A Modular Port-Hamiltonian Framework for Power System Dynamic Simulation: Methodology and Implementation

Contents

1 Port-Hamiltonian System Framework

1.1 General Port-Hamiltonian Formulation

The Port-Hamiltonian Systems (PHS) framework provides an energy-based modelling paradigm that unifies storage, dissipation, and external interaction through the concept of power ports [1]. A finite-dimensional PHS is described by

$$\begin{bmatrix} \dot{\boldsymbol{x}} \\ \boldsymbol{w} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} \mathbf{J}_{xx} - \mathbf{R}_{xx} & \mathbf{J}_{xw} - \mathbf{R}_{xw} & \mathbf{G}_x \\ \mathbf{J}_{wx} - \mathbf{R}_{wx} & \mathbf{J}_{ww} - \mathbf{R}_{ww} & \mathbf{G}_w \\ -\mathbf{G}_x^\top & -\mathbf{G}_w^\top & \mathbf{0} \end{bmatrix} \begin{bmatrix} \nabla_{\boldsymbol{x}} \mathcal{H} \\ \boldsymbol{z}(\boldsymbol{w}) \\ \boldsymbol{u} \end{bmatrix}, \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the state vector, $\mathcal{H}(\boldsymbol{x})$ is the Hamiltonian (total stored energy), $\boldsymbol{u}$ and $\boldsymbol{y}$ are input and output port variables satisfying $\boldsymbol{y}^\top \boldsymbol{u} = P_{\text{ext}}$ (supplied power), $\boldsymbol{w}$ and $\boldsymbol{z}$ are dissipative flow–effort pairs, $\mathbf{J} = -\mathbf{J}^\top$ is the skew-symmetric interconnection matrix, and $\mathbf{R} = \mathbf{R}^\top \geq 0$ is the positive semi-definite dissipation matrix.

The passivity property

$$\frac{\mathrm{d}\mathcal{H}}{\mathrm{d}t} = -\boldsymbol{z}^\top \boldsymbol{w} + \boldsymbol{y}^\top \boldsymbol{u} \leq \boldsymbol{y}^\top \boldsymbol{u} \tag{2}$$

guarantees that the internal energy can only decrease when the external supply is zero, providing a natural Lyapunov function candidate for stability analysis.

1.2 Software Architecture

The proposed framework implements the PHS formulation through a modular, component-based software architecture consisting of four principal layers:

(i) **Core Layer** (`pyphs_core.py`): A standalone symbolic PHS engine managing state variables $\boldsymbol{x}$, dissipative variables $\boldsymbol{w}$, dissipation functions $\boldsymbol{z}$, input–output ports $(\boldsymbol{u}, \boldsymbol{y})$, and the Hamiltonian $\mathcal{H}$. The `DynamicsCore` class extends this with numerical dynamics capability, providing the interface $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\theta})$ where $\boldsymbol{\theta}$ denotes component parameters.

(ii) **Component Layer** (`components/`): Individual machine models (generators, exciters, governors, renewable energy converters) each implemented as a `DynamicsCore` with a dynamics function, output function, initialization function, and parameter metadata. Components are self-describing through attributes: `n_states`, `model_name`, `component_type`.

(iii) **System Assembly Layer** (`system_builder.py`, `component_factory.py`): Reads JSON configuration files specifying bus data, line parameters, generator models, and control equipment. A factory pattern dynamically instantiates components, enabling plug-and-play model substitution without code modification.

(iv) **Simulation Layer** (`system_coordinator.py`, `fault_sim_modular.py`, `power_flow.py`): Handles network construction (Ybus), Kron reduction, Park transformations, iterative network solution, AC power flow, equilibrium initialization, and time-domain integration.

The entire system is configured through JSON files with no hardcoded topology, enabling users to define arbitrary multi-machine systems by specifying bus, line, generator, exciter, and governor data declaratively.

2 Synchronous Generator Model

2.1 GENROU: Round-Rotor Synchronous Machine

The sixth-order round-rotor model (GENROU) represents the electromechanical dynamics of synchronous generators. The state vector is

$$\boldsymbol{x}_g = \begin{bmatrix} \delta & p & \psi_d & \psi_q & \psi_f & \psi_{kd} & \psi_{kq} \end{bmatrix}^\top, \tag{3}$$

where δ is the rotor angle (rad), $p = M\omega$ is the angular momentum (with ω in per-unit), ψ_d and ψ_q are the stator flux linkages in the d- and q-axes, ψ_f is the field winding flux, ψ_{kd} and ψ_{kq} are damper winding fluxes.

2.1.1 Mechanical Dynamics

The swing equation in Port-Hamiltonian form separates energy storage (rotor inertia) from dissipation (damping):

$$\dot{\delta} = \omega_b(\omega - 1) \tag{4}$$

$$\dot{p} = T_m - T_e - D(\omega - 1) \tag{5}$$

where $\omega_b = 2\pi f_0$ is the base angular frequency, T_m is the mechanical torque from the governor, $T_e = V_d I_d + V_q I_q$ is the electrical torque, D is the damping coefficient, and the per-unit speed is recovered as $\omega = p/M$ with the inertia constant $M = 2H \cdot S_n/S_{\text{sys}}$.

The Hamiltonian for the mechanical subsystem is

$$\mathcal{H}_{\text{mech}} = \frac{p^2}{2M}, \tag{6}$$

whose gradient $\partial\mathcal{H}_{\text{mech}}/\partial p = \omega$ appears naturally in the dynamics.

2.1.2 Electromagnetic Dynamics

The stator flux dynamics, expressed in the code per-unit convention $\psi_{\text{code}} = \psi_{\text{actual}}/\omega_b$, are

$$\dot{\psi}_d = V_d + r_a I_d + \omega_b \omega \psi_q \tag{7}$$

$$\dot{\psi}_q = V_q + r_a I_q - \omega_b \omega \psi_d \tag{8}$$

where r_a is the armature resistance and the generator convention (current positive leaving the machine) is adopted.

The rotor winding dynamics are governed by the field and damper time constants:

$$\dot{\psi}_f = \frac{E_{fd} K_{fd} - \psi_f}{T'_{d0}} \tag{9}$$

$$\dot{\psi}_{kd} = \frac{-\psi_{kd}}{T''_{d0}} \tag{10}$$

$$\dot{\psi}_{kq} = \frac{-\psi_{kq}}{T''_{q0}} \tag{11}$$

where E_{fd} is the field voltage from the exciter, $K_{fd} = L_f/X_{ad}$ is the scaling factor relating exciter per-unit to machine flux per-unit, and T'_{d0}, T''_{d0}, T''_{q0} are the open-circuit time constants.

The scaling factor K_{fd} is pre-computed from machine reactances:

$$K_{fd} = \frac{L_f}{X_{ad}}, \quad X_{ad} = x_d - x_l, \quad X_{fl} = \frac{X_{ad}(x'_d - x_l)}{X_{ad} - (x'_d - x_l)}, \quad L_f = X_{ad} + X_{fl}. \tag{12}$$

2.1.3 Subtransient Voltage and Network Interface

The generator interfaces with the network through the voltage-behind-reactance model. The subtransient internal voltages are computed from flux linkages:

$$E''_q = \frac{x''_d - x_l}{x'_d - x_l}\psi_f \tag{13}$$

$$E''_d = -\frac{x''_q - x_l}{x'_q - x_l}\psi_{kq} \tag{14}$$

The complex internal voltage in the machine frame is $\underline{E}'' = E''_d + jE''_q$, which is transformed to the system (RI) frame via the Park transformation:

$$\underline{E}''_{\text{sys}} = \left(E''_d \sin\delta + E''_q \cos\delta\right) + j\left(-E''_d \cos\delta + E''_q \sin\delta\right). \tag{15}$$

The generator is represented in the network as an admittance $y_g = 1/(jx''_d)$ behind the subtransient voltage. The output current is $\underline{I} = y_g(\underline{E}''_{\text{sys}} - \underline{V}_t)$, which is then transformed back to the machine frame for the dynamics computation.

2.1.4 Per-Unit System

All quantities are expressed on a common system base S_{sys} (typically 100 MVA). Machine parameters originally on machine base S_n are converted using:

$$Z_{\text{sys}} = Z_{\text{mach}} \cdot \frac{S_{\text{sys}}}{S_n}, \quad M_{\text{sys}} = 2H \cdot \frac{S_n}{S_{\text{sys}}}. \tag{16}$$

3 Excitation System Models

The framework implements four IEEE-standard excitation system models, each represented as a `DynamicsCore` with its own dynamics function, output function, and initialization procedure.

3.1 EXDC2: DC Exciter Type 2

The IEEE EXDC2 is a DC excitation system with voltage regulator, exciter with saturation feedback, and rate feedback. The state vector is

$$\boldsymbol{x}_{\text{exc}} = \begin{bmatrix} v_m & v_r & e_{fd} & x_f \end{bmatrix}^\top, \tag{17}$$

where v_m is the measured terminal voltage, v_r is the voltage regulator output, e_{fd} is the field voltage (exciter output), and x_f is the rate feedback state.

The dynamics are:

$$\dot{v}_m = \frac{V_t - v_m}{T_R} \tag{18}$$

$$V_f = \frac{K_F(e_{fd} - x_f)}{T_{F1}}, \quad V_{\text{err}} = V_{\text{ref}} - v_m - V_f \tag{19}$$

$$\dot{v}_r = \frac{K_A V_{\text{err}} - v_r}{T_A} \quad \text{with anti-windup: } v_r \in [V_{R,\min}, V_{R,\max}] \tag{20}$$

$$\dot{e}_{fd} = \frac{v_r - (K_E + S_E(e_{fd}))e_{fd}}{T_E} \tag{21}$$

$$\dot{x}_f = \frac{e_{fd} - x_f}{T_{F1}} \tag{22}$$

Saturation Function. The quadratic saturation model is used:

$$S_E(E) = \begin{cases} 0 & \text{if } |E| \le A \text{ or } B = 0 \\ \dfrac{B(|E| - A)^2}{|E|} & \text{otherwise} \end{cases} \tag{23}$$

where the coefficients A and B are pre-computed from the IEEE standard data points (E_1, S_{E1}) and (E_2, S_{E2}):

$$A = \frac{E_1\sqrt{S_{E2}E_2} - E_2\sqrt{S_{E1}E_1}}{\sqrt{S_{E2}E_2} - \sqrt{S_{E1}E_1}}, \quad B = \left(\frac{\sqrt{S_{E1}E_1}}{E_1 - A}\right)^2. \tag{24}$$

Anti-Windup Logic. The regulator output v_r is clamped to $[V_{R,\min}, V_{R,\max}]$. When at a limit and the derivative would push further into saturation, $\dot{v}_r$ is set to zero, preventing integrator windup.

Output. The field voltage output is simply $E_{fd} = e_{fd}$ (state index 2).

3.2 EXST1: Static Exciter Type 1

The IEEE EXST1 is a static excitation system with lead-lag compensation, input limiting, and washout rate feedback. The state vector is

$$\boldsymbol{x}_{\text{exc}} = \begin{bmatrix} v_m & v_{ll} & v_r & v_f \end{bmatrix}^\top. \tag{25}$$

The signal path is:

1. **Voltage measurement**: $\dot{v}_m = (V_t - v_m)/T_R$
2. **Washout feedback**: $V_{f,\text{out}} = K_F(v_r - v_f)/T_F, \quad \dot{v}_f = (v_r - v_f)/T_F$
3. **Voltage error**: $V_{\text{err}} = V_{\text{ref}} - v_m - V_{f,\text{out}}$
4. **Input limiter**: $v_i = \text{clip}(V_{\text{err}}, V_{I,\min}, V_{I,\max})$
5. **Lead-lag compensator**: $\dot{v}_{ll} = (v_i - v_{ll})/T_C, \quad v_{ll,\text{out}} = v_{ll} + (T_B/T_C)(v_i - v_{ll})$
6. **Voltage regulator**: $\dot{v}_r = (K_A v_{ll,\text{out}} - v_r)/T_A$ with anti-windup

The output E_{fd} includes dynamic field current compensation:

$$V_{R,\max}^{\text{dyn}} = V_{R,\max} - K_C \cdot X_{ad} I_{fd}, \quad E_{fd} = \text{clip}(v_r, V_{R,\min}^{\text{dyn}}, V_{R,\max}^{\text{dyn}}). \tag{26}$$

3.3 ESST3A: Static Exciter Type ST3A

The ESST3A is the most complex excitation model implemented, featuring inner and outer voltage regulators, VE compensation with rectifier regulation, and multiple limiters. The state vector is

$$\boldsymbol{x}_{\text{exc}} = \begin{bmatrix} v_{\text{LG}} & x_{\text{LL}} & V_R & V_M & V_B \end{bmatrix}^\top. \tag{27}$$

3.3.1 VE Compensation

The VE (voltage error) compensation block computes a compensated voltage magnitude using complex phasor arithmetic:

$$V_E = \left| K_P e^{j\theta_P}(V_d + jV_q) + j(K_I + K_P e^{j\theta_P} X_L)(I_d + jI_q) \right| \tag{28}$$

In the implementation, this is computed using explicit real and imaginary arithmetic to maintain compatibility with Numba JIT compilation:

$$K_{PC,r} = K_P \cos(\theta_P), \quad K_{PC,i} = K_P \sin(\theta_P) \tag{29}$$

$$z_1 = K_{PC,r} V_d - K_{PC,i} V_q + j(K_{PC,r} V_q + K_{PC,i} V_d) \tag{30}$$

$$z_2 = [-(K_{PC,i} X_L) + j(K_I + K_{PC,r} X_L)](I_d + jI_q) \tag{31}$$

$$V_E = |z_1 + z_2| \tag{32}$$

3.3.2 Rectifier Regulation

The rectifier regulation function F_{EX} is a piecewise function of $I_N = K_C X_{ad} I_{fd} / V_E$:

$$F_{EX}(I_N) = \begin{cases} 1 - 0.577 I_N & 0 \le I_N \le 0.433 \\ \sqrt{0.75 - I_N^2} & 0.433 < I_N \le 0.75 \\ 1.732(1 - I_N) & 0.75 < I_N \le 1.0 \\ 0 & I_N > 1.0 \end{cases} \tag{33}$$

The bridge voltage is $V_B = \text{clip}(V_E \cdot F_{EX}, 0, V_{B,\max})$, and the field voltage output is

$$E_{fd} = \text{clip}(V_B \cdot V_M, E_{fd,\min}, E_{fd,\max}). \tag{34}$$

3.3.3 Control Blocks

The outer regulator (voltage regulator) uses a lead-lag compensator with input limiting:

$$v_i = \text{clip}(V_{\text{ref}} - v_{\text{LG}}, V_{I,\min}, V_{I,\max}) \tag{35}$$

followed by a lead-lag block with transfer function $(1 + sT_B)/(1 + sT_C)$, and a gain/lag $K_A/(1 + sT_A)$ with output limits $[V_{R,\min}, V_{R,\max}]$.

The inner regulator (feedback loop) computes $V_M = K_M(V_R - K_G E_{fd})/(1 + sT_M)$ with limits $[V_{M,\min}, V_{M,\max}]$.

3.4 IEEEX1: IEEE Type 1 Exciter

The IEEEX1 model is similar to EXDC2 but includes a lead-lag compensator and voltage-dependent limits. The state vector is

$$\boldsymbol{x}_{\text{exc}} = \begin{bmatrix} v_m & v_{ll} & v_r & v_p & v_f \end{bmatrix}^\top. \tag{36}$$

Key differences from EXDC2:

- Lead-lag block $(1 + sT_C)/(1 + sT_B)$ between error signal and regulator
- Voltage-dependent regulator limits: $v_{r,\max} = V_{R,\max} \cdot V_t$, $v_{r,\min} = V_{R,\min} \cdot V_t$
- Speed compensation on output: $E_{fd} = v_p \cdot \omega$
- Exciter block with saturation: $\dot{v}_p = (v_r - K_E v_p - S_E(v_p)v_p)/T_E$

4 Governor–Turbine Models

4.1 TGOV1: Steam Turbine Governor

The TGOV1 model represents a simplified steam turbine governor with two states:

$$\boldsymbol{x}_{\text{gov}} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^\top, \tag{37}$$

where x_1 is the governor valve position and x_2 is the turbine power lag state.

The dynamics include droop-based speed control with deadband and valve position limits:

$$\omega_{\text{err}} = \omega_{\text{ref}} - \omega, \quad \text{with deadband: } |\omega_{\text{err}}| < \epsilon \implies \omega_{\text{err}} = 0 \tag{38}$$

$$P_{\text{gate}} = P_{\text{ref}} + \frac{\omega_{\text{err}}}{R}, \quad P_{\text{lim}} = \text{clip}(P_{\text{gate}}, V_{\min}, V_{\max}) \tag{39}$$

$$\dot{x}_1 = \frac{P_{\text{lim}} - x_1}{T_1} \tag{40}$$

$$\dot{x}_2 = \frac{x_1 - x_2}{T_3} \tag{41}$$

The mechanical torque output combines lead-lag dynamics with turbine damping:

$$T_m = \frac{T_2}{T_3}(x_1 - x_2) + x_2 - D_t(\omega - 1) \tag{42}$$

4.2 IEEEG1: Multi-Stage Steam Turbine Governor

The IEEEG1 model represents a multi-stage steam turbine with four turbine/reheater stages. The state vector is

$$\boldsymbol{x}_{\text{gov}} = \begin{bmatrix} x_{ll} & v_{\text{pos}} & x_4 & x_5 & x_6 & x_7 \end{bmatrix}^\top, \tag{43}$$

where x_{ll} is the lead-lag filter state, v_{pos} is the valve position, and x_4–x_7 represent the four turbine stages.

4.2.1 Speed Sensing and Valve Control

The lead-lag filter processes the speed deviation:

$$\dot{x}_{ll} = \frac{(\omega_{\text{ref}} - \omega) - x_{ll}}{T_1}, \quad \text{LL}_y = K\left[(\omega_{\text{ref}} - \omega) + \frac{(T_2 - T_1)x_{ll}}{T_1}\right] \tag{44}$$

The valve position dynamics include rate limits and position limits:

$$v_s = \frac{\text{LL}_y + P_{\text{aux}} + P_{\text{ref}} - v_{\text{pos}}}{T_3}, \quad \dot{v}_{\text{pos}} = \text{clip}(v_s, U_C, U_O) \tag{45}$$

subject to $v_{\text{pos}} \in [P_{\min}, P_{\max}]$ with anti-windup.

4.2.2 Turbine Stages

Four cascaded lag blocks represent the turbine and reheater stages:

$$\dot{x}_k = \frac{x_{k-1} - x_k}{T_{k+2}}, \quad k = 4, 5, 6, 7 \tag{46}$$

where $x_3 \equiv v_{\text{pos}}$ (valve position feeds the first stage).

4.2.3 Power Output

The total mechanical power is distributed across HP (high-pressure) and LP (low-pressure) outputs via normalized power fractions K_{1n}–K_{8n}:

$$P_m = \sum_{i \in \{4,5,6,7\}} (K_{2i-7,n} + K_{2i-6,n}) x_i \tag{47}$$

where the fractions are normalized to sum to unity: $\sum_{i=1}^{8} K_{i,n} = 1$.

5 Renewable Energy: Type-3 Wind Turbine Model

The framework includes a comprehensive Type-3 Wind Turbine Generator (WTG) model consisting of seven interconnected sub-components, each implemented as a `DynamicsCore`. The total state vector per turbine unit comprises 21 states.

5.1 REGCA1: Renewable Generator/Converter

The REGCA1 model represents the power electronic converter as a controlled current source. The state vector is

$$\boldsymbol{x}_{\text{reg}} = \begin{bmatrix} I_p & I_q & V_{\text{filt}} \end{bmatrix}^\top, \tag{48}$$

where I_p and I_q are the active and reactive current outputs, and V_{filt} is a filtered terminal voltage.

The dynamics implement first-order current command tracking with a low-voltage gain (LVG) function:

$$\dot{I}_p = \frac{I_{p,\text{cmd}} \cdot \text{LVG}(V) - I_p}{T_g} \tag{49}$$

$$\dot{I}_q = \frac{I_{q,\text{cmd}} - I_q}{T_g} \tag{50}$$

$$\dot{V}_{\text{filt}} = \frac{V_t - V_{\text{filt}}}{T_{\text{fltr}}} \tag{51}$$

The LVG function reduces active current during low-voltage conditions:

$$\mathrm{LVG}(V) = \begin{cases} 0 & V < V_{\mathrm{Lvpnt0}} \\ (V - V_{\mathrm{Lvpnt0}})/(V_{\mathrm{Lvpnt1}} - V_{\mathrm{Lvpnt0}}) & V_{\mathrm{Lvpnt0}} \le V < V_{\mathrm{Lvpnt1}} \\ 1 & V \ge V_{\mathrm{Lvpnt1}} \end{cases} \tag{52}$$

5.2 REECA1: Electrical Control

The REECA1 model generates current commands ($I_{p,\mathrm{cmd}}$, $I_{q,\mathrm{cmd}}$) based on plant-level power and voltage references. The state vector is

$$\boldsymbol{x}_{\mathrm{reec}} = \begin{bmatrix} V_f & P_f & x_{piq} & P_{\mathrm{ord}} \end{bmatrix}^\top. \tag{53}$$

Key control modes:

- **QFLAG=0**: Constant reactive current mode ($I_{q,\mathrm{cmd}} = I_{q,\mathrm{cmd0}}$)
- **QFLAG=1**: Voltage regulation mode with PI controller

Active current command is derived from power order divided by terminal voltage:

$$I_{p,\mathrm{cmd}} = \mathrm{clip}\left(\frac{P_{\mathrm{ord}}}{V_t}, I_{p,\mathrm{min}}, I_{p,\mathrm{max}}\right) \tag{54}$$

5.3 REPCA1: Plant Controller

The REPCA1 model provides plant-level active and reactive power regulation. The state vector is

$$\boldsymbol{x}_{\mathrm{repc}} = \begin{bmatrix} V_f & Q_f & x_{s2} & P_f & x_{s5} \end{bmatrix}^\top, \tag{55}$$

comprising voltage and power measurement filters, and two PI controller integrator states for reactive (x_{s2}) and active (x_{s5}) power regulation.

5.4 WTDTA1: Drive Train

The two-mass drive train model captures the torsional dynamics between turbine and generator rotors:

$$\boldsymbol{x}_{\mathrm{dt}} = \begin{bmatrix} \theta_{tw} & p_t & p_g \end{bmatrix}^\top, \tag{56}$$

where θ_{tw} is the shaft twist angle, $p_t = J_t\omega_t$ is the turbine momentum, and $p_g = J_g\omega_g$ is the generator momentum.

The dynamics follow from Newton's second law for rotating masses:

$$\dot{\theta}_{tw} = \omega_b(\omega_t - \omega_g) \tag{57}$$

$$\dot{p}_t = T_{\mathrm{aero}} - K_{\mathrm{sh}}\theta_{tw} - D_{\mathrm{sh}}(\omega_t - \omega_g) \tag{58}$$

$$\dot{p}_g = K_{\mathrm{sh}}\theta_{tw} + D_{\mathrm{sh}}(\omega_t - \omega_g) - T_e \tag{59}$$

where $\omega_t = p_t/J_t$, $\omega_g = p_g/J_g$, K_{sh} is the shaft stiffness, D_{sh} is the shaft damping, T_{aero} is the aerodynamic torque, and T_e is the electrical torque.

5.5 WTTQA1, WTPTA1, WTARA1: Torque, Pitch, and Aerodynamics

- **WTTQA1** (3 states): Torque controller with power-speed lookup table, PI speed controller, and power reference filtering. Computes the electrical power reference P_{ref}.
- **WTPTA1** (3 states): Pitch controller with two PI loops (speed-based and power-based) and pitch actuator dynamics. Outputs blade pitch angle θ.
- **WTARA1** (algebraic): Aerodynamics model computing aerodynamic power $P_{\text{aero}} = f(\omega_t, \theta)$ from lookup tables.

6 Network Solution

6.1 Admittance Matrix Construction

The network is represented by the bus admittance matrix $\mathbf{Y}_{\text{bus}}$, constructed from Line data in the JSON configuration. Transmission lines use the standard π-model:

$$y_s = \frac{1}{r + jx}, \quad y_{\text{sh}} = j\frac{b}{2}, \tag{60}$$

with the Ybus entries:

$$Y_{ii} \mathrel{+}= y_s + y_{\text{sh}}, \quad Y_{jj} \mathrel{+}= y_s + y_{\text{sh}}, \quad Y_{ij} \mathrel{-}= y_s. \tag{61}$$

Transformers with off-nominal tap ratio t are modelled as:

$$Y_{ii} \mathrel{+}= \frac{y_s}{t^2} + y_{\text{sh}}, \quad Y_{jj} \mathrel{+}= y_s + y_{\text{sh}}, \quad Y_{ij} = Y_{ji} \mathrel{-}= \frac{y_s}{t}. \tag{62}$$

6.2 Kron Reduction

For computational efficiency, non-generator (load) buses are eliminated through Kron reduction. Partitioning the Ybus into active (a: generator + grid buses) and load (l) subsets:

$$\begin{bmatrix} \mathbf{Y}_{aa} & \mathbf{Y}_{al} \\ \mathbf{Y}_{la} & \mathbf{Y}_{ll} \end{bmatrix} \begin{bmatrix} \boldsymbol{V}_a \\ \boldsymbol{V}_l \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_a \\ \boldsymbol{I}_l \end{bmatrix}, \tag{63}$$

the reduced admittance matrix retaining only active buses is:

$$\mathbf{Y}_{\text{red}} = \mathbf{Y}_{aa} - \mathbf{Y}_{al}\mathbf{Y}_{ll}^{-1}\mathbf{Y}_{la}. \tag{64}$$

Load admittances are incorporated into $\mathbf{Y}_{ll}$ as constant impedance loads at nominal voltage: $Y_{\text{load}} = (P - jQ)/|V|^2$ with $|V| = 1.0\,\text{pu}$.

6.3 Iterative Network Solution

The network is solved using an iterative current-balance approach that handles constant power loads. The augmented admittance matrix incorporates generator internal admittances:

$$\mathbf{Y}_{\text{aug}} = \mathbf{Y}_{\text{bus}} + \text{diag}(y_{g,1}, \ldots, y_{g,N})_{\text{gen buses}} \tag{65}$$

where $y_{g,i} = 1/(jx''_{d,i})$ is the i-th generator's subtransient admittance.

At each iteration k:

1. Compute generator current injection: $I_{\text{gen},i} = y_{g,i} \cdot \underline{E}''_{\text{sys},i}$
2. Compute constant power load current: $I_{\text{load},j} = (S_j / V_j^{(k)})^*$
3. Compute renewable current injection: $I_{\text{ren}} = (I_p + jI_q) \cdot V/|V|$
4. Total injection: $\boldsymbol{I}_{\text{total}} = \boldsymbol{I}_{\text{gen}} + \boldsymbol{I}_{\text{ren}} - \boldsymbol{I}_{\text{load}}$
5. Solve: $\boldsymbol{V}^{(k+1)} = \mathbf{Y}_{\text{aug}}^{-1} \boldsymbol{I}_{\text{total}}$
6. Check convergence: $\max_i |V_i^{(k+1)} - V_i^{(k)}| < 10^{-6}$

This process typically converges within 3–5 iterations.

6.4 Fault Application

Faults are modelled by adding a shunt admittance at the faulted bus:

$$Y_{\text{bus},ii}^{\text{fault}} = Y_{\text{bus},ii} + \frac{1}{Z_f} \tag{66}$$

where Z_f is the fault impedance. A solid three-phase fault corresponds to $Z_f \to 0$ (practically $Z_f = 0.01j$ pu). The fault is applied by switching the Ybus during the fault period and restoring it upon clearance.

7 Power Flow Analysis

7.1 Newton-Raphson AC Power Flow

The framework includes a Newton-Raphson AC power flow solver for computing the steady-state operating point. The power mismatch equations are:

$$\Delta P_i = P_{i,\text{spec}} - V_i \sum_{j=1}^{n} V_j \left[G_{ij} \cos(\theta_i - \theta_j) + B_{ij} \sin(\theta_i - \theta_j) \right] \tag{67}$$

$$\Delta Q_i = Q_{i,\text{spec}} - V_i \sum_{j=1}^{n} V_j \left[G_{ij} \sin(\theta_i - \theta_j) - B_{ij} \cos(\theta_i - \theta_j) \right] \tag{68}$$

The Jacobian matrix is structured as:

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_1 & \mathbf{J}_2 \\ \mathbf{J}_3 & \mathbf{J}_4 \end{bmatrix} = \begin{bmatrix} \frac{\partial \boldsymbol{\Delta P}}{\partial \boldsymbol{\theta}} & \frac{\partial \boldsymbol{\Delta P}}{\partial \boldsymbol{V}} \\ \frac{\partial \boldsymbol{\Delta Q}}{\partial \boldsymbol{\theta}} & \frac{\partial \boldsymbol{\Delta Q}}{\partial \boldsymbol{V}} \end{bmatrix} \tag{69}$$

The Newton update step with adaptive damping factor α is:

$$\begin{bmatrix} \Delta \boldsymbol{\theta} \\ \Delta \boldsymbol{V}/\boldsymbol{V} \end{bmatrix}^{(k+1)} = -\alpha \mathbf{J}^{-1} \begin{bmatrix} \boldsymbol{\Delta P} \\ \boldsymbol{\Delta Q} \end{bmatrix}^{(k)} \tag{70}$$

7.2 Automatic Slack Bus Selection

The power flow implements a three-tier automatic slack bus selection:

1. **User-specified**: From the `Slack` section in JSON (highest priority)
2. **Grid/voltage source**: Infinite bus models (for SMIB systems)
3. **Largest generator**: Generator with highest MVA rating (last resort)

7.3 Equilibrium Initialization

After power flow convergence, component equilibrium states are computed in a four-part process:

Part 1: Power Flow. Newton-Raphson solver provides voltage magnitudes and angles at all buses.

Part 2: Generator Internal States. For each generator:

$$\underline{V} = V_{\text{mag}} e^{j\theta_V}, \quad \underline{I} = \left(\frac{P + jQ}{\underline{V}} \right)^* \tag{71}$$

$$\underline{E} = \underline{V} + (r_a + jx''_d)\underline{I}, \quad \delta_0 = \angle \underline{E} \tag{72}$$

$$\psi_{f,0} = |\underline{E}|/g_{d1}, \quad p_0 = M \cdot 1.0, \quad P_{m,0} = P_e \tag{73}$$

Part 3: Exciter and Governor Initialization. Each component's `init_fn` computes equilibrium states by back-calculating through the signal chain. For example, EXDC2: $v_m = V_t$, $v_r = (K_E + S_E(E_{fd})) \cdot E_{fd}$, $e_{fd} = E_{fd}$, then $V_{\text{ref}} = v_m + K_F E_{fd}/T_{F1} + v_r/K_A$.

Part 4: Network Consistency Refinement. Stator fluxes ψ_d, ψ_q are recomputed from the Park transformation, and exciter voltage sensing states are updated with refined terminal voltages.

The result is machine-precision equilibrium with $\max |\dot{\boldsymbol{x}}| < 10^{-12}$.

8 Time-Domain Simulation

8.1 ODE Formulation

The complete system dynamics are assembled into a single ODE right-hand side function $\dot{\boldsymbol{x}} = f(t, \boldsymbol{x})$. The flat state vector concatenates all component states:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_{g,1}^\top & \boldsymbol{x}_{e,1}^\top & \boldsymbol{x}_{v,1}^\top & \cdots & \boldsymbol{x}_{g,N}^\top & \boldsymbol{x}_{e,N}^\top & \boldsymbol{x}_{v,N}^\top & \boldsymbol{x}_{\text{ren}}^\top \end{bmatrix}^\top \tag{74}$$

where $\boldsymbol{x}_{g,i}$, $\boldsymbol{x}_{e,i}$, $\boldsymbol{x}_{v,i}$ are the generator, exciter, and governor states for machine i, and $\boldsymbol{x}_{\text{ren}}$ collects all renewable component states.

The state dimension is:

$$n = \sum_{i=1}^{N_{\text{gen}}} (n_{g,i} + n_{e,i} + n_{v,i}) + \sum_{r=1}^{N_{\text{ren}}} n_{\text{ren},r} \tag{75}$$

For a heterogeneous system, $n_{e,i}$ and $n_{v,i}$ vary per machine (e.g., EXDC2 has 4 exciter states while ESST3A has 5).

8.2 Numerical Integration

The ODE is integrated using SciPy's `solve_ivp` with the explicit Runge-Kutta method (RK45, the default) or implicit methods (Radau, BDF) for stiff systems. Typical solver settings are:

- Relative tolerance: 10^{-6}
- Absolute tolerance: 10^{-8}
- Maximum step size: 0.01 s (to capture fast dynamics)

Event detection handles fault transitions: the solver is called in three phases (pre-fault, during-fault, post-fault) with the Ybus switched at each transition.

8.3 Evaluation Loop

At each evaluation of $f(t, \boldsymbol{x})$, the following computational sequence is executed:

1. **State extraction**: Unpack flat vector into per-machine component states
2. **Fault determination**: Check if $t_{\text{fault}} \leq t < t_{\text{fault}} + t_{\text{dur}}$
3. **Network solution**: Solve for I_d, I_q, V_d, V_q using the iterative method of Section 6
4. **Exciter output**: Compute $E_{fd,i}$ from exciter states and terminal voltage
5. **Governor output**: Compute $T_{m,i}$ from governor states and rotor speed
6. **Component dynamics**: Evaluate $\dot{\boldsymbol{x}}_{g,i}$, $\dot{\boldsymbol{x}}_{e,i}$, $\dot{\boldsymbol{x}}_{v,i}$ for each machine
7. **Renewable dynamics**: Evaluate all WT3 sub-component derivatives
8. **Assembly**: Concatenate derivatives into flat output vector

9 Performance Optimization

Two complementary optimization strategies were implemented to accelerate the simulation, achieving a combined speedup of approximately 6×.

9.1 Numba JIT Compilation

All seven component dynamics functions were converted to Numba `@njit(cache=True)` compiled versions. The key transformation replaces Python dictionary-based function signatures with flat NumPy array interfaces:

$$f_{\text{dict}}(\boldsymbol{x}, \texttt{ports\_dict}, \texttt{meta\_dict}) \rightarrow f_{\text{jit}}(\boldsymbol{x}, \boldsymbol{p}, \boldsymbol{m}) \tag{76}$$

where $\boldsymbol{p} \in \mathbb{R}^{n_p}$ is a pre-allocated ports array and $\boldsymbol{m} \in \mathbb{R}^{n_m}$ is a packed metadata array.

Table 1: JIT-compiled component functions

Component	States	Ports	Meta	Special Handling
GENROU	7	6	8	Pre-computed K_{fd}
EXDC2	4	1	13	Pre-computed saturation A, B
EXST1	4	2	13	Dynamic field current limits
ESST3A	5	6	24	Explicit real/imag arithmetic
IEEEX1	5	2	15	Pre-computed saturation A, B
TGOV1	2	1	10	Deadband logic
IEEEG1	6	2	22	Normalized power fractions

9.1.1 Implementation Details

Metadata Packing. Each component provides a `pack_meta()` function that converts the parameter dictionary to a flat `float64` array with named index constants. For models with saturation (EXDC2, IEEEX1), the coefficients A and B in (23) are pre-computed during packing.

Port Buffer Pre-allocation. Fixed-size port arrays are allocated once per machine and reused across all ODE evaluations, eliminating per-call allocation overhead.

JIT Infrastructure. A central registry (`numba_kernels.py`) maps model names to their JIT functions:

```
JIT_REGISTRY = {
    'GENROU': {
        'dynamics_jit': genrou_dynamics_jit,
        'output_jit': None,
        'pack_meta': pack_genrou_meta,
        'ports_size': 6, 'meta_size': 8
    },
    ...
}
```

Listing 1: JIT registry structure

Warmup. All JIT functions are triggered once with dummy data before simulation begins, ensuring compilation overhead does not affect timing measurements.

9.2 LU Factorization Caching

Profiling revealed that the network solver (`np.linalg.solve`) consumed 72% of total simulation time through approximately 125,000 calls, while component dynamics accounted for only 3.5%. Since the augmented admittance matrix $\mathbf{Y}_{\text{aug}}$ is constant between fault transitions, LU factorization caching was implemented:

$$\underbrace{\mathbf{Y}_{\text{aug}} = \mathbf{L}\mathbf{U}}_{\text{factored once per fault phase}} \quad \Longrightarrow \quad \boldsymbol{V} = \mathbf{U}^{-1}\mathbf{L}^{-1}\boldsymbol{I}_{\text{total}} \tag{77}$$

The cache key is the tuple $(\texttt{use\_fault}, \texttt{fault\_bus}, Z_f)$, which changes only at fault application and clearance events. This reduces the per-call cost from $\mathcal{O}(n^3)$ (full factorization) to $\mathcal{O}(n^2)$ (forward/back substitution).

Algorithm 1: Cached LU Network Solve

Input: $\mathbf{Y}_{\text{aug}}$, $\boldsymbol{I}_{\text{total}}$, fault state key k

if $k \neq k_{\text{cached}}$ **then**
  $(\mathbf{L}, \mathbf{U}, \boldsymbol{\pi}) \leftarrow \text{LU\_Factor}(\mathbf{Y}_{\text{aug}})$;
  $k_{\text{cached}} \leftarrow k$;
$\boldsymbol{V} \leftarrow \text{LU\_Solve}(\mathbf{L}, \mathbf{U}, \boldsymbol{\pi}, \boldsymbol{I}_{\text{total}})$;

9.3 Performance Results

Table 2 summarizes the speedup achieved on two benchmark systems.

Table 2: Simulation performance comparison ($t_{\text{end}} = 15$ s)

System	Machines	States	Before	After	Speedup
Kundur 2-area	4	52	53 s	9 s	5.9×
IEEE 14-bus	5	85	33 s	5 s	6.6×

10 Test Systems and Validation

10.1 Kundur Two-Area Four-Machine System

The Kundur two-area system [2] is a widely used benchmark for inter-area oscillation studies. It consists of:

- 11 buses, 4 synchronous generators (GENROU, $S_n = 900$ MVA each)
- 2 areas connected by a weak tie-line
- Excitation: EXDC2 on all machines
- Governor: TGOV1 on all machines
- Total load: $\approx 28\,\text{pu}$ on 100 MVA base
- State dimension: 52 (7 gen + 4 exc + 2 gov per machine)

10.2 IEEE 14-Bus System

The IEEE 14-bus system provides validation for heterogeneous component configurations:

- 14 buses, 5 generators (GENROU), 20 lines/transformers
- Mixed exciters: ESST3A (3 machines), EXST1 (2 machines)
- Mixed governors: IEEEG1 (3 machines), TGOV1 (2 machines)
- State dimension: 85 (varying per machine)

10.3 IEEE 39-Bus (New England) System

The IEEE 39-bus system tests scalability:

- 39 buses, 10 generators (GENROU)
- All ESST3A exciters, all IEEEG1 governors
- State dimension: 180

10.4 Validation Criteria

The implementation is validated against the following criteria:

Equilibrium Accuracy. After power flow initialization, the maximum derivative norm should satisfy $\max_i |\dot{x}_i(0)| < 10^{-10}$. Achieved values:

- Kundur system: $\max |\dot{\boldsymbol{x}}| = 4.2 \times 10^{-13}$
- IEEE 14-bus: $\max |\dot{\boldsymbol{x}}| = 2.9 \times 10^{-13}$

No-Fault Steady State. With no disturbance, rotor angles should remain constant ($\Delta\delta < 0.01^\circ$ over 15 s) and generator speeds should be $\omega = 1.0 \pm 10^{-6}$ pu.

JIT vs. Non-JIT Consistency. The maximum absolute difference between JIT and dictionary-based simulation results should be $< 10^{-5}$ over the entire trajectory, confirming numerical equivalence.

Post-Fault Recovery. Following a three-phase fault (typically $Z_f = 0.01j$ pu for 150 ms), all generators should recover synchronism with damped oscillations, demonstrating transient stability.

11 Summary

This paper presented the methodology and implementation of a modular Port-Hamiltonian framework for power system dynamic simulation. The key contributions are:

1. A fully modular, JSON-configured simulation platform supporting arbitrary multi-machine systems with heterogeneous component models (GENROU, EXDC2/EXST1/ESST3A/IE exciters, TGOV1/IEEEG1 governors, and Type-3 wind turbines).
2. Rigorous implementation of IEEE-standard component models with proper initialization procedures achieving machine-precision equilibrium ($\max |\dot{\boldsymbol{x}}| < 10^{-12}$).
3. An iterative current-balance network solver with constant power load modelling, Kron reduction, and Park transformations for the generator–network interface.
4. Performance optimization through Numba JIT compilation of all component dynamics functions and LU factorization caching in the network solver, achieving approximately 6× speedup.
5. A comprehensive Type-3 Wind Turbine model (21 states per unit) with seven interconnected sub-components following the PHS framework.
6. Validation on standard benchmark systems (Kundur 4-machine, IEEE 14-bus, IEEE 39-bus) demonstrating numerical accuracy, equilibrium quality, and transient stability.

The Port-Hamiltonian formulation provides energy-based stability certificates through the Hamiltonian as a natural Lyapunov function, enabling passivity analysis and region-of-attraction estimation directly from the system model.

References

[1] A. J. van der Schaft, *L2-Gain and Passivity Techniques in Nonlinear Control*, 2nd ed. London: Springer, 2000.

[2] P. Kundur, *Power System Stability and Control*. New York: McGraw-Hill, 1994.

[3] IEEE Std 421.5-2016, *IEEE Recommended Practice for Excitation System Models for Power System Stability Studies*, IEEE, 2016.

[4] IEEE Std 1110-2019, *IEEE Guide for Synchronous Generator Modeling Practices and Parameter Verification with Applications in Power System Stability Analyses*, IEEE, 2019.

[5] P. W. Sauer and M. A. Pai, *Power System Dynamics and Stability*. Upper Saddle River, NJ: Prentice Hall, 1998.

[6] WECC Renewable Energy Modeling Task Force, "WECC Wind Power Plant Dynamic Modeling Guide," Western Electricity Coordinating Council, Tech. Rep., Apr. 2014.

[7] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: A LLVM-based Python JIT compiler," in *Proc. 2nd Workshop on the LLVM Compiler Infrastructure in HPC*, 2015, pp. 1–6.

[8] P. Virtanen *et al.*, "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, no. 3, pp. 261–272, 2020.

[9] F. Milano, *Power System Modelling and Scripting*. London: Springer, 2010.

[10] J. Machowski, J. W. Bialek, and J. R. Bumby, *Power System Dynamics: Stability and Control*, 2nd ed. Chichester: Wiley, 2008.