MATLAB Circuit Visualization Guide

Digital Twin with Fault Location Highlighting

Overview

This document describes the MATLAB-based circuit visualization system for the DC Microgrid Fault Detection Platform. Unlike the previous node-graph approach, this system renders a proper electrical circuit schematic with bus bars, cables, generators, and loads. When a fault is injected, the exact location is visually highlighted on the schematic with animated indicators.

Key Features

Feature	Description
Proper Schematic	Renders electrical circuit with bus bars, cables, generators, loads using standard symbols
Fault Highlighting	Animated pulsing circles and lightning bolt symbols indicate exact fault location
Real-time Updates	Circuit state updates in real-time via MATLAB Engine API or file-based mode
Voltage Display	Each bus shows current voltage level, updating as conditions change
Component Status	Generators, loads, and buses show operational status (normal/fault)
Interactive Legend	Color-coded legend explains all visual elements

Files Provided

File	Purpose	Location
DC_Microgrid_Visualize.m	MATLAB script for rendering circuit schematic with fault highlighting	matlab/
matlab_visualizer.py	Python bridge to MATLAB visualization	src/adapters/
digital_twin_enhanced.py	Streamlit page with embedded circuit schematic	src/ui/pages/

MATLAB Script Usage

Basic Usage (No Fault):

```
% Load circuit data
circuit_data = load('circuit_state.mat');

% Visualize circuit
result = DC_Microgrid_Visualize(circuit_data.circuit);

% Result contains figure handle for updates
disp(result.figure);
```

With Fault Highlighting:

```
% Define fault
fault_data.active = true;
fault_data.location = '4';        % Bus ID or Line ID (e.g., 'L2')
fault_data.type = 'LINE_TO_LINE';
fault_data.severity = 0.8;

% Visualize with fault
result = DC_Microgrid_Visualize(circuit_data.circuit, fault_data);
```

Real-time Updates:

```
% Initial visualization
result = DC_Microgrid_Visualize(circuit_data.circuit);

% Later, when fault occurs:
new_fault.active = true;
new_fault.location = '2';
new_fault.type = 'ARC_FAULT';
new_fault.severity = 0.6;

% Update the display
result.update(new_fault);
```

Python Integration

Initialization:

```
from src.adapters.matlab_visualizer import MatlabCircuitVisualizer

# Create visualizer (auto-detects MATLAB availability)
viz = MatlabCircuitVisualizer()
```

Update Circuit:

```
# Update with current circuit state
viz.update_circuit(circuit_model, emulator)

# Highlight fault
viz.highlight_fault(
    location='4',              # Bus ID
    fault_type='LINE_TO_LINE',
    severity=0.8,
    distance=150.0             # meters from sensor
)

# Clear fault
viz.clear_fault()
```

Event Handler Integration:

```
from src.adapters.matlab_visualizer import VisualizationEventHandler

# Create handler
handler = VisualizationEventHandler(viz)

# Subscribe to events
bus.subscribe(FaultLocationEvent, handler.on_fault_location)
bus.subscribe(SystemTripEvent, handler.on_system_trip)
```

Circuit Data Format

MATLAB Input Structure:

```
circuit.buses(i).id       = 1;
circuit.buses(i).name     = 'Main PCC';
circuit.buses(i).type     = 'Slack';
circuit.buses(i).x        = 0.5;      % Layout coordinate
circuit.buses(i).y        = 0.85;
circuit.buses(i).voltage  = 400.0;    % Real-time voltage (V)

circuit.lines(i).id       = 1;
circuit.lines(i).from_bus = 6;
circuit.lines(i).to_bus   = 1;
circuit.lines(i).r_ohm    = 0.01;
circuit.lines(i).length_km = 0.05;

circuit.generators(i).id    = 1;
circuit.generators(i).bus_id = 6;
circuit.generators(i).p_mw  = 0.5;

circuit.loads(i).id      = 1;
circuit.loads(i).bus_id  = 4;
circuit.loads(i).p_mw    = 0.15;
```

Supported Fault Types

Fault Type	Symbol	Visual Effect
LINE_TO_LINE	Lightning Bolt	Red bus/cable, pulsing circles, flash symbol
LINE_TO_GROUND	Ground Arc	Ground symbol with arc, voltage drop display
ARC_FAULT	Zigzag Arc	Intermittent flashing, zigzag pattern
NOISE	Noise Wave	Wavy pattern overlay
DRIFT	Down Arrow	Gradual voltage decline indicator

Layout Recommendations

For optimal schematic visualization, use the following layout conventions:

```
RECOMMENDED LAYOUT:
====================

        [Grid Infeed] - Bus 6 (y=0.95)
              |
        [Main PCC] ---- Bus 1 (y=0.85)
           /    \
    [Solar]    [Battery]    (y=0.85, left/right)
      |            |
   [Load A]    [Load B]     (y=0.35)

COORDINATE GUIDELINES:
- y=0.9-1.0: Main DC bus / Grid connection
- y=0.7-0.9: Generation sources
- y=0.3-0.5: Load centers
- x spread: 0.1 to 0.9 for visual separation
```

Standalone Mode (No MATLAB Engine)

If the MATLAB Engine API is not available, the visualizer operates in standalone mode, exporting circuit state to files that can be manually loaded in MATLAB:

```
# Files generated in standalone mode:
/tmp/dc_microgrid_viz/circuit_state.mat   # MATLAB format
/tmp/dc_microgrid_viz/circuit_state.json  # JSON format

# In MATLAB, load and visualize:
cd /tmp/dc_microgrid_viz
data = load('circuit_state.mat');
DC_Microgrid_Visualize(data.circuit);
```

Installation

Requirements:

```
# Python dependencies (already in project)
scipy>=1.0.0
numpy>=1.20.0

# Optional: MATLAB Engine for Python
pip install matlabengine
```

MATLAB Setup:

```
1. Copy DC_Microgrid_Visualize.m to your MATLAB path or project's matlab/ directory

2. Ensure MATLAB can find the script:
   addpath('/path/to/project/matlab');

3. Test the visualization:
   circuit = load('test_circuit.mat');
   DC_Microgrid_Visualize(circuit);
```

Summary

The MATLAB circuit visualization system provides a professional electrical schematic display with real-time fault location highlighting. Unlike the previous node-graph approach, this renders a proper circuit diagram with standard electrical symbols. The system supports both direct MATLAB Engine integration for real-time updates and file-based mode for standalone operation. When a fault is injected, the exact component (bus or line) is visually highlighted with animated indicators, making it immediately clear where the fault occurred in the physical circuit.