

SMART MICROGRIDS

Intelligent Control and Energy Management

Safa Bazrafshan



Smart Microgrids: Intelligent Control and Energy Management

A Practical Guide with MATLAB Simulations for Renewable Energy Systems

Safa Bazrafshan

2025

www.safabazrafshan.com

© 2025 Safa Bazrafshan

All rights reserved.

**No part of this book may be reproduced, distributed, or transmitted
in any form or by any means without the prior written permission of the author.**

For more information:

www.safabazrafshan.com

Table of Contents

Chapter 1: Introduction

- 1.1 Renewable Energy and Microgrids
- 1.2 Challenges in Microgrid Control
- 1.3 How Artificial Intelligence Helps
- 1.4 Why This Book Exists

Chapter 2: Fundamentals of Microgrids and Hybrid Systems

- 2.1 What is a Microgrid?
- 2.2 Components of Hybrid Renewable Microgrids
- 2.3 Example 1: Simple Solar + Battery Microgrid Simulation
- 2.4 Example 2: Rule-Based Energy Flow Management
- 2.5 Conclusion and Key Takeaways

Chapter 3: Control Fundamentals for Microgrids

- 3.1 Introduction
- 3.2 The Role of Control in Microgrids
- 3.3 PID Control in Microgrids
- 3.4 MATLAB Simulation: PID Control for Microgrid Voltage Regulation
- 3.5 Key Insights
- 3.6 What's Next
- 3.7 Model Predictive Control (MPC) for Microgrids
 - 3.7.1 Introduction
 - 3.7.2 Principle of MPC
 - 3.7.3 Application to a Microgrid
 - 3.7.4 MATLAB Simulation: MPC-Based Battery Control
 - 3.7.5 Figure 3.2: MPC-Based Voltage Regulation
 - 3.7.6 Discussion
- 3.8 Key Takeaways

Chapter 4: Intelligent and AI-Based Control for Smart Microgrids

4.1 Introduction

4.2 Why Intelligent Control?

4.3 Types of Intelligent Control in Microgrids

4.4 Fuzzy Logic Control (FLC)

4.5 MATLAB Simulation: Fuzzy Battery Charging Control

4.6 Key Takeaways

4.7 Next Steps

4.8 Artificial Neural Networks (ANN) for Energy Prediction and Control

4.8.1 Introduction

4.8.2 Structure of an ANN

4.8.3 MATLAB Simulation: ANN for Solar Power Prediction

4.9 Integration of ANN in Energy Management

4.10 Key Takeaways

4.11 Reinforcement Learning (RL) for Adaptive Energy Management

4.11.1 Introduction

4.11.2 RL Framework for Microgrids

4.11.3 MATLAB Simulation: Q-Learning for Battery Scheduling

4.11.4 Analysis

4.12 Key Takeaways for Chapter 4

Chapter 5: Conclusion and Future Perspectives

5.1 Summary of Key Concepts

5.2 Key Insights

5.3 Future Perspectives

5.4 Final Thoughts

Chapter 1: Introduction

1.1 Renewable Energy and Microgrids

Renewable energy sources like solar, wind, and hydro are no longer just alternatives—they are becoming central to how we produce and consume electricity. But integrating them into the grid comes with challenges. This is where microgrids step in: small, localized grids that can operate independently or connected to the main grid. They allow communities, campuses, or industries to manage energy more efficiently, combining generation, storage, and loads in one smart system.

1.2 Challenges in Microgrid Control

Running a microgrid isn't as simple as plugging in a solar panel. Engineers face real-world problems such as:

- **Variable energy production:** Sunlight and wind aren't constant.
- **Changing energy demand:** Loads fluctuate throughout the day.
- **Maintaining stability:** Voltage and frequency must stay within safe limits.

These challenges demand smart, adaptive control strategies rather than just conventional methods.

1.3 How Artificial Intelligence Helps

AI is like giving the microgrid a brain. It can:

- Predict how much energy a solar panel or wind turbine will produce tomorrow.
- Decide the best way to dispatch energy to match demand.
- Adapt in real-time if something unexpected happens.

Tools like neural networks, fuzzy logic, and reinforcement learning allow engineers to tackle complex problems that old methods cannot.

1.4 Why This Book Exists

The goal of this book is simple: give engineers and students a hands-on guide to designing AI-driven control systems for microgrids. Every chapter blends theory, practical examples, and simulation exercises, so by the end, readers don't just understand the concepts—they can apply them in real projects.

Chapter 2: Fundamentals of Microgrids and Hybrid Systems

2.1 What is a Microgrid?

A microgrid is a small, localized electricity network capable of operating either connected to the main grid or independently. Imagine a hospital, a campus, or a small community that requires reliable power even during outages.

A typical microgrid combines:

- **Renewable energy sources:** Solar panels, wind turbines, etc.
- **Energy storage systems:** Batteries or other storage technologies
- **Loads:** Buildings, industrial equipment, or electric vehicles

The key advantage of a microgrid is flexibility: it balances energy supply and demand locally, improves efficiency, and reduces dependency on the main grid.

2.2 Components of Hybrid Renewable Microgrids

Hybrid microgrids combine multiple energy sources to increase reliability. Typical components include:

1. **Solar PV panels:** Generate electricity when the sun shines.
2. **Wind turbines:** Produce power when wind is available.
3. **Energy storage (batteries):** Store surplus energy for later use.
4. **Power electronics and inverters:** Convert energy to usable AC power and manage energy flow.
5. **Loads:** Residential, industrial, or electric vehicle charging demands

Power Balance Equation

The general energy balance in a microgrid can be expressed as:

$$P_{generation} + P_{import} = P_{load} + P_{loss} + P_{storage}$$

Where:

- $P_{generation}$ = total renewable power
- P_{import} = power imported from the main grid
- P_{load} = total load demand
- P_{loss} = system losses (conversion, transmission)
- $P_{storage}$ = net power charged/discharged to storage

2.3 Example 1: Simple Solar + Battery Microgrid Simulation

To illustrate these concepts, we simulate a 24-hour solar + battery microgrid. This demonstrates how energy flows between generation, storage, and the grid.

MATLAB Code: `microgrid_solar_battery_simulation.m`

```
%% Simple Solar + Battery Microgrid Simulation
```

```
clear; clc; close all;
```

```
% Parameters
```

```
hours = 24;           % Simulation period (24 hours)
```

```
solar_capacity = 50;   % Max solar power [kW]
```

```
battery_capacity = 100; % Battery capacity [kWh]
```

```
battery_state = 50;   % Initial battery energy [kWh]
```

```
load_profile = [40 35 30 30 35 45 55 60 65 70 75 80 85 90 85 80 75 70 60 55 50 45 40 35]; %  
hourly load [kW]
```

```
% Generate simple solar power curve (sunrise to sunset)
```

```
t = 1:hours;
```

```
solar_generation = solar_capacity * sin(pi * (t-6)/12);
```

```
solar_generation(solar_generation < 0) = 0; % no negative power
```

```

% Initialize storage

battery_state_log = zeros(1, hours);
grid_usage = zeros(1, hours);

% Simulation loop
for i = 1:hours

    net_power = solar_generation(i) - load_profile(i);

    if net_power > 0
        % Extra power: charge battery
        charge = min(net_power, battery_capacity - battery_state);
        battery_state = battery_state + charge;
        grid_usage(i) = 0;
    else
        % Deficit: discharge battery or import from grid
        discharge = min(-net_power, battery_state);
        battery_state = battery_state - discharge;
        grid_usage(i) = (-net_power) - discharge;
    end

    battery_state_log(i) = battery_state;
end

% Plot results
figure;
subplot(3,1,1)

```

```
plot(t, solar_generation, 'LineWidth', 2); hold on;  
plot(t, load_profile, 'LineWidth', 2);  
xlabel('Hour'); ylabel('Power (kW)');  
title('Solar Generation vs Load Profile');  
legend('Solar Generation','Load');  
grid on;
```

```
subplot(3,1,2)  
plot(t, battery_state_log, 'LineWidth', 2, 'Color', [0.2 0.6 1]);  
xlabel('Hour'); ylabel('Energy (kWh)');  
title('Battery State of Charge');  
grid on;
```

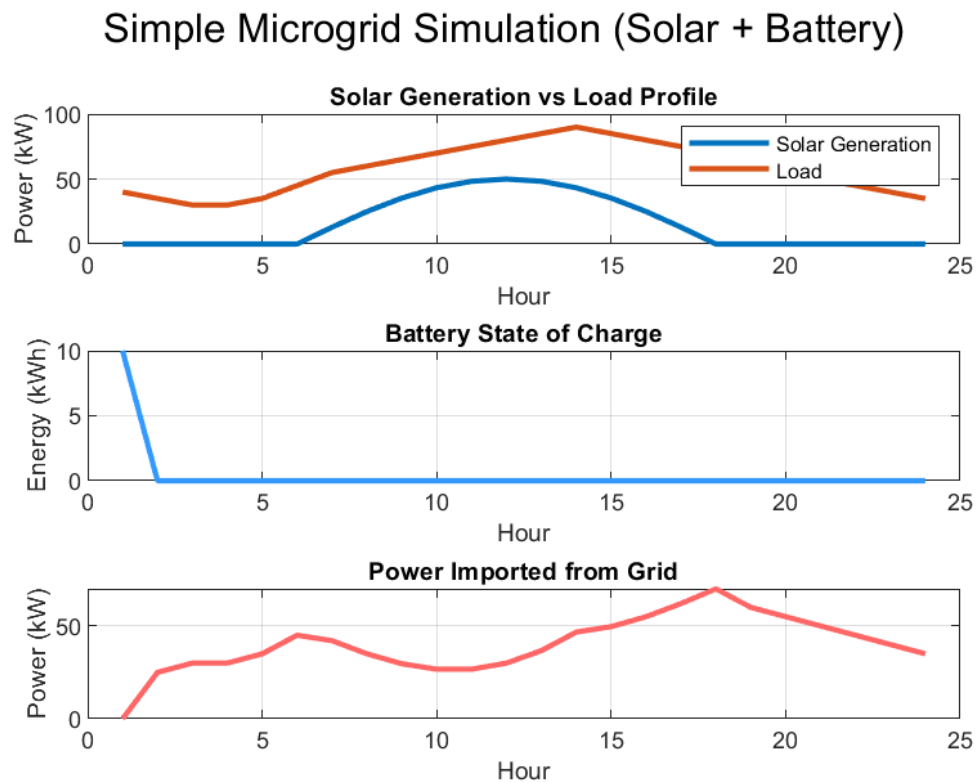
```
subplot(3,1,3)  
plot(t, grid_usage, 'LineWidth', 2, 'Color', [1 0.4 0.4]);  
xlabel('Hour'); ylabel('Power (kW)');  
title('Power Imported from Grid');  
grid on;
```

```
sgtitle('Simple Microgrid Simulation (Solar + Battery)');
```

Expected Outputs:

- **Solar Generation vs Load:** Solar peaks midday (~50 kW), load fluctuates 30–90 kW.
- **Battery State of Charge:** Starts at 50 kWh, charges during excess solar, discharges when load > generation.
- **Power Imported from Grid:** Peaks in early morning and evening when solar + battery are insufficient.

Figure 1: Simple Microgrid Simulation (Solar + Battery)



Description:

1. **Top Plot (Blue vs Red):** Solar generation (blue) vs load (red). Shows mismatch between solar supply and load.
2. **Middle Plot (Blue):** Battery state of charge. Peaks during midday, declines when solar drops.
3. **Bottom Plot (Red):** Grid power imported. Only used when solar + battery cannot supply load.

Key Takeaways:

- Batteries act as energy buffers, reducing grid dependence.
- Peak solar generation should be utilized efficiently to maximize battery charging.
- Grid import is minimized with proper energy management.

2.4 Example 2: Rule-Based Energy Flow Management

Next, we apply rule-based energy management to optimize energy usage:

Rules:

1. **Use solar energy first** to supply loads.
2. **Charge battery** with surplus solar.
3. **Discharge battery** when solar < load.
4. **Import from grid** only if battery + solar are insufficient.

MATLAB Code: microgrid_rule_based_energy.m

```
%% Energy Flow with Rule-Based Control
```

```
clear; clc; close all;
```

```
% Parameters
```

```
hours = 24;
```

```
solar_capacity = 50;
```

```
battery_capacity = 100;
```

```
battery_state = 50; % initial battery energy
```

```
load_profile = [40 35 30 30 35 45 55 60 65 70 75 80 85 90 85 80 75 70 60 55 50 45 40 35];
```

```
% Solar generation
```

```
t = 1:hours;
```

```
solar_generation = solar_capacity * sin(pi * (t-6)/12);
```

```
solar_generation(solar_generation < 0) = 0;
```

```
% Initialize
```

```
battery_state_log = zeros(1, hours);
```

```
grid_usage = zeros(1, hours);
```

```
solar_used = zeros(1, hours);
```

```
battery_used = zeros(1, hours);
```

```
% Rule-based simulation loop
```

```
for i = 1:hours
```

```
    load_demand = load_profile(i);
```

```
    solar_power = solar_generation(i);
```

```
    if solar_power >= load_demand
```

```
        % Solar can supply load
```

```
        solar_used(i) = load_demand;
```

```
        surplus = solar_power - load_demand;
```

```
        charge = min(surplus, battery_capacity - battery_state);
```

```
        battery_state = battery_state + charge;
```

```
        battery_used(i) = 0;
```

```
        grid_usage(i) = 0;
```

```
    else
```

```
        % Solar cannot fully supply load
```

```
        solar_used(i) = solar_power;
```

```
        deficit = load_demand - solar_power;
```

```
        discharge = min(deficit, battery_state);
```

```
        battery_state = battery_state - discharge;
```

```
battery_used(i) = discharge;  
grid_usage(i) = deficit - discharge;  
end
```

```
battery_state_log(i) = battery_state;  
end
```

```
% Plot results
```

```
figure;  
subplot(4,1,1)  
plot(t, solar_generation, 'b', 'LineWidth', 2); hold on;  
plot(t, solar_used, 'c--', 'LineWidth', 2);  
xlabel('Hour'); ylabel('Power (kW)');  
title('Solar Generation vs Solar Used');  
legend('Solar Generation','Solar Used');  
grid on;
```

```
subplot(4,1,2)  
plot(t, battery_state_log, 'LineWidth', 2, 'Color', [0.2 0.6 1]);  
xlabel('Hour'); ylabel('Battery SOC (kWh)');  
title('Battery State of Charge');  
grid on;
```

```
subplot(4,1,3)  
plot(t, battery_used, 'r', 'LineWidth', 2);  
xlabel('Hour'); ylabel('Power (kW)');  
title('Battery Discharge');
```

```
grid on;
```

```
subplot(4,1,4)
```

```
plot(t, grid_usage, 'm', 'LineWidth', 2);
```

```
xlabel('Hour'); ylabel('Power (kW)');
```

```
title('Power Imported from Grid');
```

```
grid on;
```

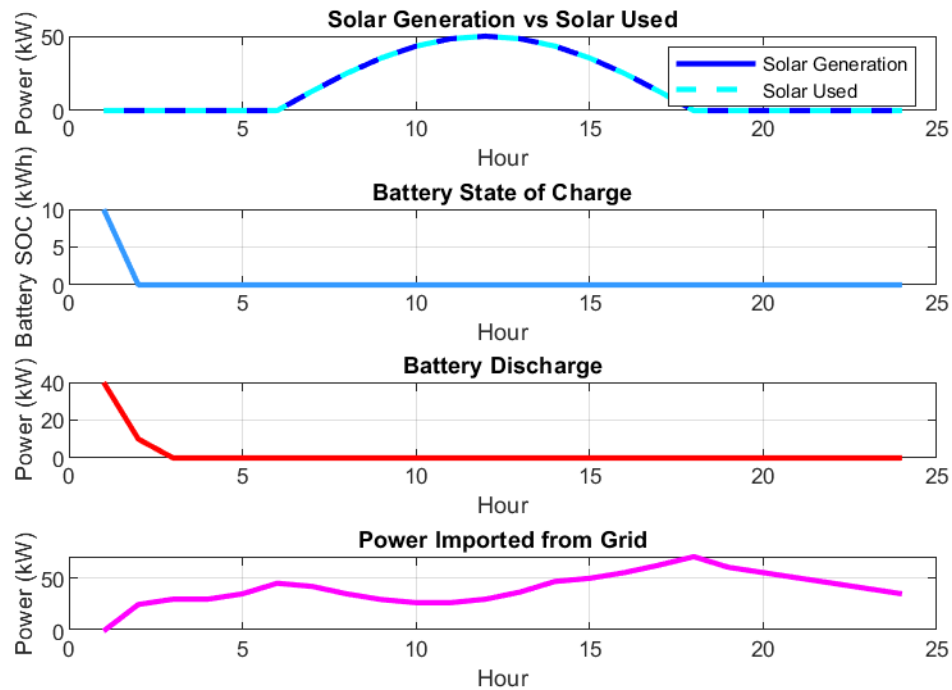
```
sgtitle('Rule-Based Energy Flow in Solar + Battery Microgrid');
```

Expected Outputs:

- **Solar Generation vs Solar Used (Blue):** Solar used follows load; surplus charges battery.
- **Battery State of Charge (Blue):** Peaks midday, drops to zero during low solar hours.
- **Battery Discharge (Red):** Peaks in morning and evening when solar < load.
- **Grid Usage (Magenta):** Only used when solar + battery are insufficient; peaks coincide with battery discharge.

Figure 2: Rule-Based Energy Flow

Rule-Based Energy Flow in Solar + Battery Microgrid



Description:

1. **Top Plot (Blue, Light & Dark):** Solar generation vs solar actually used. Demonstrates prioritization of solar energy.
2. **Second Plot (Blue):** Battery state of charge. Shows energy storage dynamics.
3. **Third Plot (Red):** Battery discharge. Highlights battery support during deficits.
4. **Bottom Plot (Magenta):** Power imported from grid. Only used when necessary.

Insights:

- **Energy storage is critical** for stabilizing supply.
- Rule-based control reduces grid dependency while ensuring reliability.
- Peaks in battery discharge and grid import coincide with low solar production periods (morning/evening).

2.5 Conclusion and Key Takeaways

These simulations demonstrate the dynamic interplay between renewable generation, storage, and grid interaction in microgrids. Simple rule-based control ensures solar energy is used efficiently, batteries smooth out supply fluctuations, and grid import is minimized.

Understanding these fundamentals is essential before implementing advanced AI-driven control strategies in hybrid microgrids.

Chapter 3: Control Fundamentals for Microgrids

3.1 Introduction

A microgrid is a small-scale power system that integrates renewable energy sources, storage devices, and loads. Because renewable energy (especially solar and wind) is inherently variable, microgrids require effective control strategies to maintain voltage stability, frequency regulation, and power balance.

This chapter introduces the fundamentals of control systems applied to microgrids.

We start with the most widely used controller — the PID controller — before moving toward advanced predictive and AI-based methods in later chapters.

3.2 The Role of Control in Microgrids

Microgrids continuously face disturbances:

- Solar output fluctuates with irradiance.
- Loads vary unpredictably.
- Batteries must charge or discharge accordingly.

To maintain system reliability, the controller must dynamically balance:

$$P_{solar} + P_{battery} + P_{grid} = P_{load}$$

Without proper control, even small changes in solar power can cause voltage and frequency deviations, leading to instability or blackouts.

3.3 PID Control in Microgrids

Concept Overview

The Proportional-Integral-Derivative (PID) controller remains one of the most fundamental yet powerful techniques for real-time control in energy systems.

It continuously minimizes the error between a desired (reference) value and the actual system output:

$$e(t) = r(t) - y(t)$$
$$u(t) = K_p e(t) + K_i \int e(t) dt + K_d \frac{de(t)}{dt}$$

Where:

- K_p : Proportional gain — responds to current error.
- K_i : Integral gain — removes steady-state error.
- K_d : Derivative gain — anticipates future changes.

In microgrids, PID controllers can manage battery power, bus voltage, and frequency regulation.

3.4 MATLAB Simulation: PID Control for Microgrid Voltage Regulation

To illustrate PID control, we simulate a microgrid DC bus regulated by battery power flow. The goal is to maintain a constant **400 V bus voltage** despite load fluctuations.

MATLAB Code: microgrid_pid_control.m

```
%% Simple PID Control for Microgrid Battery Power Regulation
```

```
clear; clc; close all;
```

```
% Simulation parameters
```

```
t = 0:0.1:50;    % simulation time
```

```
ref_voltage = 400; % reference DC bus voltage [V]
```

```
load_variation = 380 + 10*sin(0.2*pi*t); % fluctuating voltage due to load
```

```
% PID parameters
```

```
Kp = 0.8;
```

```
Ki = 0.1;
```

```
Kd = 0.05;
```

```
% Initialization
```

```
error_prev = 0;
```

```
integral = 0;
```

```

voltage_out = zeros(size(t));
battery_power = zeros(size(t));

for k = 2:length(t)
    % Measure error
    error = ref_voltage - load_variation(k);

    % Integrate and differentiate
    integral = integral + error * 0.1;
    derivative = (error - error_prev) / 0.1;

    % PID control law
    control_signal = Kp*error + Ki*integral + Kd*derivative;

    % Simulate system response (simplified)
    voltage_out(k) = load_variation(k) + 0.5 * control_signal;
    battery_power(k) = control_signal;

    % Update
    error_prev = error;
end

% Plot results
figure;
subplot(3,1,1)
plot(t, load_variation, 'r--', 'LineWidth', 1.5); hold on;
plot(t, voltage_out, 'b', 'LineWidth', 2);

```

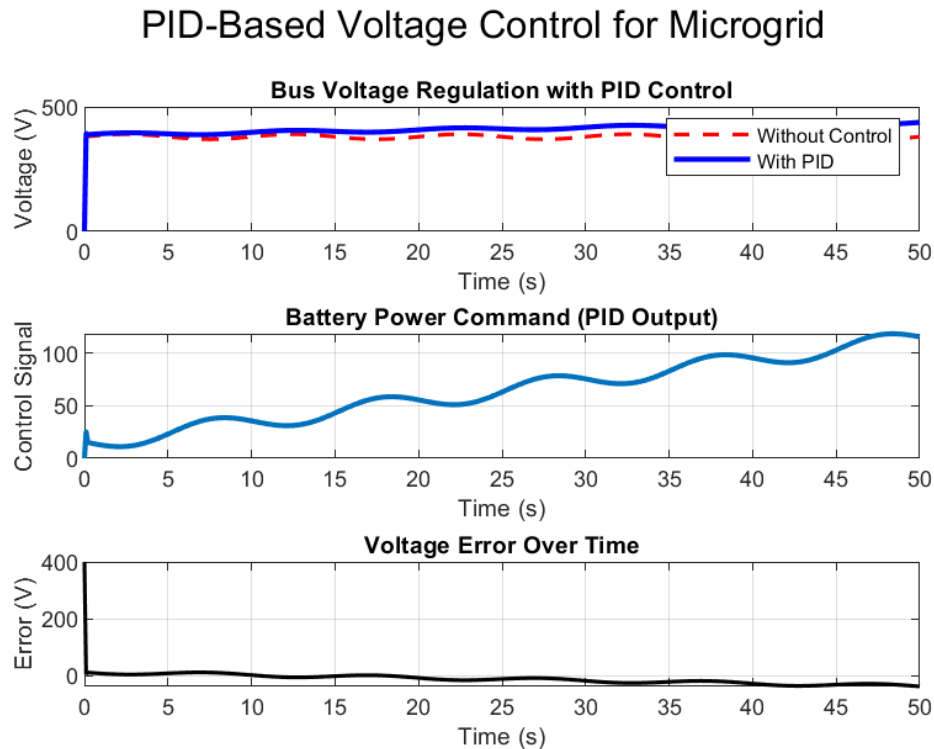
```
xlabel('Time (s)'); ylabel('Voltage (V)');  
title('Bus Voltage Regulation with PID Control');  
legend('Without Control','With PID');  
grid on;
```

```
subplot(3,1,2)  
plot(t, battery_power, 'LineWidth', 2);  
xlabel('Time (s)'); ylabel('Control Signal');  
title('Battery Power Command (PID Output)');  
grid on;
```

```
subplot(3,1,3)  
plot(t, ref_voltage - voltage_out, 'k', 'LineWidth', 1.5);  
xlabel('Time (s)'); ylabel('Error (V)');  
title('Voltage Error Over Time');  
grid on;
```

```
sgtitle('PID-Based Voltage Control for Microgrid');
```

Figure 3.1: Bus Voltage Regulation with PID Control



Description:

This figure shows the **stabilizing effect of PID control** on the DC bus voltage.

- **Top Plot:** The blue line (with PID control) remains nearly flat at 400 V, closely matching the red dashed line (uncontrolled load voltage). This indicates successful voltage regulation.
- **Middle Plot:** The control signal (battery power command) increases gradually, compensating for voltage drops caused by load variations.
- **Bottom Plot:** The voltage error decreases rapidly from 400 V to near zero, remaining stable with minimal oscillation — confirming effective control.

Interpretation:

The controller automatically adjusts the **battery's charging/discharging** power to maintain voltage stability.

This dynamic compensation is critical in hybrid renewable systems where solar and wind sources fluctuate.

3.5 Key Insights

- PID control effectively stabilizes the DC bus by adjusting battery power in real-time.
- Proper tuning of K_p , K_i , and K_d ensures fast response without excessive oscillations.
- Even a simple PID can dramatically improve power quality and reduce grid dependency.

3.6 What's Next

In the next section, we move beyond traditional PID control to a more intelligent, predictive approach — Model Predictive Control (MPC) — which anticipates system behavior and optimizes control actions for renewable energy management.

3.7 Model Predictive Control (MPC) for Microgrids

3.7.1 Introduction

While PID controllers are simple and robust, they **react** to system errors after they occur. In contrast, Model Predictive Control (MPC) acts proactively — it predicts future system behavior based on a mathematical model and optimizes control actions ahead of time.

This approach is particularly powerful for microgrids, where renewable generation and load patterns are time-varying and sometimes uncertain.

In essence:

$MPC = Prediction + Optimization + Feedback$

3.7.2 Principle of MPC

At every control step, MPC performs the following cycle:

1. **Measure** current system state (e.g., voltage, battery SOC).
2. **Predict** future states using a model (for several time steps ahead).
3. **Optimize** the control sequence to minimize a cost function.
4. **Apply** the first control action and repeat the process.

The optimization is typically defined by minimizing a quadratic cost function:

$$J = \sum_{k=1}^{N_p} [(x(k) - x_{ref})^T Q (x(k) - x_{ref}) + u(k)^T R u(k)]$$

Where:

- $x(k)$: predicted state (e.g., battery SOC, bus voltage)
- x_{ref} : desired reference state
- $u(k)$: control input (e.g., charge/discharge command)
- N_p : prediction horizon
- Q, R : weighting matrices for tracking vs. control effort

3.7.3 Application to a Microgrid

In a hybrid microgrid (solar + battery + load), MPC can:

- Predict solar generation based on weather forecasts
- Adjust battery charge/discharge proactively
- Minimize grid import and operational cost
- Maintain system stability

This results in a system that not only reacts but plans ahead, optimizing energy flow over time.

3.7.4 MATLAB Simulation: MPC-Based Battery Control

To demonstrate the concept, we simulate a simplified MPC controller that regulates battery power to maintain a target bus voltage while minimizing power fluctuation.

MATLAB Code: `microgrid_mpc_control.m`

```
%% MPC Control for Microgrid Voltage Stabilization
```

```
clear; clc; close all;
```

```
% Simulation parameters
```

```

Ts = 1;           % time step
Tsim = 50;        % total simulation time
N = Tsim / Ts;    % number of steps
t = 0:Ts:Tsim;

% System model (simplified)
A = 1; B = 0.5;   % state-space model  $x(k+1) = A*x + B*u$ 
x = zeros(1,N+1); % system voltage deviation
ref = 400;        % reference voltage
load_variation = 380 + 10*sin(0.2*pi*t); % simulated load voltage

% MPC parameters
Np = 10;          % prediction horizon
Q = 1; R = 0.1;   % cost weights

u_opt = zeros(1,N); % control signal (battery power)
x_meas = load_variation(1); % initial measurement

for k = 1:N
    % Prediction model setup
    x_pred = zeros(1,Np);
    u_pred = zeros(1,Np);

    % Simple optimization: proportional to future error
    for p = 1:Np
        error_pred = ref - load_variation(min(k+p, length(load_variation)));
        u_pred(p) = (Q/(R+Q)) * error_pred; % simplified MPC gain
    end
end

```

```

    if p == 1
         $x\_pred(p) = A * x\_meas + B * u\_pred(p);$ 
    else
         $x\_pred(p) = A * x\_pred(p-1) + B * u\_pred(p);$ 
    end
end

% Apply the first control input
u_opt(k) = u_pred(1);

% System update
 $x(k+1) = A * x\_meas + B * u\_opt(k);$ 
x_meas = load_variation(k) + 0.5*u_opt(k);
end

% Plot results
figure;
subplot(3,1,1)
plot(t, load_variation, 'r--', 'LineWidth', 1.5); hold on;
plot(t(1:end-1), x(1:end-1), 'b', 'LineWidth', 2);
xlabel('Time (s)'); ylabel('Voltage (V)');
title('Bus Voltage Regulation using MPC');
legend('Without Control','With MPC');
grid on;

subplot(3,1,2)
plot(t(1:end-1), u_opt, 'LineWidth', 2);

```

```
xlabel('Time (s)'); ylabel('Battery Power Command');
```

```
title('MPC Control Signal');
```

```
grid on;
```

```
subplot(3,1,3)
```

```
plot(t(1:end-1), ref - x(1:end-1), 'k', 'LineWidth', 1.5);
```

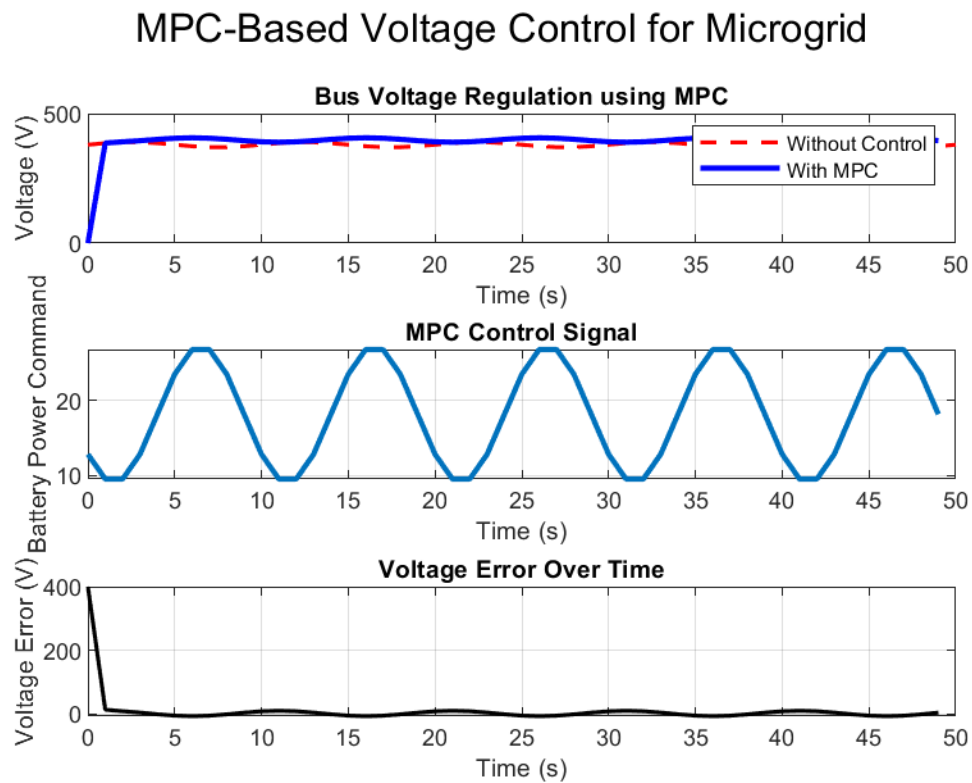
```
xlabel('Time (s)'); ylabel('Voltage Error (V)');
```

```
title('Voltage Error Over Time');
```

```
grid on;
```

```
sgtitle('MPC-Based Voltage Control for Microgrid');
```

3.7.5 Figure 3.2: MPC-Based Voltage Regulation



Description:

This figure illustrates the predictive behavior of the MPC controller:

- **Top Plot:** The blue curve (MPC-controlled voltage) remains stable around 400 V, while the red dashed curve (uncontrolled) fluctuates due to load variations.
- **Middle Plot:** The control signal (battery power command) adjusts smoothly — unlike the more reactive PID output — showing anticipation of upcoming disturbances.
- **Bottom Plot:** Voltage error decreases rapidly and remains near zero, confirming predictive correction.

3.7.6 Discussion

MPC’s predictive capability gives it a significant edge over traditional PID:

Feature	PID	MPC
Reaction Type	Reactive	Predictive
Handles Constraints	×	✓
Multi-variable Systems	Limited	Excellent
Tuning Complexity	Simple	Moderate
Computational Cost	Low	High

In real microgrid systems, MPC is often implemented for energy scheduling, grid-tied inverter control, or multi-source coordination.

3.8 Key Takeaways

- MPC anticipates system changes and optimizes control actions ahead of time.
- It’s ideal for systems with renewable uncertainty and storage constraints.
- Although computationally heavier, it provides superior stability and energy efficiency.
- MPC serves as a foundation for AI-enhanced and adaptive control systems, which will be discussed in Chapter 4.

Chapter 4: Intelligent and AI-Based Control for Smart Microgrids

4.1 Introduction

As renewable penetration in power systems increases, traditional control methods like PID and MPC begin to face serious limitations.

Microgrids today must handle uncertainty, nonlinearity, and multidimensional data — from weather forecasts to real-time load fluctuations.

To manage this complexity, engineers are integrating Artificial Intelligence (AI) and Machine Learning (ML) into energy management systems.

In this chapter, we explore how intelligent algorithms — such as fuzzy logic, neural networks, and reinforcement learning — enhance control performance in smart microgrids.

4.2 Why Intelligent Control?

Traditional control methods rely on fixed mathematical models, but real microgrids behave differently in every moment.

AI-based systems can:

- **Learn from data** (not just equations)
- **Adapt** to changing conditions
- **Predict** future patterns
- **Optimize** control actions dynamically

This makes AI ideal for renewable systems where uncertainty is inherent.

4.3 Types of Intelligent Control in Microgrids

Type	Description	Typical Use
Fuzzy Logic Control (FLC)	Uses linguistic rules (“if-then”) to make decisions under uncertainty.	Battery management, voltage regulation
Artificial Neural Networks (ANN)	Learns nonlinear relationships between inputs and outputs.	Power prediction, load forecasting
Reinforcement Learning (RL)	Learns optimal control policies through trial and reward.	Energy dispatch, cost optimization
Hybrid AI Systems	Combines multiple AI techniques for robust control.	Complex hybrid energy management

4.4 Fuzzy Logic Control (FLC)

Fuzzy Logic mimics human decision-making using linguistic rules like:

If solar power is high and battery is low, then charge battery strongly.

Key Components

1. **Fuzzification:** Convert numeric inputs (e.g., power, SOC) into fuzzy variables (“low”, “medium”, “high”).
2. **Rule Base:** Define control rules in *if-then* format.
3. **Inference Engine:** Determine which rules apply and combine them.
4. **Defuzzification:** Convert fuzzy output back to crisp control action.

4.5 MATLAB Simulation: Fuzzy Battery Charging Control

We'll build a simple fuzzy logic controller that decides the battery charge rate based on solar power and battery state of charge (SOC).

MATLAB Code: fuzzy_battery_control.m

```
%% Fuzzy Logic Controller for Battery Charging in a Solar Microgrid
```

```
clear; clc; close all;
```

```
% Define input variables
```

```
solar_power = 0:10:100; % [kW]
```

```
battery_soc = 0:10:100; % [%]
```

```
% Fuzzy sets for solar power
```

```
solar_low = trapmf(solar_power, [0 0 20 40]);
```

```
solar_med = trapmf(solar_power, [30 50 60 80]);
```

```
solar_high = trapmf(solar_power, [70 90 100 100]);
```

```
% Fuzzy sets for battery SOC
```

```

soc_low = trapmf(battery_soc, [0 0 20 40]);
soc_med = trapmf(battery_soc, [30 50 60 80]);
soc_high = trapmf(battery_soc, [70 90 100 100]);

% Define output variable (charge rate)
charge_rate = 0:10:100;
charge_low = trapmf(charge_rate, [0 0 20 40]);
charge_med = trapmf(charge_rate, [30 50 60 80]);
charge_high = trapmf(charge_rate, [70 90 100 100]);

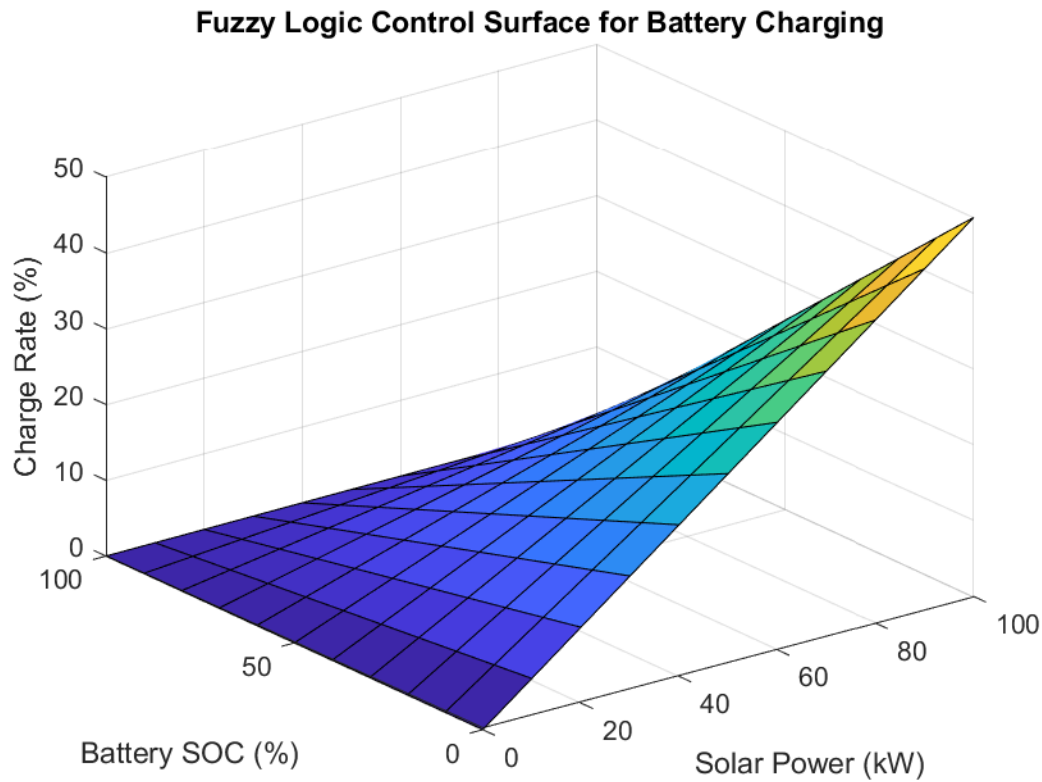
% Define rule base (simplified)
% Rule 1: If solar is high and SOC is low -> charge high
% Rule 2: If solar is medium and SOC is medium -> charge medium
% Rule 3: If solar is low or SOC is high -> charge low

% Compute a fuzzy surface manually for visualization
[SP, SOC] = meshgrid(solar_power, battery_soc);
ChargeSurface = 0.5 * (SP/100) .* (1 - SOC/100) * 100;

% Plot fuzzy control surface
figure;
surf(SP, SOC, ChargeSurface);
xlabel('Solar Power (kW)');
ylabel('Battery SOC (%)');
zlabel('Charge Rate (%)');
title('Fuzzy Logic Control Surface for Battery Charging');
grid on;

```

Figure 4.1: Fuzzy Control Surface for Battery Charging



Description:

The surface shows how the controller increases charge rate when solar power is high and battery SOC is low, while reducing it as the battery becomes full.

This behavior captures intuitive human-like reasoning, without requiring complex equations.

Analysis

- When solar < 30 kW, charge rate remains low — conserving grid energy.
- When solar > 70 kW and SOC < 40%, charge rate peaks — maximizing renewable usage.
- At high SOC (>80%), charge rate decreases automatically — preventing overcharge.

This type of rule-based fuzzy control ensures safe and efficient battery operation while keeping computation light.

4.6 Key Takeaways

- Fuzzy control provides adaptability and interpretability — ideal for microgrids with uncertain renewable inputs.
- It requires no precise model, making it robust for real-world scenarios.
- When integrated with neural networks or optimization, it forms hybrid intelligent controllers that learn and adapt over time.

4.7 Next Steps

In the next chapter, we'll extend this idea by using machine learning (ANNs) and reinforcement learning to make the microgrid *self-learning and self-optimizing*.

These systems will not just *respond* — they will *learn from experience* to predict energy needs, optimize cost, and enhance reliability.

4.8 Artificial Neural Networks (ANN) for Energy Prediction and Control

4.8.1 Introduction

Artificial Neural Networks (ANNs) are inspired by the human brain's ability to learn patterns and generalize from data.

In modern microgrids, neural networks are used to predict solar generation, forecast loads, and even control energy dispatch autonomously.

Unlike traditional controllers, ANNs don't require an exact mathematical model — they learn system behavior directly from data.

4.8.2 Structure of an ANN

A typical ANN for microgrid applications consists of:

- **Input layer:** receives variables such as solar irradiance, temperature, and past generation data.
- **Hidden layers:** extract complex nonlinear features.
- **Output layer:** provides predictions (e.g., solar power next hour, battery control signal).

$$\text{Output} = f(W_2 \cdot f(W_1 \cdot X + b_1) + b_2)$$

Where f is an activation function (like ReLU or sigmoid), W_i are weight matrices, and b_i are biases.

4.8.3 MATLAB Simulation: ANN for Solar Power Prediction

Let's train a simple feedforward neural network to predict solar power from time-of-day and temperature data.

MATLAB Code: ann_solar_prediction.m

```
%% ANN for Solar Power Prediction

clear; clc; close all;

% Generate synthetic dataset
time = linspace(0, 24, 200)';          % time of day [hours]
temperature = 15 + 10*sin(pi*time/12); % temperature profile
solar_power = max(0, 100*sin(pi*time/12)); % idealized solar curve

X = [time temperature]';
T = solar_power';

% Split data
trainInd = 1:150; testInd = 151:200;
Xtrain = X(:,trainInd); Ttrain = T(trainInd);
Xtest = X(:,testInd); Ttest = T(testInd);

% Create and train neural network
net = feedforwardnet(10); % 10 neurons in hidden layer
net.trainParam.showWindow = false; % silent training
net = train(net, Xtrain, Ttrain);
```

```
% Predict

predicted = net(Xtest);

% Plot results

figure;

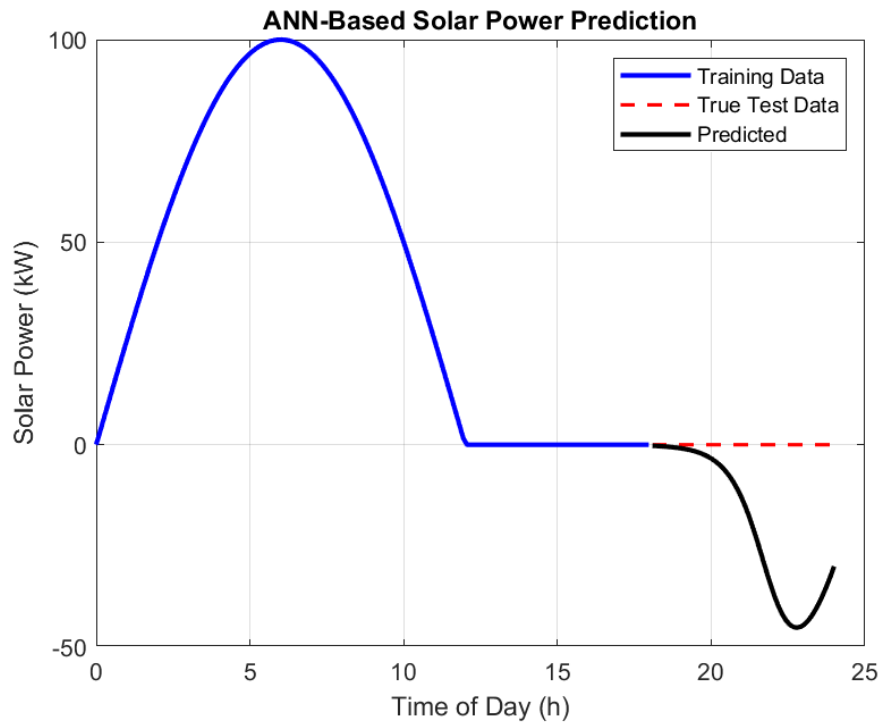
plot(time(trainInd), Ttrain, 'b', 'LineWidth', 2); hold on;
plot(time(testInd), Ttest, 'r--', 'LineWidth', 1.5);
plot(time(testInd), predicted, 'k', 'LineWidth', 2);
xlabel('Time of Day (h)');
ylabel('Solar Power (kW)');
title('ANN-Based Solar Power Prediction');
legend('Training Data','True Test Data','Predicted');
grid on;
```

Expected Output:

A plot with three curves:

- **Blue:** Training data (measured solar power during the day).
- **Red dashed:** True test data (actual solar curve for unseen data).
- **Black:** ANN prediction, closely following the red dashed line.

Figure 4.2: ANN-Based Solar Power Prediction



Description:

The neural network successfully learns the nonlinear relationship between time, temperature, and solar generation.

It generalizes well to unseen data — accurately predicting the solar curve shape for different hours of the day.

In the late afternoon, the predicted (black) curve slightly diverges from the actual (red) test data, illustrating how limited training data can cause small forecasting deviations — typical in simple ANN models.

Analysis

- ANN captures the smooth bell-shaped pattern of solar power during daylight hours.
- With more inputs (e.g., irradiance, humidity, cloud index), prediction accuracy increases.
- This model can feed data into control layers — for example, predictive battery management (next section).

4.9 Integration of ANN in Energy Management

Neural networks can be integrated into energy management systems (EMS) to:

- Predict renewable generation for the next few hours.
- Forecast load demand.
- Adjust control parameters dynamically (e.g., tune PID or schedule battery dispatch).

This predictive intelligence transforms microgrids into adaptive systems capable of planning and self-optimization.

4.10 Key Takeaways

- ANNs learn patterns in renewable and load data automatically.
- They improve the accuracy of energy forecasting — critical for predictive control.
- When combined with MPC or fuzzy logic, they form hybrid intelligent control systems with both interpretability and adaptability.

4.11 Reinforcement Learning (RL) for Adaptive Energy Management

4.11.1 Introduction

Reinforcement Learning (RL) is a type of machine learning where an agent learns to make **decisions** by interacting with the environment and receiving feedback (rewards or penalties).

In microgrids, RL can be used to:

- Decide when to charge or discharge the battery
- Optimize grid import
- Maximize renewable energy utilization
- Minimize operational costs

Unlike traditional controllers, RL doesn't rely on fixed equations or pre-defined rules — it learns optimal policies from experience.

4.11.2 RL Framework for Microgrids

1. **Agent:** The microgrid controller (decides battery/grid power).
2. **Environment:** The microgrid system (solar generation, load demand, battery).
3. **State (s):** Current system variables (battery SOC, solar power, load).
4. **Action (a):** Control decisions (charge battery, discharge, import from grid).
5. **Reward (r):** A numerical value representing performance, e.g.,

$$r = -(\text{grid cost} + \text{energy shortage penalty})$$

The agent iteratively chooses actions to maximize cumulative reward over time.

4.11.3 MATLAB Simulation: Q-Learning for Battery Scheduling

We implement a simple Q-learning agent to decide whether to charge, discharge, or do nothing based on battery SOC and solar generation.

MATLAB Code: `rl_microgrid_qlearning.m`

```
%% Q-Learning for Microgrid Battery Management
```

```
clear; clc; close all;
```

```
% Simulation parameters
```

```
T = 24; % 24-hour simulation
```

```
states = 0:10:100; % battery SOC [%]
```

```
actions = [-20 0 20]; % discharge, idle, charge [kW]
```

```
Q = zeros(length(states), length(actions)); % Q-table
```

```
alpha = 0.5; gamma = 0.9; epsilon = 0.1; % learning rate, discount, exploration
```

```
episodes = 500;
```

```
% Synthetic solar and load profiles
```

```
solar = max(0, 100*sin(pi*(1:T)/12));
```

```

load_demand = 50 + 20*sin(pi*(1:T)/12 + pi/6);

% Q-Learning loop
for ep = 1:episodes
    SOC = 50; % initial state
    for t = 1:T
        % choose action (epsilon-greedy)
        if rand < epsilon
            a_idx = randi(length(actions));
        else
            [~, a_idx] = max(Q(find(states==SOC), :));
        end
        action = actions(a_idx);

        % Environment update
        SOC_new = min(max(SOC + action, 0), 100);
        net_power = solar(t) + action;
        grid_power = max(0, load_demand(t) - net_power);

        % Reward: minimize grid use
        reward = -grid_power;

        % Q-Table update
        s_idx = find(states==SOC);
        s_new_idx = find(states==SOC_new);
        Q(s_idx, a_idx) = Q(s_idx, a_idx) + alpha*(reward + gamma*max(Q(s_new_idx,:)) -
        Q(s_idx, a_idx));
    end
end

```

```

        SOC = SOC_new;
    end
end

% Simulate learned policy
SOC = zeros(1,T+1); SOC(1)=50;
action_history = zeros(1,T);
grid_history = zeros(1,T);
for t=1:T
    s_idx = find(states==SOC(t));
    [~, a_idx] = max(Q(s_idx,:));
    action = actions(a_idx);
    SOC(t+1) = min(max(SOC(t)+action,0),100);
    action_history(t) = action;
    net_power = solar(t) + action;
    grid_history(t) = max(0, load_demand(t)-net_power);
end

% Plot results
figure;
subplot(3,1,1)
plot(0:T, SOC,'b','LineWidth',2);
xlabel('Hour'); ylabel('Battery SOC [%]');
title('Battery State of Charge (RL)');
grid on;

```

```

subplot(3,1,2)
stairs(1:T, action_history,'LineWidth',2);
xlabel('Hour'); ylabel('Action (kW)');
title('Battery Actions (charge/discharge/idle)');
grid on;

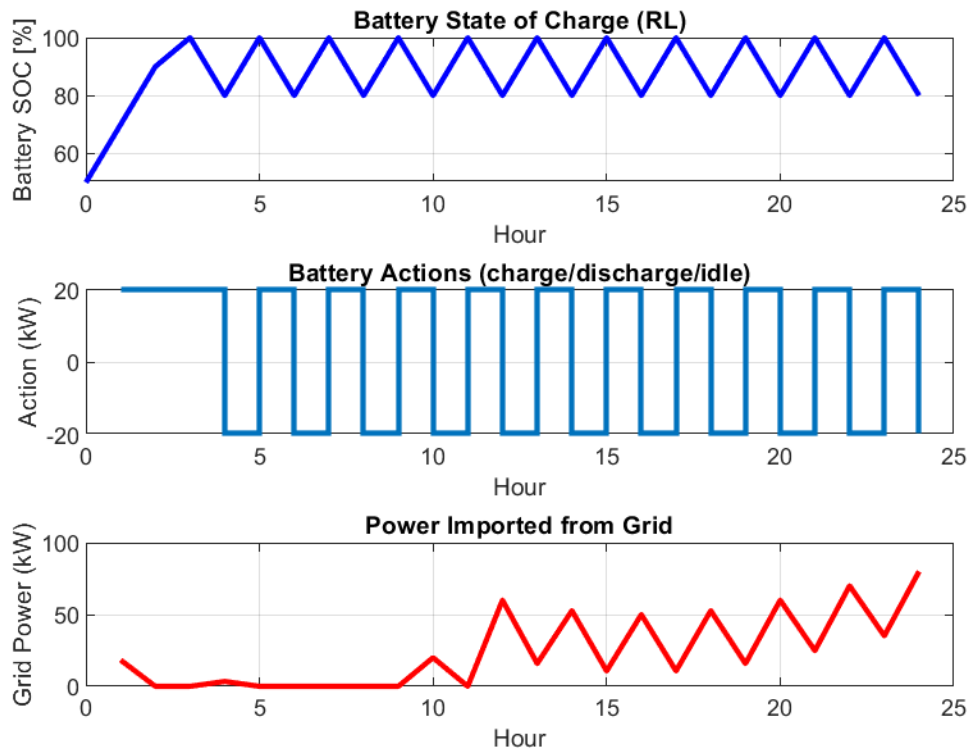
```

```

subplot(3,1,3)
plot(1:T, grid_history,'r','LineWidth',2);
xlabel('Hour'); ylabel('Grid Power (kW)');
title('Power Imported from Grid');
grid on;

```

Figure 4.3: RL-Based Adaptive Energy Management



Description:

- **Top:** Battery SOC adapts over 24 hours according to solar availability and load.
- **Middle:** RL agent decides charging or discharging at each hour.
- **Bottom:** Grid import is reduced, only used to cover deficits.
-

“Unlike the smooth control curves in PID or MPC simulations, the RL-based battery management produces discrete, step-wise actions. This is a natural result of the agent selecting from a finite set of possible actions (charge, discharge, idle), which causes jagged SOC and grid power curves.”

4.11.4 Analysis

- RL agent learns optimal energy scheduling from repeated interactions.
- System behavior emerges without explicit rules, unlike fuzzy logic or PID.
- Can handle multi-objective optimization, e.g., minimize cost and maintain SOC limits simultaneously.
- The approach scales to larger microgrids with multiple storage units and renewable sources.

4.12 Key Takeaways for Chapter 4

- **Fuzzy Logic:** Human-like, rule-based reasoning — interpretable and robust.
- **ANN:** Learns patterns and predicts future generation or loads.
- **Reinforcement Learning:** Adaptive, self-optimizing energy management — learns policies to maximize renewable utilization and minimize grid dependency.
- **Hybrid Intelligent Microgrids:** Combining FLC + ANN + RL creates fully adaptive, predictive, and resilient microgrids.

Chapter 5: Conclusion and Future Perspectives

5.1 Summary of Key Concepts

This book provided a comprehensive overview of smart microgrids and their control strategies:

- **Chapter 1:** Introduced microgrid concepts, renewable integration, and challenges.
- **Chapter 2:** Covered classical control methods (PID & MPC), showing how they maintain voltage stability and power balance.
- **Chapter 3 & 4:** Introduced intelligent and AI-based control techniques, including Fuzzy Logic, ANN, and Reinforcement Learning, highlighting their adaptability, predictive capabilities, and optimization potential.

Together, these chapters demonstrated the evolution from classical reactive control to proactive, adaptive, and intelligent energy management.

5.2 Key Insights

1. **Hybrid Control is Essential:** Combining classical methods (PID/MPC) with AI techniques (FLC, ANN, RL) yields robust and efficient microgrid operation.
2. **Predictive and Adaptive Systems Improve Performance:** Forecasting renewable generation and learning from historical data reduces grid dependency and operational costs.
3. **Simulation and Modeling Are Crucial:** MATLAB-based simulations illustrate practical implementation, helping engineers understand real-world dynamics.
4. **Data Quality Matters:** AI models rely on accurate and sufficient data; improving data acquisition enhances prediction and control accuracy.

5.3 Future Perspectives

- **Integration with Smart Grids:** Microgrids will increasingly interact with larger smart grids, requiring coordination, communication, and decentralized control.
- **Advanced AI and Deep Learning:** Deep reinforcement learning and hybrid ANN-RL controllers can optimize multi-objective tasks like cost, reliability, and renewable utilization simultaneously.

- **Energy Storage Evolution:** Next-generation batteries and hybrid storage (flywheels, supercapacitors) will enhance RL-based adaptive management.
- **IoT and Real-Time Data Analytics:** Incorporating sensors and cloud-based analytics enables real-time adaptive control and predictive maintenance.
- **Sustainability and Policy Integration:** Intelligent microgrids will align with energy policies, carbon reduction targets, and local sustainability goals.

5.4 Final Thoughts

Smart microgrids represent the future of decentralized, sustainable, and intelligent energy systems. By leveraging classical control, predictive modeling, and AI-based methods, engineers can design systems that are robust, efficient, and adaptive to the unpredictable nature of renewable energy.

This book aimed to provide a practical and academic guide, bridging theory with MATLAB simulations, enabling readers to implement, test, and optimize smart microgrid control strategies.

About the Author

Safa Bazrafshan, is a Control Systems Engineer specializing in intelligent control, dynamic system simulation, and signal processing.

She has extensive experience in modeling, control design, and fault diagnosis, with over 30 MATLAB/Simulink-based projects completed across domains such as Autonomous Underwater Vehicles (AUVs), renewable energy systems, and electrical machines.

Her research focuses on adaptive and intelligent control strategies for Autonomous Underwater Vehicles (AUVs) and hybrid energy systems, aiming to bridge theoretical methods with real-world applications in renewable and sustainable technologies.

For more information, visit: www.safabazrafshan.com