tackle the hyperstability issue of BEM — note slight difference in tripping times in BEM-PC due to the OC relays' window-based averaging described in Section 4. The estimated unstable modes by BEM-PC are $0.1239 \pm j2.4889$ and $0.1121 \pm j2.0531$ and the corresponding estimated modeshapes are shown in Fig. 16. Based on the modeshapes, $G51$ and $G39$ are tripped after a 7.5 s delay. Eventually, TM and BEM-PC lead to 25 tiers of cascade (not shown here).

Tables 3 and 4 compare path agreement and various end-of-cascade measures for these three models in cases #1 and #2. Clearly, BEM-PC solved the hyperstability issue of BEM, had identical cascade propagation path, and replicated the end results of cascade in the ground truth in much shorter time. As expected, BEM without PC approach shows significantly different results than the ground truth.

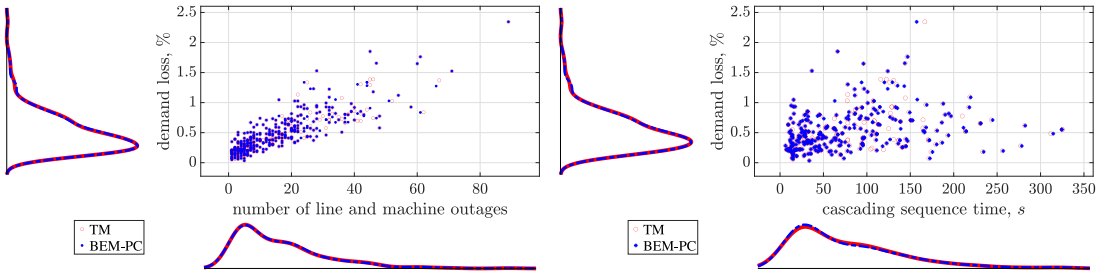


Fig. 13. Left: Demand loss vs number of branch and machine outages. Right: Demand loss vs cascade sequence time for Polish system. The density curves of the x and y variables are shown on the bottom and left of each subplot, respectively.

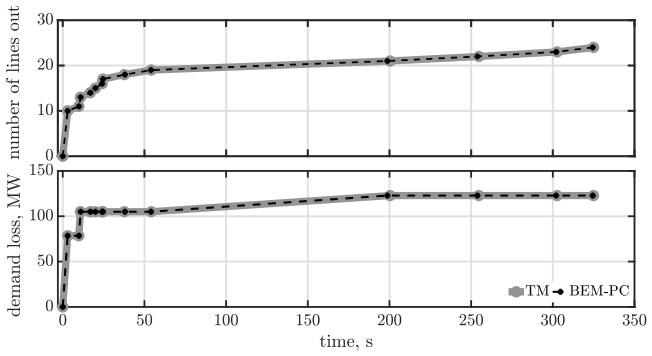


Fig. 14. Timeline of number of branch outages and demand loss during cascade: Polish system.

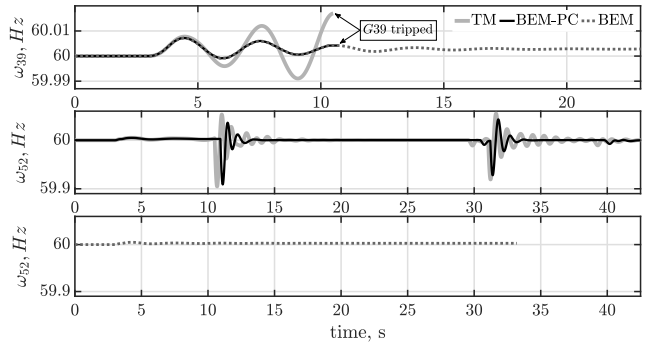


Fig. 15. IEEE 118-bus system case #1: Variation of speeds of $G39$ and $G52$ with oscillatory instability from beginning of cascade.

Table 3
End-of-cascade comparison: IEEE 118-bus system.

		Demand loss, %	Lines out	Mach. out	Cascade time, s	Runtime ratio
Case #1	TM	83.41	157	44	146.93	7.72
	BEM-PC	83.41	157	44	146.23	1
	BEM	1.06	4	0	3	0.09
Case #2	TM	7.01	31	6	337.11	5.29
	BEM-PC	7.01	31	6	336.78	1
	BEM	1.88	9	0	69.88	0.27

5.2.2. Polish system

Generators $G286$ and $G287$ are selected to create the oscillatory instability in the system. Fig. 17 shows rotor speed variation in two selected machines in the system for case #2. While BEM without PC is not able to capture the oscillatory instability and diverges from the ground truth, this figure along with Tables 5 and 6 reveal that in both cases, BEM-PC attain the identical end results of cascade as TM. As an example, for case #2, BEM-PC estimates the unstable modes

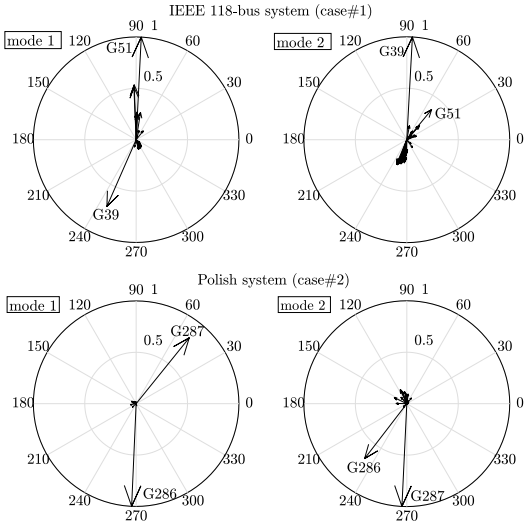


Fig. 16. Modeshapes of generator speeds estimated by BEM-PC corresponding to the unstable modes for case #1 in IEEE 118-bus and for case #2 in Polish system under the predictor subprocess (b) in Fig. 3.

$0.4306 \pm j9.7845$ and $0.4464 \pm j9.3910$, and corresponding modeshapes in Fig. 16, which leads to trippings of $G286$ and $G287$ after a 4.5 s delay by the SPS. Finally, Fig. 18 represents branch outages and demand loss against the cascade progression time for case #2. It reveals that cascade in BEM is stopping after 4 tiers around 43 s. Although, because of very close trip delays of two overloaded lines around $t = 45$ s, these lines are tripped together in one tier of cascade in BEM-PC, and in two tiers in TM, they follow identical cascade propagation paths (see, R in Table 6) and produce the same results at the end-point of cascade.

For further judging the efficacy of the proposed BEM-PC approach in handling the hyperstability issue, it would be ideal to study a system that naturally exhibits oscillatory instability as the cascade propagates. To that end, we have also considered the IEEE 68-bus New England-New York (NE-NY) benchmark test system [31] that is widely used for studying oscillatory instability problems.

5.2.3. IEEE 68-bus NE-NY system

The system has 5 areas, 87 lines, and 16 generators – a detailed description can be found in [31]. The grid exhibits multiple oscillatory modes among which the mode with eigenvalues $-0.0804 \pm 2.4474j$ has the least damping ratio. The modeshapes of generator speeds for this mode is shown in Fig. 26(a). We run 500 Monte Carlo (MC) simulations with two random initial line outages in AMD Ryzen 7 3800X CPU with 32 GB RAM. Unlike the IEEE 118-bus and Polish systems, we use $T_w^{OC} = 4$ s, since this system exhibits low-frequency interarea modes.

As shown in Table 7, BEM-PC encounters hyperstability issues in 16.4% of cases. The modeshapes of the generator speeds for the unstable mode with eigenvalue $0.0061 \pm 2.3740j$ is estimated by BEM-PC,

Table 4
End-of-cascade and path agreement comparison: IEEE 118-bus System.

Case	TM vs	Error in state of			Max error in			R
		buses	mach.	lines	$\|v\|, pu$	$\angle v, deg.$	f, Hz	
#1	BEM-PC	0	0	0	7.8e−7	2e−4	1.8e−4	1
	BEM	93	44	153	9.7e−2	13.58	4.09	0
#2	BEM-PC	0	0	0	5.8e−6	6.4e−3	5.3e−5	1
	BEM	6	6	22	9.7e−2	15.63	1.4e−4	0.19

Table 5
End-of-cascade comparison: Polish system.

		Demand loss, %	Lines out	Mach. out	Cascade time, s	Runtime ratio
Case #1	TM	0.96	34	4	90.84	25.74
	BEM-PC	0.96	34	4	90.74	1
	BEM	0.22	9	1	11.07	0.15
Case #2	TM	0.92	35	3	115.66	40.37
	BEM-PC	0.92	35	3	115.96	1
	BEM	0.18	10	0	43.69	0.16

Table 6
End-of-cascade and path agreement comparison: Polish system.

Case	TM vs	Error in state of			Max error in			R
		buses	mach.	lines	$\|v\|, pu$	$\angle v, deg.$	f, Hz	
#1	BEM-PC	0	0	0	3.2e−5	1.2e−3	7.3e−5	1
	BEM	15	3	25	5.7e−2	4.15	9.7e−4	0.11
#2	BEM-PC	0	0	0	3.3e−5	1.2e−3	7.6e−5	1
	BEM	15	3	25	5.7e−2	4.19	1.1e−3	0.14

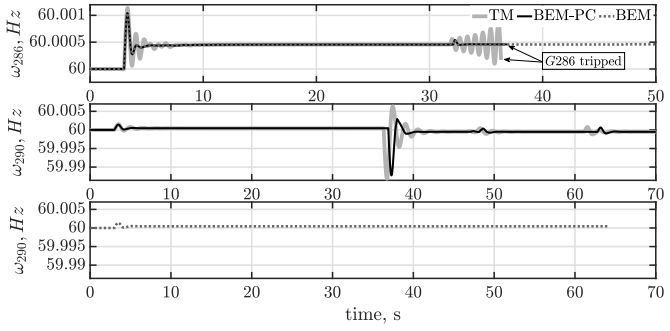


Fig. 17. Polish system case #2: Variation of speeds of $G286$ and $G290$ with oscillatory instability in the middle of cascade.

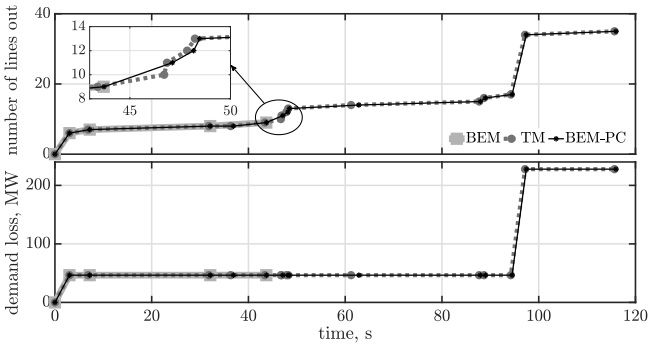


Fig. 18. Timeline of number of branch outages and demand loss during cascade in the oscillatory instability case #2: Polish system.

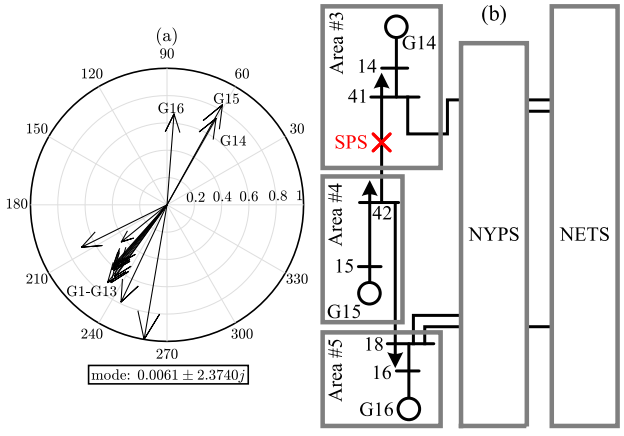


Fig. 19. Left: Modeshapes of generator speeds estimated by BEM-PC corresponding to the unstable interarea mode for a typical case with hyperstability issue in NE-NY system under the predictor subprocess (b) in Fig. 3. Right: One-line diagram of NE-NY test system highlighting SPS action.

and is shown in Fig. 19(a). Clearly, the most poorly-damped mode in the predisturbance condition has become unstable during cascade propagation and generators $G14$–$G16$ oscillate against the rest of the generators in this mode. Upon prediction of hyperstability, BEM-PC performs a corrective step by tripping line 41–42 after 5 s of the latest event using the predefined SPS action, see Fig. 19(b).

The fraction of cases where the demand loss and line outages at the end of cascade are above a threshold are compared in Fig. 20. We see a very close match between TM and BEM-PC. Table 8 shows breakdown of errors in states of buses, machine, and lines at the end of cascade of BEM-PC along with its path agreement measure R. It can be seen that the proposed approach is highly accurate in following the actual cascade path, while achieving $\approx 20\times$ speedup on an average, in spite of naturally occurring hyperstability problem.

5.3. Analysis of computational efficiency

In support of the logical arguments presented in Section 3.3, we present our analysis of computational efficiency of BEM-PC based on simulation data. To that end, we performed rigorous data collection

Table 7
Number of cases with/without hyperstability detection by BEM-PC during cascade: NE-NY system.

	# of cases with hyperstability	# of cases without hyperstability
Number	82	418
Percentage, %	16.4	83.6

relating individual subprocesses in Fig. 3. In addition to TM, we also present performance comparison with the partitioned approach widely used in production-grade softwares.

5.3.1. Performance comparison with TM

Statistical analysis of the overall CPU time comparison between BEM-PC and TM was presented using the runtime ratio in Tables 1–2. The box plots of this metric are also shown in Fig. 24. Although the runtime ratio is a good measure of the overall computational efficiency of BEM-PC, it is important to understand how the individual

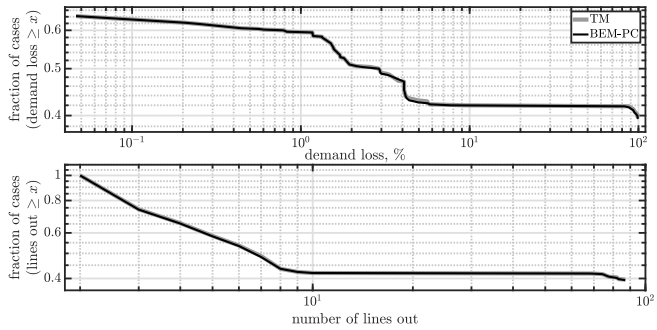


Fig. 20. Fraction of cases with % demand loss $\geq x$ and line outage $\geq x$ at the end of cascade in NE-NY system: comparison between TM and BEM-PC.

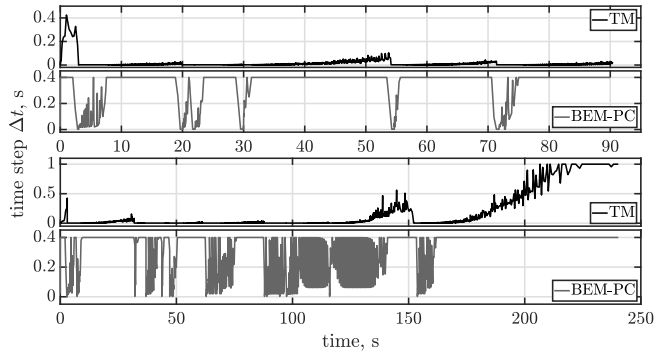


Fig. 21. Adaptation of integration time step Δt in TM and BEM-PC. Case (I): Top subplots. Case (II) Bottom subplots.

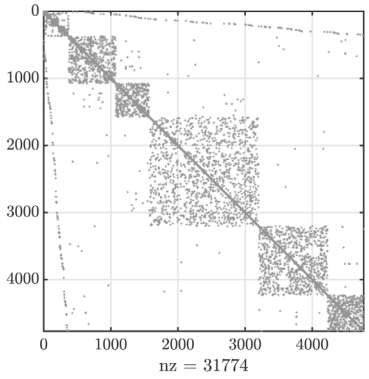


Fig. 22. Visualization of sparsity pattern of J_{22} in predisturbance condition with dimension of 4766 × 4766, where nz shows number of nonzero elements. The matrix is 99.86% sparse.

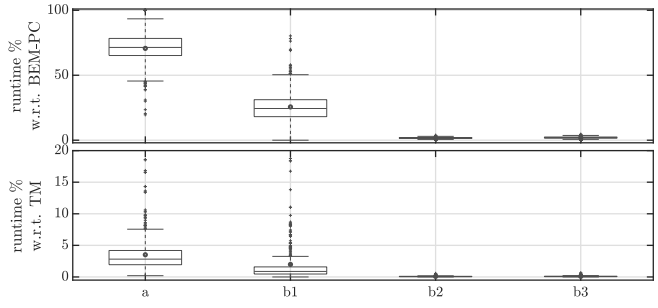


Fig. 23. Boxplots of runtimes of different subprocesses of BEM-PC as in Fig. 3 during 500 MC runs of Polish system. **a**: subprocess (a). **b1**: equilibrium calculation in subprocess (b). **b2**: A matrix calculation from (14) in subprocess (b), and **b3**: eigendecomposition of A matrix in subprocess (b). Runtimes are expressed as a % of total runtimes of BEM-PC (top) and TM (bottom).

Table 8
(a) End of cascade error, (b) path agreement measure, and (c) run time in TM w.r.t. BEM-PC: NE-NY system.

		Mean	Min	Max	Median
Error in State of	buses	0.132	0	18	0
	machines	0.032	0	5	0
	lines	0.138	0	19	0
R		0.997	0.428	1	1
Runtime ratio		**19.687**	**0.235**	**120.737**	**15.582**

subprocesses in Fig. 3 contribute towards that. We choose Polish system to demonstrate this due to the scalability challenge it poses. To this end, first we choose two cases — Case (I): a case without hyperstability, and Case (II): the hyperstability case #2 analyzed in the previous Section. The analysis of BEM-PC subprocesses are performed below.

1. Subprocess (a): Fig. 21 shows the adaptation of integration time step Δt in BEM-PC and TM while solving the cascading process leading to the main surviving island. It can be seen that TM demands much shorter time step whereas BEM-PC enjoys simulation with $\Delta t_{max} = 0.4$ s for most of the simulation period. It can be calculated from Table 9 that this subprocess consumes ≈ 70% and ≈ 75% of BEM-PC's overall CPU time for Cases (I) and (II), respectively. These are however, merely ≈ 2% of the TM's runtime in both cases.

2. Subprocess (b): Table 9 also shows a breakdown of BEM-PC's runtime within subprocess (b) in Fig. 3. As mentioned in Section 3.3, this subprocess can be segmented further into (b1), (b2), and (b3). Subprocess (b1) is the most expensive among the three, and based on the runtimes shown in Table 9, it uses ≈ 23% of BEM-PC's overall CPU time for both cases. Subprocess (b2) requires inversion of the submatrix J_{22}, which is very sparse. To get an idea, Fig. 22 shows the sparsity pattern of J_{22} in pre-disturbance condition. The dimension of the matrix is 4766 × 4766, and it is 99.86% sparse. It takes ≈ 2.16 s to form the corresponding largest A matrix in PSU's ROAR computers [30] when sparse objects are used in conjunction with Matlab's most comprehensive inversion routine [21]. The dimension of the A matrix in this case is 1964 × 1964, and it takes ≈ 2.5 s for its eigendecomposition in subprocess (b3). Based on the results in Table 9, (b2) and (b3) consume ≈ 1.5–2% and ≈ 1.5–2.5% of total CPU time of BEM-PC, respectively. Note that the subprocess (c) is lookup table-based and consumes negligible CPU time.

After an in-depth analysis of two specific cases, statistical analysis is performed to assess the computational burden of the subprocesses in 500 cases of Polish system. Fig. 23 shows the boxplots of these runtimes expressed as percentages of total runtimes of BEM-PC (top) and TM (bottom). The mean and median figures are specified in Table 10. The following conclusions can be drawn from these data –

1. Subprocess (a) consumes the most significant computational burden followed by (b1). In comparison, both calculation of A matrix and its eigendecomposition requires negligible CPU time.
2. Aided by the *stiff-decay* property, both of subprocesses (a) and (b1) run significantly faster than TM due to their ability to use larger integration step length Δt.

5.3.2. Performance comparison with partitioned approach

As described in the Introduction section, production grade stability programs use the partitioned approach for dynamic simulation. For a fair comparison, we have performed cascading failure simulations using the partitioned approach with 4th-order Runge–Kutta *(R–K)* method, which is a widely-used explicit numerical integration technique [10]. The simulations were run at a fixed time-step of 0.002 s. Both R–K and TM produces near-identical simulation results. Boxplots of normalized runtime of R–K w.r.t. BEM-PC are shown in Fig. 24. The following are the key observations–

Table 9
Runtime in seconds of different sub-processes in BEM-PC shown in Fig. 3. Case (I): generic case without hyperstability, and Case (II): with hyperstability in Polish system.

Case	BEM-PC									TM
	Round 1				Round 2				Total	Total
	a	b1	b2	b3	a	b1	b2	b3		
I	829.3	276.5	25.2	28.5	0	0	0	0	1159.5	37615.2
II	440.2	103.9	8.1	8.3	1608.0	528.2	32.2	33.2	2762.1	111521.0

a: subprocess (a) **b1**: equilibrium calculation in subprocess (b) **b2**: A matrix calculation attained from (14) in subprocess (b) **b3**: eigen decomposition in subprocess (b), see Fig. 3.

Table 10
Mean and median values of runtimes of different subprocesses of BEM-PC in Fig. 3 as a % of total runtime of BEM-PC and TM in 500 MC runs for Polish system.

		BEM subprocess			
		a	b1	b2	b3
Mean	BEM-PC	70.69	25.62	1.71	1.98
	TM	3.51	1.98	0.08	0.09
Median	BEM-PC	71.42	24.41	1.7	1.94
	TM	2.82	0.87	0.06	0.07

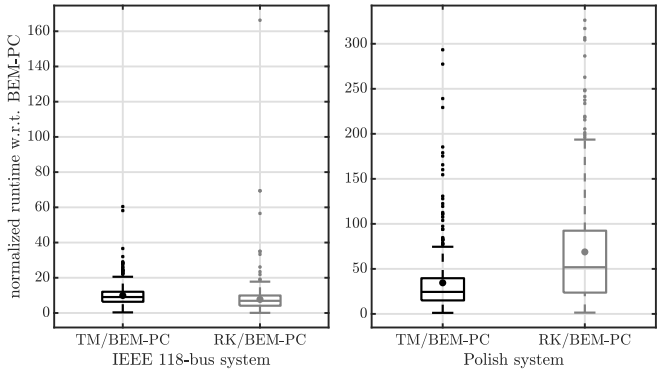


Fig. 24. Boxplots of normalized runtime of TM and R–K w.r.t. BEM-PC for cascading failure in — Left: IEEE 118-bus system, and Right: Polish system.

Table 11
Number of cases with nonzero errors in demand loss and line outage at the end of cascade comparing ground truth (TM) against classical and AC-QSS models: NE-NY and Polish systems.

TM vs	# of cases with error in			
	Demand loss		Lines out	
	NE-NY	Polish	NE-NY	Polish
TM classical	79	76	62	113
AC-QSS	297	296	322	342

1. For the relatively smaller IEEE 118-bus system, R–K is slightly faster than the TM. However, the mean and median ratios of runtime between R–K and BEM-PC from 500 MC runs are 7.72 and 6.91, respectively. For TM these numbers are 9.96 and 9.05, respectively.
2. For the Polish system, R–K is slower than TM. The mean and median ratios of runtime between R–K and BEM-PC from 393 MC runs are 68.82 and 51.84, respectively. For TM, these numbers corresponding to the same MC runs are 34.54 and 24.44, respectively. Note that the remaining 107 cases of the 500 MC runs could not be simulated using R–K method as those are running beyond 60 hours.

Clearly, BEM-PC retains its advantage over traditional partitioned approach. This is in line with what we mentioned in point #5 of remarks under Section 2.2.

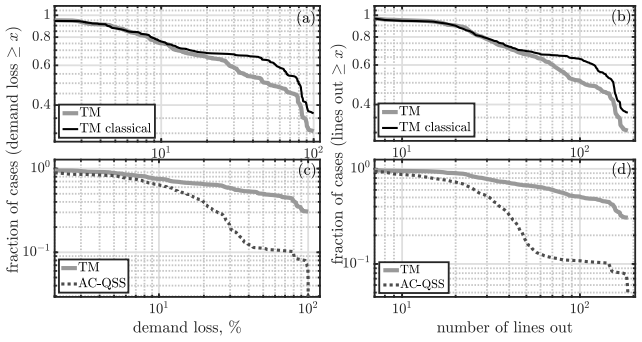


Fig. 25. Fraction of cases with % demand loss $\geq x$ and line outage $\geq x$ at the end of cascade in IEEE 118-bus system: (a), (b)- comparison between ground truth (TM) and classical models. (c), (d)- comparison between ground truth (TM) and AC-QSS models.

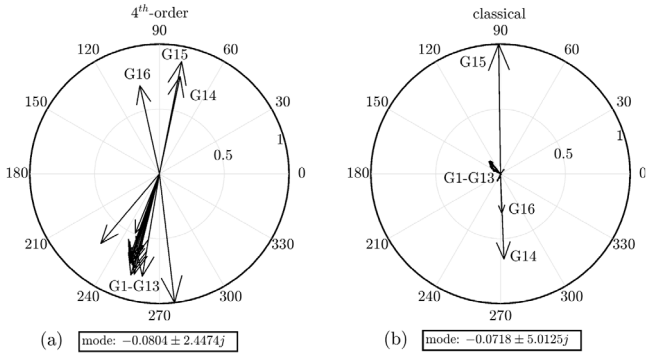


Fig. 26. Modeshapes of generator speeds corresponding to the most poorly-damped mode for 4th-order, and classical models of NE-NY system under the predisturbance condition.

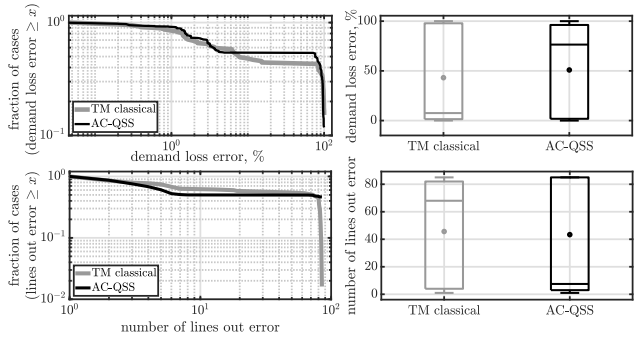


Fig. 27. Fraction of cases with % demand loss error $\geq x$ and line outage error $\geq x$ at the end of cascade in NE-NY system: comparison between ground truth against classical and AC-QSS models for cases with nonzero errors in demand loss and line outage.

5.4. Comparison with AC-QSS and classical models

In this section, we contrast the cascading failure simulation results in the ground truth (that uses a 4th-order generator model solved using TM) with the AC-QSS model [32] and dynamic model with classical generator representation. Going forward, *'error'* will imply difference w.r.t. ground truth denoted as 'TM.'

5.4.1. IEEE 118-bus system

Fig. 25 shows that both models demonstrate similarity with the 4th-order model, when cascading failure is less severe. However, as the cascade leads to further line outages and load tripping, the AC-QSS and the classical model start departing from the ground truth. For this system, the AC-QSS model shows an optimistic result, while the classical model presents a pessimistic result, when compared with the ground truth.

5.4.2. IEEE 68-bus NE-NY system

We first look into the modal characteristics of the system before initial outages. The modeshapes of generator speeds corresponding to the most poorly-damped modes are shown in Fig. 26 for 4th-order synchronous generator representation vs classical model representation. Next, Table 11 shows the number of cases with nonzero error with respect to ground truth in line outage and demand served at the end of cascade when classical generator-based model and AC-QSS model are used in NE-NY system. Finally, Fig. 27 quantifies the error in such cases. The following are the key observations from Figs. 26, 27, and Table 11 –

- The modal characteristics of the 4th-order and classical model-based systems are quite different. For the former, the most poorly-damped mode is $-0.0804 \pm 2.4474j$, whereas for the latter, it is $-0.0718 \pm 5.0125j$. In 4th-order model, generators $G14 - 16$ oscillate against those in NETS and NYPS for this mode, whereas for the classical model, $G15$ oscillates against $G14$ and $G16$ for the corresponding mode.
- Fig. 27 reveals that in the classical model among the cases in Table 11 the mean demand loss error is higher than 40% and the mean line outage error is more than 40. These errors are similar for the AC-QSS model, but for a much larger number of cases, as shown in Table 11.

5.4.3. Polish system

Table 11 shows the number of cases with nonzero error with respect to ground truth in line outage and demand served at the end of cascade when classical generator-based model and AC-QSS model are used in Polish system. Fig. 28 quantifies the error in such cases. In line with the expectations, the AC-QSS model shows higher error than the classical model.

6. Conclusion and future work

A fast time-domain cascading failure simulation approach based on implicit Backward Euler method (BEM) with stiff decay property is proposed in this work. To solve the hyperstability problem of BEM, we proposed a parallelizable predictor–corrector (BEM-PC) approach requiring eigendecomposition of the system matrix corresponding to the linear model obtained around the post-event unstable equilibrium, which BEM converges to. The system matrix is obtained as a by-product of BEM. The proposed BEM-PC approach is benchmarked in a serial implementation against the traditional Trapezoidal method (TM)-based approach. It has shown on an average $\approx 10\times$ speedup in IEEE 118-bus system, $\approx 20\times$ speedup in IEEE 68-bus system, and $\approx 35\times$ speedup in the Polish grid based on 500 simulations in each system with random node outages while following exact cascade paths and end results as in TM in most of the cases. It was also shown that BEM-PC retains its computational advantage with respect to partitioned approach using Runge–Kutta-based numerical integration method. Finally, it was shown that AC-Quasi-Steady-State and classical generator model-based representations can lead to different results when compared with a detailed model with 4th-order generator and exciter dynamics. Our ongoing and future work focuses on parallelization of BEM-PC, which should lead to further speedup.

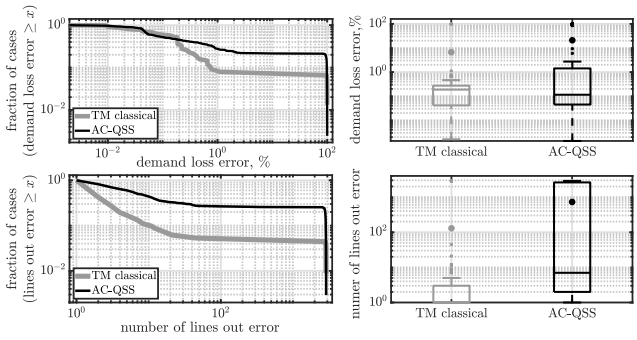


Fig. 28. Fraction of cases with % demand loss error $\geq x$ and line outage error $\geq x$ at the end of cascade in Polish system: comparison between ground truth against classical and AC-QSS models for cases with nonzero errors in demand loss and line outage.

CRediT authorship contribution statement

Sina Gharebaghi: Methodology, Software, Validation, Investigation, Data curation, Writing – original draft. **Nilanjan Ray Chaudhuri:** Conceptualization, Methodology, Validation, Investigation, Data curation, Writing – original draft, Supervision, Project administration, Funding acquisition. **Ting He:** Validation, Formal analysis, Writing – review & editing. **Thomas La Porta:** Validation, Formal analysis, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Nilanjan Ray Chaudhuri reports financial support was provided by National Science Foundation. Nilanjan Ray Chaudhuri is the lead inventor of a provisional US patent titled FAST CONTINGENCY SIMULATION IN DYNAMIC MODELS OF POWER SYSTEMS filed by The Pennsylvania State University.

Data availability

Data will be made available on request.

References

[1] Henneaux P, et al. Benchmarking quasi-steady state cascading outage analysis methodologies. In: 2018 IEEE int. conf. on prob. methods applied to power systems. 2018, p. 1–6.
[2] Pierre BJ, Arguello B, Garcia MJ. Optimal investments to improve grid resilience considering initial transient response and long-term restoration. In: 2020 Int. conf. on prob. methods applied to power systems. 2020, p. 1–6.
[3] Song J, et al. Dynamic modeling of cascading failure in power systems. IEEE Trans Power Syst 2016;31(3):2085–95.
[4] Vallem M, et al. Hybrid cascading outage analysis of extreme events with optimized corrective actions. In: Int. conf. on intelligent system app. to power systems. 2017, p. 1–6.
[5] Henneaux P, et al. A two-level probabilistic risk assessment of cascading outages. IEEE Trans Power Syst 2016;31(3):2393–403.
[6] Flueck AJ, et al. Dynamics and protection in cascading outages. In: 2020 IEEE Power Energy Society general meeting. 2020, p. 1–5.
[7] Parmer C, et al. Developing a dynamic model of cascading failure for high performance computing using trilinos. Assoc Comput Mach 2011;25–34.
[8] Khaitan SK, et al. Fast parallelized algorithms for on-line extended-term dynamic cascading analysis. In: 2009 IEEE/PES power systems conference and exposition. 2009, p. 1–7.

[9] Schafer B, Witthaut D, Timme M, Latora V. Dynamically induced cascading failures in power grids. Nature Commun 2018;9.
[10] Kundur P. Power system stability and control. NY, USA: McGraw-Hill; 1994.
[11] Sauer PW, Pai MA, Chow JH. Power system dynamics and stability: with synchrophasor measurement and power system toolbox. John Wiley & Sons; 2021.
[12] Concepcion R, et al. On extended-term dynamic simulations with high penetrations of photovoltaic generation. In: 2016 IEEE Power and Energy Society general meeting. 2016, p. 1–5.
[13] Ascher UM, Petzold LR. Computer methods for ordinary differential equations and differential-algebraic equations. Siam; 1998.
[14] Power system dynamic analysis, phase I. Final report EPRI EL-484, Electric Power Research Institute; Jul. 1977.
[15] Tinney W, Meyer W. Solution of large sparse systems by ordered triangular factorization. IEEE Trans Automat Control 1973;18(4):333–46.
[16] Davis T, Stanley K. Klu: a "clark kent" sparse lu factorization algorithm for circuit matrices. In: 2004 SIAM conference on parallel processing for scientific computing. 2004.
[17] Ortega J. Introduction to parallel and vector solution of linear systems. Frontiers in computer science, US: Springer; 1988.
[18] Dongarra JJ, et al. Solving linear systems on vector and shared memory computers. USA: Society for Industrial and Applied Mathematics; 1990.
[19] Griffiths D, Higham D. Numerical methods for ordinary differential equations: Initial value problems. Springer; 2010.
[20] MATLAB. Natick, Massachusetts: The MathWorks Inc.; 2022, 9.9.0.1524771, R2020b.
[21] mldivide. 2022, URL: https://mathworks.com/help/matlab/ref/mldivide.html.
[22] Zimmerman RD, et al. MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education. IEEE Trans Power Syst 2011;26(1):12–9.
[23] Fabozzi D, Van Cutsem T. Simplified time-domain simulation of detailed long-term dynamic models. In: 2009 IEEE Power Energy Society general meeting. 2009, p. 1–8.
[24] Uchida N, Nagao T. A new eigen-analysis method of steady-state stability studies for large power systems: S matrix method. IEEE Trans Power Syst 1988;3(2):706–14.
[25] Bezerra LH, Martins N. Eigenvalue methods for calculating dominant poles of a transfer function and their applications in small-signal stability. Appl Math Comput 2019;347:113–21.
[26] Rommes J, et al. Computing rightmost eigenvalues for small-signal stability assessment of large-scale power systems. IEEE Trans Power Syst 2010;25(2):929–38.
[27] Davis TA, et al. Algorithm 832: UMFPACK V4.3 – an unsymmetric-pattern multifrontal method. ACM Trans Math Softw 2004;30(2):196–9.
[28] spparms. 2022, URL: https://www.mathworks.com/help/matlab/ref/spparms.html.
[29] Golub GH, Van Loan CF. Matrix computations. 4th ed. JHU Press; 2013.
[30] Penn state, institute for computational and data sciences. 2022, URL: https://www.icds.psu.edu/computing-services/.
[31] Chaudhuri NR. Wide-area monitoring and control of future smart grids [Ph.D. thesis], London, U.K.: Imperial College; 2011.
[32] Gharebaghi S, Vennelaganti SG, Chaudhuri NR, He T, Porta TFL. Inclusion of pre-existing undervoltage load shedding schemes in AC-QSS cascading failure models. IEEE Trans Power Syst 2021;36(6):5645–56.