Grid-Tie Inverter Simulation Software

May 31, 2025

1 Introduction

This document details the functioning logic, simulation logic, physics models, and algorithms of Grid-Tie Inverter Simulation software. The software simulates a grid-connected inverter system, supporting various DC sources, phase topologies, multilevel inverter configurations, pulse-width modulation (PWM) techniques, maximum power point tracking (MPPT) algorithms, control strategies, islanding detection, and grid interactions. The software is written in Python using numerical methods and simplified physics models to emulate real-world inverter behavior.

2 Functioning Logic

The simulation models a grid-tie inverter that converts DC power from sources like photovoltaic (PV) panels, batteries, or fuel cells into AC power synchronized with the grid.

- **DC Source**: Models the input power source (Fixed, PV Panel, Battery, Fuel Cell, Hybrid).
- **Inverter Topology**: Supports single-phase or three-phase configurations with optional multilevel topologies.
- **PWM Techniques**: Implements Multicarrier or Space Vector PWM for switching control.
- **MPPT Algorithms**: Optimizes power extraction from PV sources.
- **Control Strategies**: Includes PI, PR, Sliding Mode, and MPC for voltage and current regulation.
- **Islanding Detection**: Detects grid disconnection using passive and active methods.
- **Phase-Locked Loop (PLL)**: Synchronizes inverter output with the grid.
- **Grid Simulation**: Models grid voltage with faults (sag, swell, harmonics, frequency shift).
- **Adaptive Control**: Uses Q-learning for dynamic modulation index adjustment.

The simulation operates in discrete time steps, generating waveforms for voltage and current, which are processed through control algorithms and visualized or exported as data.

3 Simulation Logic

The simulation logic is orchestrated by the `InverterSimulation` class, which integrates all components and updates them based on parameters and time steps. The core simulation flow is as follows:

1. **Initialization**: Sets default parameters (e.g., DC voltage = 400 V, frequency = 50 Hz, modulation index = 0.8, time step = 0.001 s, time window = 0.04 s).
2. **Parameter Update**: Updates DC voltage, frequency, modulation index, and other parameters from user inputs via `update_simulation_parameters`.
3. **DC Source Update**: Computes the DC voltage based on the selected source type (e.g., PV Panel, Battery) using `DCSource.update`.
4. **MPPT Application**: If enabled, adjusts the DC voltage to maximize power extraction using the selected MPPT algorithm.
5. **Grid Synchronization**: Uses PLL to estimate the grid phase angle via `PLL.update`.
6. **Islanding Detection**: Checks for grid disconnection using `IslandingDetector.detect`. If detected, outputs zero voltage and current.
7. **Waveform Generation**: Generates voltage and current waveforms using the selected phase topology (`SinglePhaseTopology` or `ThreePhaseTopology`) or multilevel topology (`NPCInverter`, `MMCInverter`, etc.).
8. **Control Application**: Applies the selected control strategy (PI, PR, Sliding Mode, MPC) to regulate output voltage and current.
9. **Design Application**: Applies transformerless or transformer-based design effects (e.g., DC offset or efficiency loss).
10. **Time Advancement**: Increments the simulation time for the next cycle.

The simulation supports time-domain and frequency-domain analyses, with data export capabilities for further analysis.

4 Physics Models

The software employs simplified physics models to simulate inverter behavior. Below are the key models used:

4.1 DC Source Models

The `DCSource` class and its subclasses model different power sources:

- **Fixed DC Source**: Outputs a constant voltage (default 400 V).
- **PV Panel**: Models a PV panel with open-circuit voltage $V_{\text{oc,STC}} = 500\,\text{V}$, short-circuit current $I_{\text{sc,STC}} = 10\,\text{A}$, and temperature/irradiance dependence:

$$I_{\text{sc}} = I_{\text{sc,STC}} \cdot \frac{G}{G_{\text{STC}}} \cdot (1 + \alpha(T - T_{\text{STC}}))$$

$$V_{\text{oc}} = V_{\text{oc,STC}} \cdot (1 + \beta(T - T_{\text{STC}}))$$

Current is computed as $I = I_{\text{sc}} \cdot \left(1 - \left(\frac{V}{V_{\text{oc}}}\right)^2\right)$, and voltage is iteratively adjusted toward the maximum power point (MPP) using an empirical factor $k = 0.05$.

- **Battery**: Models voltage based on state of charge (SOC):

$$V = V_{\min} + (V_{\max} - V_{\min}) \cdot \text{SOC}$$

  where $V_{\min} = 0.9V_{\text{nom}}$, $V_{\max} = 1.1V_{\text{nom}}$, and SOC decreases with discharge rate $C_{\text{rate}} = 0.1$.

- **Fuel Cell**: Models voltage with linear drop based on load current:

$$V = V_{\text{nom}} \cdot \eta - 0.05I_{\text{load}}$$

  where $\eta = 0.6$ is the efficiency, and I_{load} is clipped to 020 A.

- **Hybrid Source**: Combines PV, battery, and fuel cell voltages with weights (0.5, 0.3, 0.2, respectively).

All voltages are clipped to 100800 V for safety.

4.2 Inverter Topologies

The `InverterTopology` and `MultilevelInverter` classes model single-phase and three-phase inverters, with multilevel options:

- **Single-Phase Topology**: Generates sinusoidal voltage $V = 230\sqrt{2}\sin(2\pi ft)$ and current proportional to the modulation index.

- **Three-Phase Topology**: Generates three-phase voltages with 120ř phase shifts: $V_i = 230\sqrt{2}\sin(2\pi ft + \theta_i)$, where $\theta_i = \{0, -2\pi/3, 2\pi/3\}$.

- **Multilevel Topologies**: Include Neutral Point Clamped (NPC), Flying Capacitor, Cascaded H-Bridge, Modular Multilevel Converter (MMC), Reduced Switch Count, and Hybrid CHB+NPC. Each defines discrete voltage levels (e.g., 5-level for NPC: $[-V_{\text{dc}}/2, -V_{\text{dc}}/4, 0, V_{\text{dc}}/4, V_{\text{dc}}/2$ and applies PWM to generate stepped waveforms.

4.3 Grid Model

The `GridSimulation` class models the grid as a sinusoidal voltage source $V = 230\sqrt{2}\sin(2\pi ft)$, with configurable faults:

- **Sag**: Reduces voltage by 20%.

- **Swell**: Increases voltage by 20%.

- **Harmonics**: Adds 5th and 7th harmonics (5% amplitude each).

- **Frequency Shift**: Increases frequency by 2 Hz.

Impedance effects are modeled as $V_{\text{drop}} = RI_{\text{load}} + L \cdot 2\pi fI_{\text{load}}$, with R and L adjustable for weak grid conditions.

5 Algorithms

The simulation implements several algorithms for control, power optimization, and synchronization:

5.1 MPPT Algorithms

The `MaximaleLeistungspunktverfolgung` module implements MPPT algorithms to maximize PV power output:

- **Perturb and Observe (P&O)**: Perturbs voltage by $\Delta V = 5\,\mathrm{V}$ and observes power change. If power increases, continues in the same direction; otherwise, reverses direction.
- **Incremental Conductance**: Compares incremental conductance $\frac{dI}{dV}$ to instantaneous conductance $-\frac{I}{V}$. Adjusts voltage by $\Delta V = 5\,\mathrm{V}$ based on:

$$\begin{cases} \frac{dI}{dV} > -\frac{I}{V} & \text{increase } V \\ \frac{dI}{dV} < -\frac{I}{V} & \text{decrease } V \\ \frac{dI}{dV} = -\frac{I}{V} & \text{MPP reached} \end{cases}$$

- **Constant Voltage**: Maintains voltage at 80% of $V_{\mathrm{oc}} = 500\,\mathrm{V}$.
- **Constant Current**: Targets 90% of $I_{\mathrm{sc}} = 10\,\mathrm{A}$, adjusting voltage accordingly.
- **Ripple Correlation Control (RCC)**: Uses power and voltage ripple correlation at 100 Hz:

$$\Delta V = -k \cdot \frac{dP}{dt} \cdot \frac{dV}{dt}, \quad k = 0.1$$

  Voltage is adjusted to minimize correlation, driving operation toward the MPP.

5.2 Control Strategies

The `InverterSimulation.apply_control` method implements four control strategies:

- **PI Control**: Uses proportional-integral control for voltage and current:

$$V_{\mathrm{out}} = V + K_{p,v} e_v + K_{i,v} \int e_v \, dt, \quad I_{\mathrm{out}} = I + K_{p,i} e_i + K_{i,i} \int e_i \, dt$$

  where $K_{p,v} = 0.5$, $K_{i,v} = 10$, $K_{p,i} = 0.2$, $K_{i,i} = 5$.
- **PR Control**: Adds resonant terms tuned to the grid frequency:

$$V_{\mathrm{out}} = V + K_{p,v} e_v + K_{r,v} \sin(2\pi f t + \theta) e_v$$

$$I_{\mathrm{out}} = I + K_{p,i} e_i + K_{r,i} \sin(2\pi f t + \theta) e_i$$

  where $K_{r,v} = 50$, $K_{r,i} = 20$.
- **Sliding Mode Control**: Uses sliding surfaces $s_v = e_v + \lambda_v (e_v - e_{v,\mathrm{prev}})$, $s_i = e_i + \lambda_i (e_i - e_{i,\mathrm{prev}})$:

$$V_{\mathrm{out}} = V + K_v \mathrm{sgn}(s_v), \quad I_{\mathrm{out}} = I + K_i \mathrm{sgn}(s_i)$$

  where $\lambda_v = 100$, $\lambda_i = 50$, $K_v = 10$, $K_i = 5$.
- **Model Predictive Control (MPC)**: Predicts voltage and current one step ahead:

$$V_{\mathrm{pred}} = V + \frac{\Delta t}{L}(V_{\mathrm{ref}} - V - RI)$$

$$I_{\mathrm{pred}} = I + \frac{\Delta t}{L}(V_{\mathrm{pred}} - V_{\mathrm{ref}})$$

  Outputs are selected if the cost $(V_{\mathrm{ref}} - V_{\mathrm{pred}})^2 + (I_{\mathrm{ref}} - I_{\mathrm{pred}})^2 < 2$.

5.3 Q-Learning Controller

The `AdaptiveKontrollstrategien` module implements a Q-learning controller to adjust the modulation index dynamically:

- **State**: Discretized voltage tracking error ($-50\,\mathrm{V}$ to $50\,\mathrm{V}$, 10 bins).
- **Action**: Modulation index adjustment (-0.1 to 0.1, 5 steps).
- **Reward**: Negative of the voltage error magnitude, $r = -|V_{\text{grid}} - V_{\text{out}}|$.
- **Q-Table Update**:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right)$$

  where $\alpha = 0.1$, $\gamma = 0.9$, and exploration rate $\epsilon = 0.1$.

The controller operates in "Train" or "Apply" modes, updating the Q-table during training and applying learned actions otherwise.

5.4 PWM Techniques

The `MultilevelInverter.apply_pwm` method implements:

- **Multicarrier PWM**: Compares a reference sinusoid to multiple carrier thresholds to select discrete voltage levels.
- **Space Vector PWM (SVPWM)**: Discretizes the reference to the nearest voltage level with a scaling factor of 1.1 for improved utilization.

5.5 Phase-Locked Loop (PLL)

The `Phasenregelkreis` module implements a Second-Order Generalized Integrator (SOGI)-based PLL:

$$\begin{aligned}
v_{\text{error}} &= V_{\text{grid}} - v \\
v &\leftarrow v + \Delta t \cdot (k v_{\text{error}} \omega - q\omega^2) \\
q &\leftarrow q + \Delta t \cdot v \\
\theta = \arctan\left(\frac{q}{v}\right), &\quad \phi_{\text{error}} = \sin(\theta - \phi) \\
\phi &\leftarrow \phi + (\omega + K_p \phi_{\text{error}} + K_i \int \phi_{\text{error}}\, dt) \cdot \Delta t
\end{aligned}$$

where $k = 0.5$, $K_p = 2.0$, $K_i = 100.0$.

5.6 Islanding Detection

The `IslandingDetector` uses passive and active methods:

- **Passive**: Detects over/under voltage ($0.88V_{\text{nom}} < V < 1.1V_{\text{nom}}$) and frequency ($f_{\text{nom}} - 1 < f < f_{\text{nom}} + 1$).
- **Active**: Applies frequency shift ($\pm 0.5\,\mathrm{Hz}$) every 100 ms and reactive power injection ($Q = 0.05V_{\text{peak}}$). Islanding is detected if reactive power variation exceeds 10% of nominal voltage.