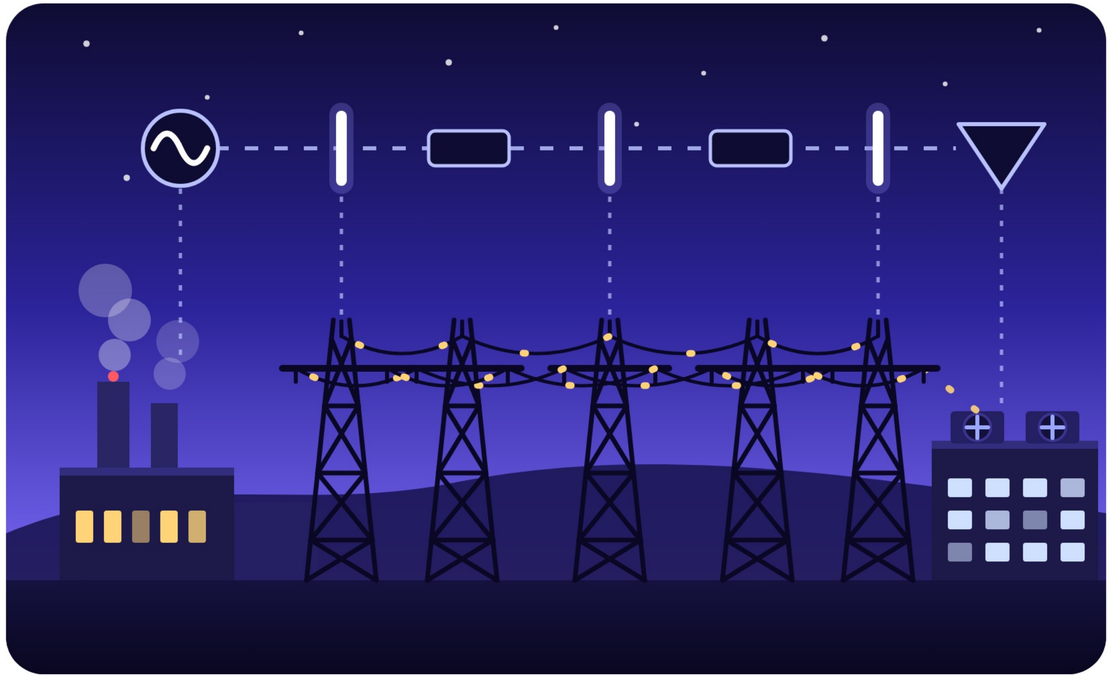
American Conference
modelica & fmi
Atlanta USA 2026

Implementation of OpenIPSL in Dyad





John Batteh · JuliaHub
Luigi Vanfretti · Rensselaer Polytechnic Institute

American Modelica & FMI Conference 2026 · October 14

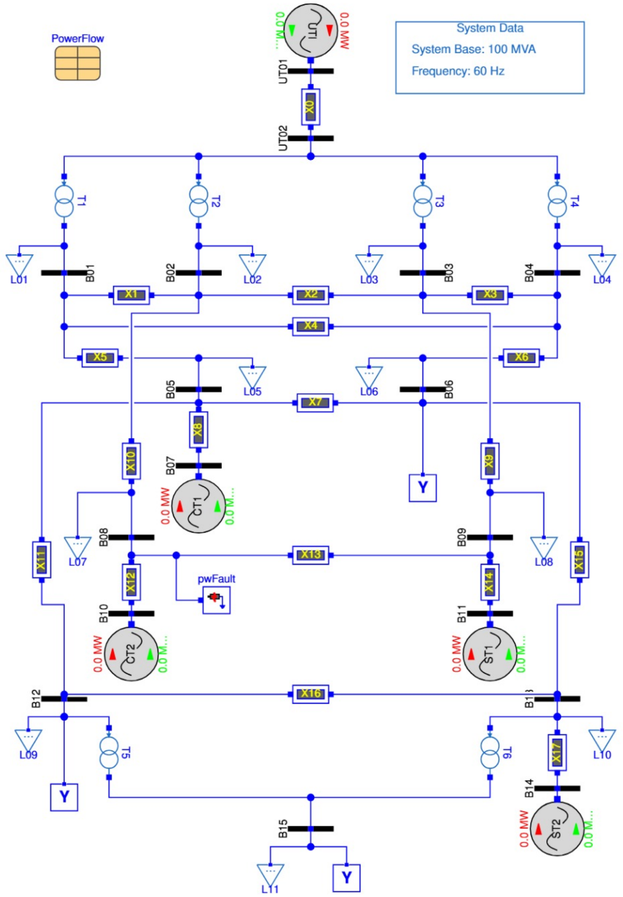
JuliaHub RPI

OPENIPSL

OpenIPSL



- **Open source** — power system library in Modelica
- **Lineage** — forked from iPSL after the iTesla project; maintained at RPI
- **Phasor-domain models** — machines, exciters, governors, loads, lines, transformers
- **Validated** — software-to-software against PSS/E
- **Multi-tool** — works in multiple Modelica tools
- **Applications** — power grids, microgrids, hydro systems and controls

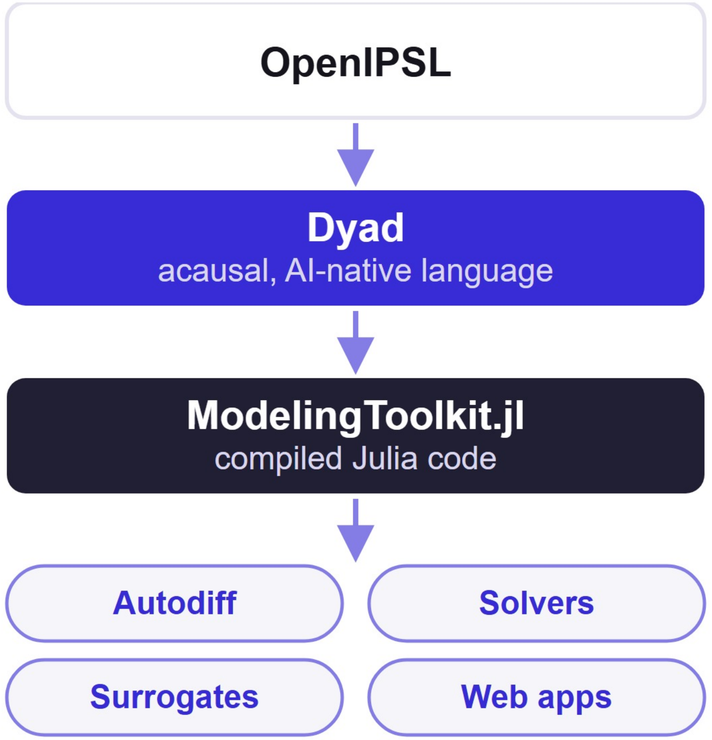


Campus microgrid model in OpenIPSL · Fachini et al. (2023), CC BY 4.0

MOTIVATION

Bringing a proven Modelica library into Dyad

- **OpenIPSL** — proven power system library in Modelica
- **Dyad** — acausal, equation-based, like Modelica
- **Julia/SciML** — autodiff, solvers, surrogates, deployment
- **AI-native** — built-in system-simulation agent
- **Goal** — make OpenIPSL broadly available to power system analysts and engineers in Dyad
- **Proof of concept** — data center power systems



OpenIPSL: de Castro et al. (2023), SoftwareX 21, 101277, DOI 10.1016/j.softx.2022.101277 · de Castro Fernandes et al. (2019), 13th Intl. Modelica Conf., DOI 10.3384/ecp19157617 · Fachini et al. (2023), 15th Intl. Modelica Conf., DOI 10.3384/ecp21181147 · Winkler (2018), American Modelica Conf., DOI 10.3384/ecp18154103 · Vanfretti & Laughman (2024), IEEE CCTA, DOI 10.1109/CCTA60707.2024.10666594

MOTIVATION

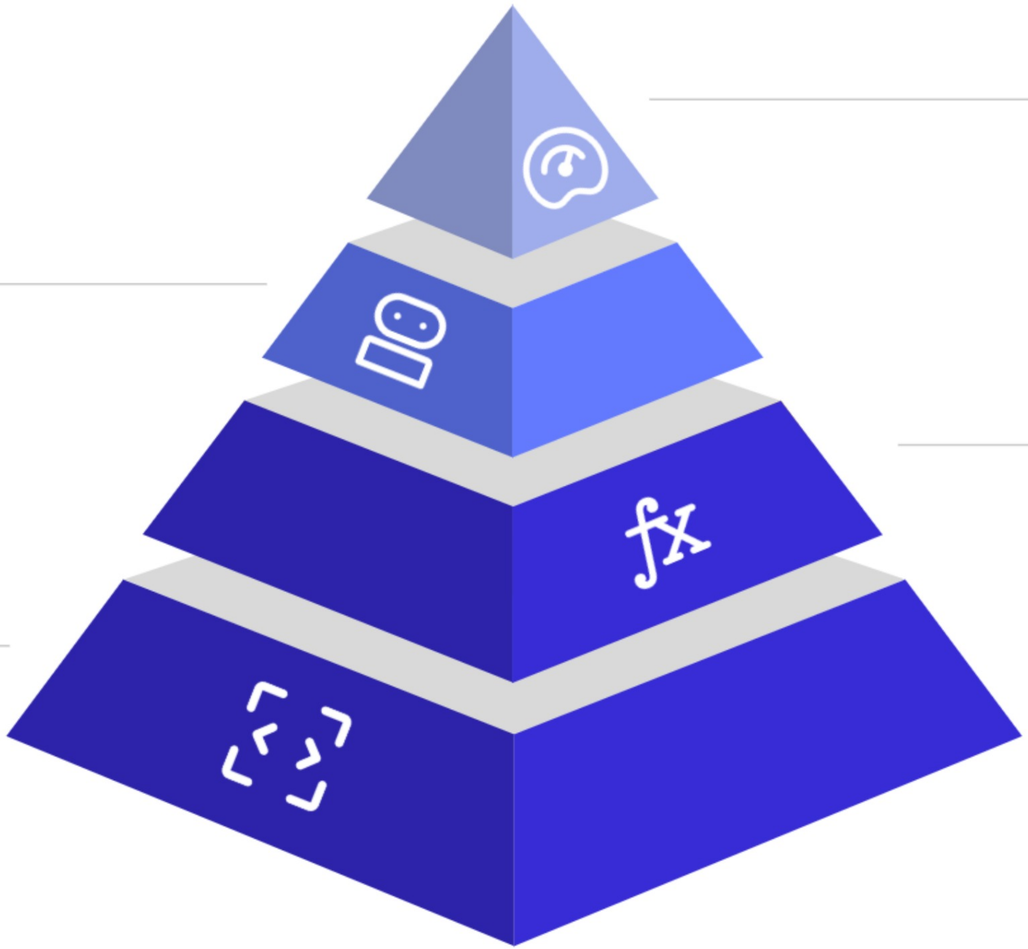
Dyad sits on a Julia and SciML foundation

Dyad Digital Twin

Leverages Dyad + SciML + Julia to build physics based **digital twins** of assets in operation for optimizing product performance, predictive maintenance.

Dyad

Agentic **design and analysis** that leverages an industry first physics compiler that is purpose built for AI engineering tasks.



SciML

Combines traditional physical/ mathematical modeling with data-driven AI delivering high quality results, faster. SciML is an Open source framework with hundreds of thousands of users

Julia Language

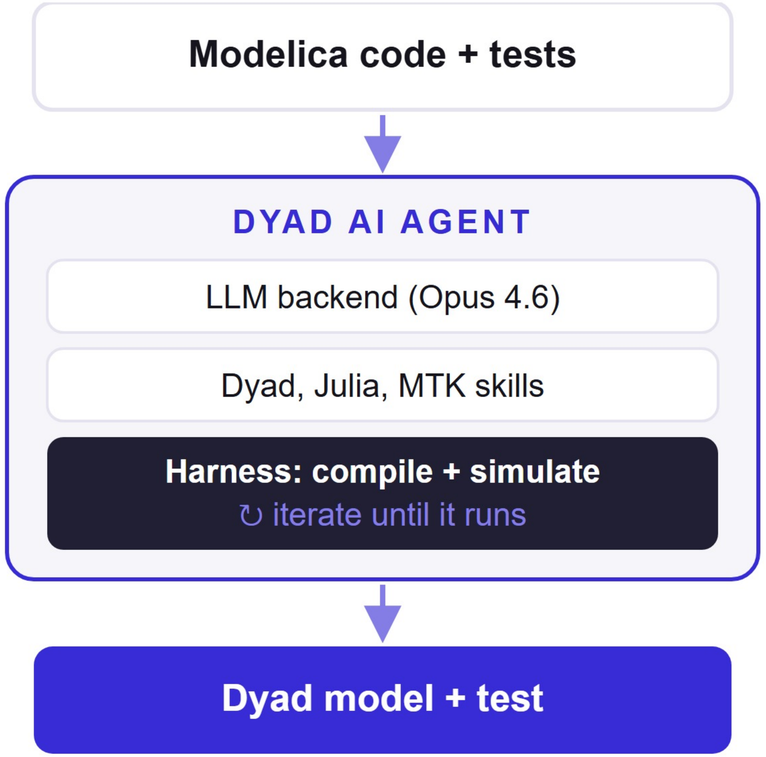
High performance scientific language and compiler optimized for engineering and AI tasks. Open source with over 2M innovation leaders.

OpenIPSL in Dyad gains the whole stack, from solvers to digital twins

APPROACH

Hundreds of components made hand translation impractical

- **Dyad AI agent** — skills for Dyad, Julia and MTK
- **Custom harness** — every model must compile and simulate
- **Human in the loop** — step-by-step review
- **Scope** — critical components, not the whole library



APPROACH

Five principles kept the translation faithful

FAITHFUL

=

Same code, organization, graphics and paradigms

SAME PHYSICS

Σ

Language differences must not change results

USE DYAD

✓

Native language features where behavior is identical

RENAME

→

Any model that differs gets a different name

TESTED

▲

Every model ships with at least one test case

Faithful reproduction of OpenIPSL, not a reinterpretation

TRANSLATION

It all starts with the connector

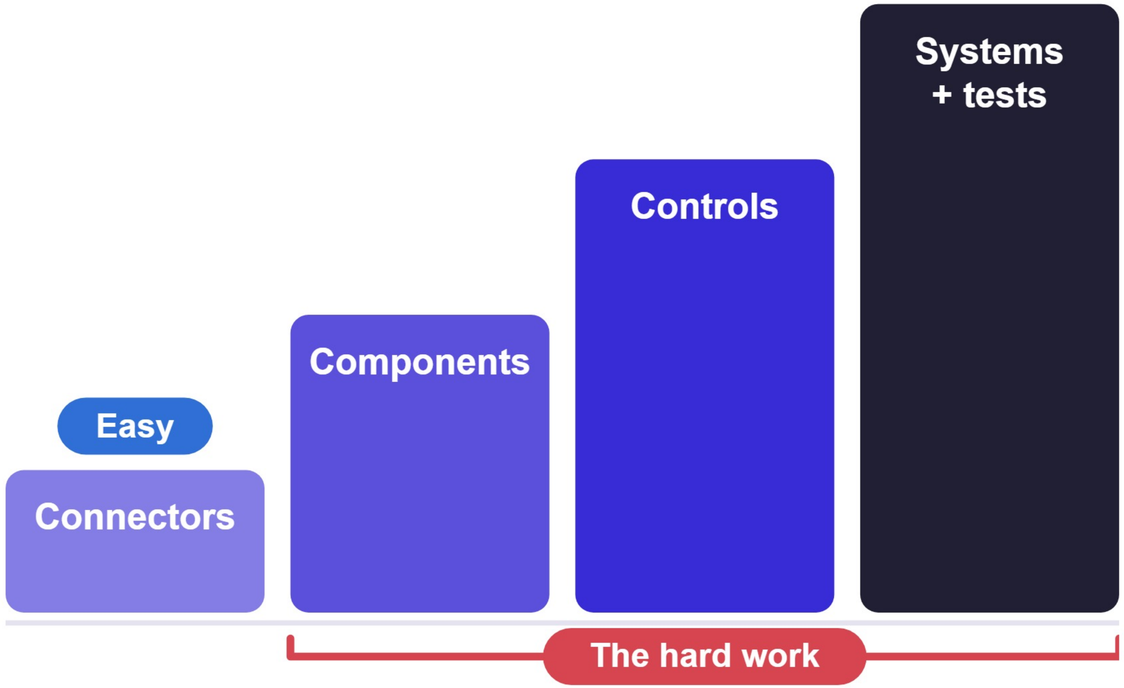
MODELICA

```
connector PwPin
  "Connector for electrical blocks treating voltage and current as complex variables"
  Types.PerUnit vr "Real part of the voltage";
  Types.PerUnit vi "Imaginary part of the voltage";
  flow Types.PerUnit ir(start=Modelica.Constants.eps) "Real part of the current";
  flow Types.PerUnit ii(start=Modelica.Constants.eps) "Imaginary part of the current";
  annotation ( ...);
end PwPin;
```

DYAD

```
"Connector for quasi-static single-phase electrical systems using phasor representation."
connector Pin
  "Real part of voltage phasor"
  potential vr::Voltage
  "Imaginary part of voltage phasor"
  potential vi::Voltage
  "Real part of current phasor"
  flow ir::Current
  "Imaginary part of current phasor"
  flow ii::Current
metadata {
  "Dyad": {
    "labels": [
      { "label": "instance_name", "x": 0, "y": 750, "layer": "diagram" }
    ],
    "icons": {
      "default": "dyad://OpenIPSLComponents/Pin.svg"
    },
    "path": { "stroke": "#0000ff" }
  }
}
end
```

Connectors are the easy part to translate. The hard work comes next.



JuliaHub

TRANSLATION

Each component follows the same loop

1 · PROVIDE

Modelica source and its OpenIPSL test placed in the workspace

→

2 · PLAN

Agent proposes a formulation; human reviews **before** code

→

3 · GENERATE

Dyad model plus the identical test; must compile and simulate

→

4 · VALIDATE

Compare to OpenModelica; iterate until results agree

→

5 · REPORT

Agent documents the model and its validation

```
Review OpenIPSL.Electrical.Loads.PSAT.PQ in the resources folder.  Propose an implementation in Dyad.
Validate using OpenIPSL.Tests.Loads.PSAT.LoadTestPQ.
```

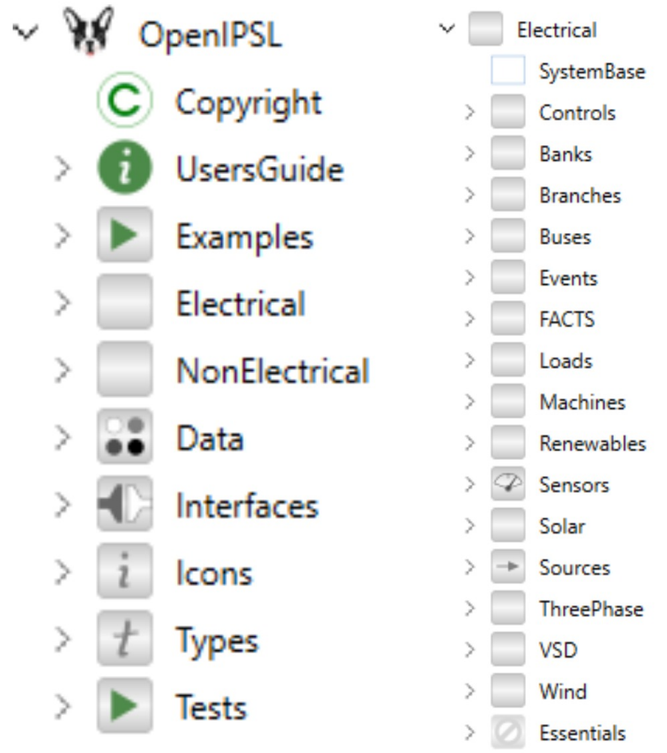
Sample prompt: point the agent at the Modelica source and its existing test

TRANSLATION

POC scope

70 components across **23** packages

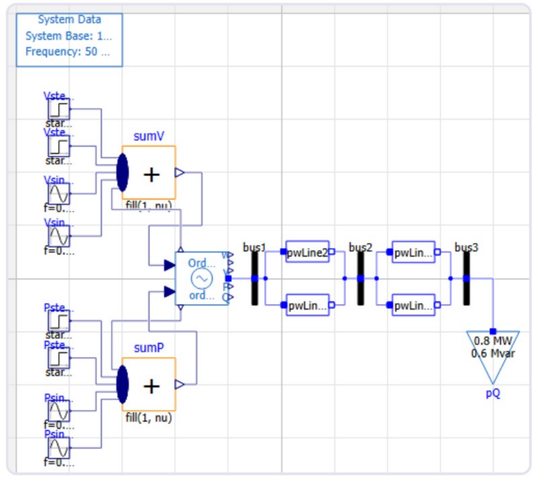
- **Machines** — PSAT Order3/4/6; PSS/E GENCLS, GENROU, GENROE
- **Controls** — IEEET1, ESST1A, ST5B exciters; GAST, IEEEG1 governors
- **Network** — 11 load models, lines, transformers, protection
- **Building blocks** — lags, lead-lags, limiters, gates



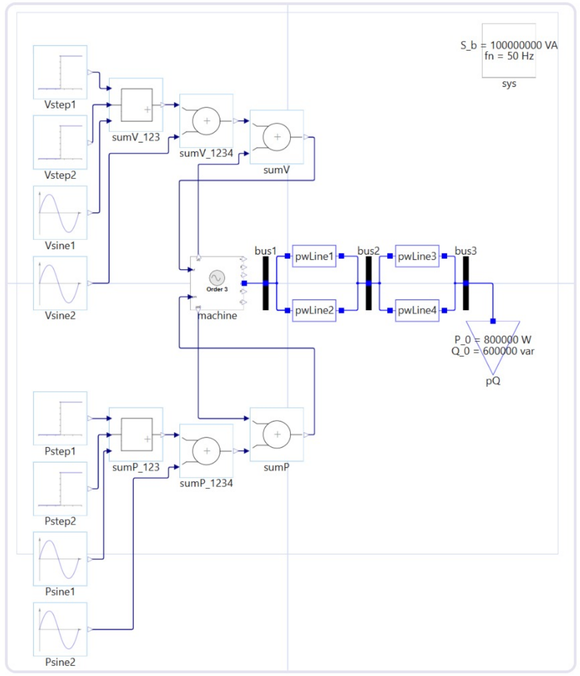
OpenIPSL 3.1 package structure

VALIDATION

LoadTestPQ: same test, matching results

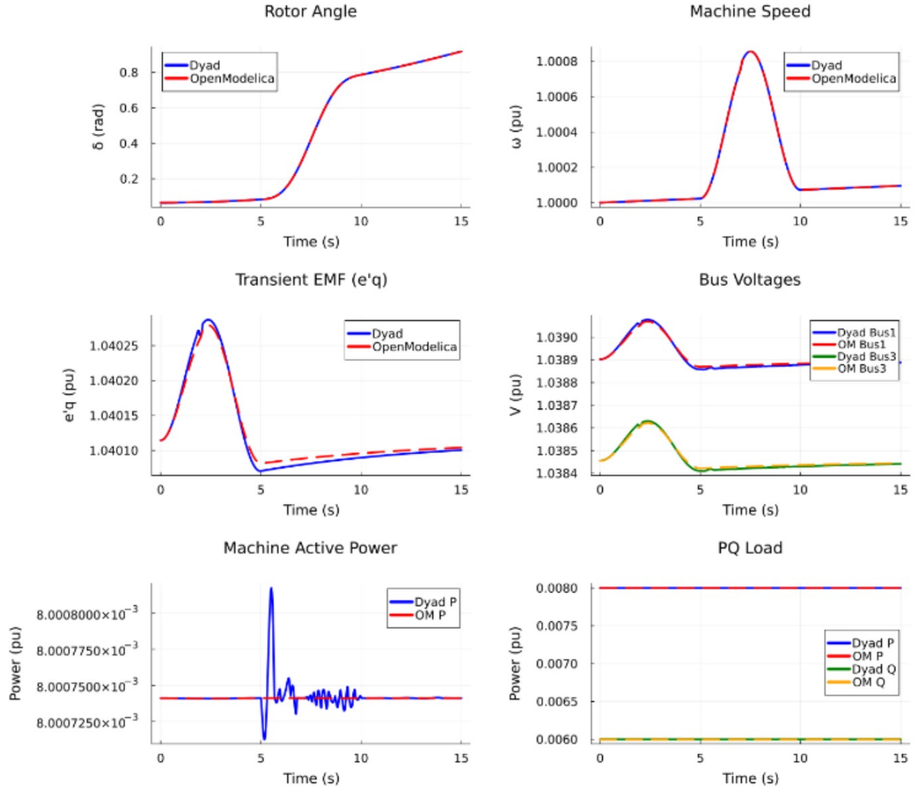


OpenModelica



Dyad

- **OpenModelica** — DASSL, tolerance 10^{-6}
- **Dyad** — auto-switching solver, tolerance 10^{-6}
- **Same output points** — no interpolation artifacts



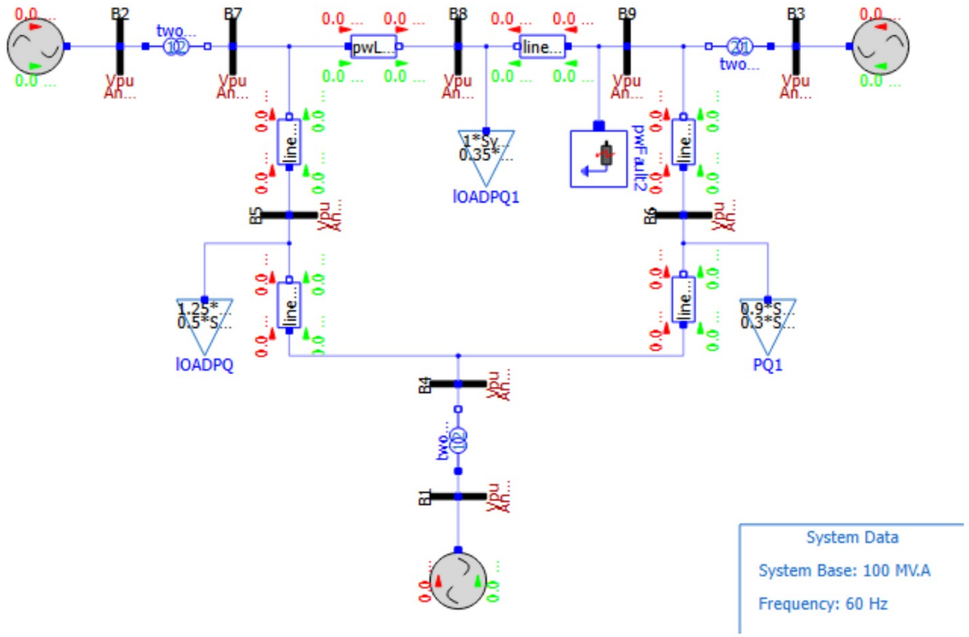
4.2×10^{-4} rad	1.3×10^{-6} pu	1.8×10^{-5} pu	7.6×10^{-8} pu
rotor angle	machine speed	e′q, bus voltages	machine power

Maximum differences, all at or below the solver tolerance

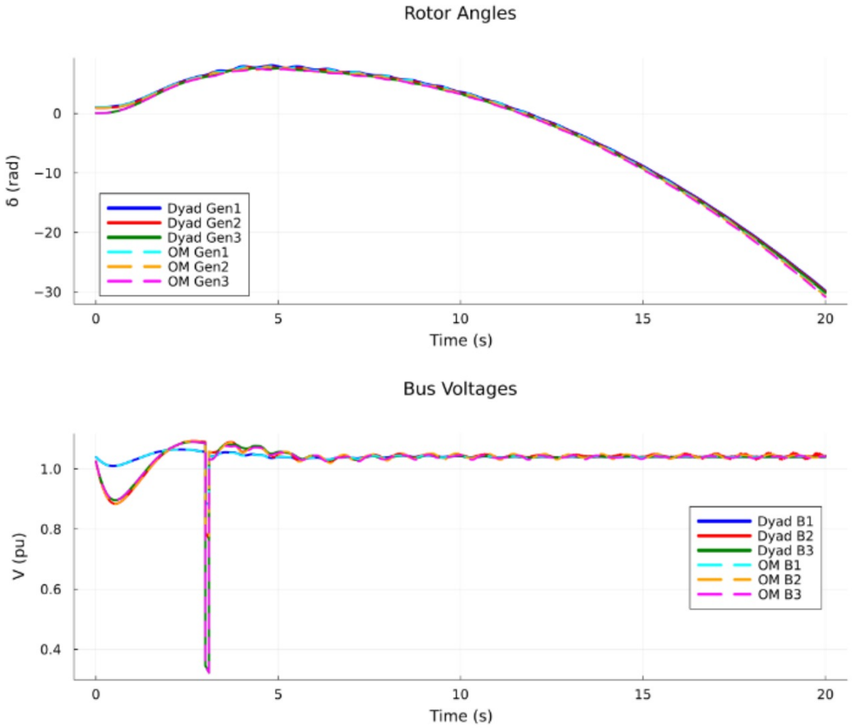
JuliaHub

VALIDATION

A marginally stable 9-bus case tracks OpenModelica



Example_3: 9 buses, 3 generators



- **Fault** — bus 9, 3.0 to 3.1 s
- **Damping** — under 2%: very sensitive

FOUND ALONG THE WAY

The example should start flat, so **Example_3 needs debugging**. Maintainers opened an issue and plan a fix before year end.

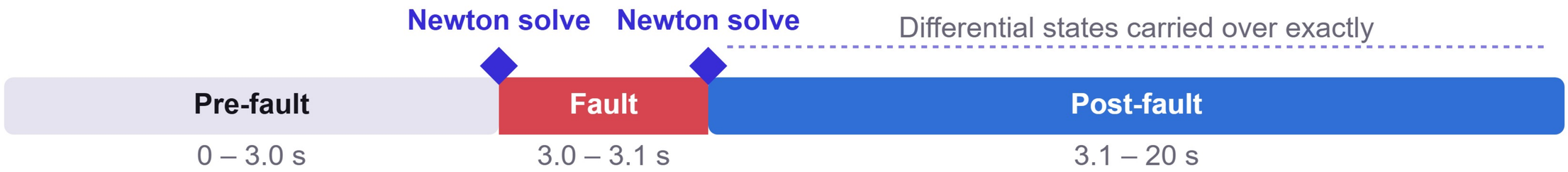
AI can translate, but only experts can see a non-obvious bug.

JuliaHub

OpenIPSL issue: **github.com/OpenIPSL/OpenIPSL/issues/402**

VALIDATION

Event handling workaround (for now)



- **Gap** — no automatic events for `if` in component equations yet
- **Workaround** — three simulation segments, consistent states at each switch
- **Next** — automatic event handling implemented, in testing

AI INSIGHTS

The agent found bugs in the original Modelica models

SMIB BASE CLASS

° ≠ rad

`angle_0` given in degrees; in radians it satisfies power flow exactly

GENROU / GENROE

+ −

Inconsistent equation and flux signs in the PSS/E generators

ST5B EXCITER

∅

Unconnected blocks and copy-pasted parameters: incomplete model

Multi-tool testing strengthens the whole ecosystem

AI INSIGHTS

Fixes flow back: the agent patched ST5B in Modelica

Ignoring the initialization differences, can you propose how you would backport the changes to Modelica?

Objective: Review the Dyad ST5B excitation system model, compare it against the original OpenIPSL Modelica source, and backport the bug fixes to Modelica.

Analysis performed:

Compared `dyad/Electrical/Controls/PSSE/ES/st5b_modinit.dyad` (corrected Dyad implementation) against `resources/PSSE/ST5B.mo` (original OpenIPSL Modelica) and the PSS®E reference block diagram (`resources/PSSE/ST5B.png`). Identified three bugs in the original Modelica:

1. `imLimitedLeadLag3` (2nd UEL block) used `T_UC1/T_UB1` instead of `T_UC2/T_UB2` — copy-paste error.
2. `imLimitedLeadLag5` (2nd OEL block) used `T_OC1/T_OB1` instead of `T_OC2/T_OB2` — same copy-paste error.
3. **UEL/OEL lead-lag outputs were dead-ended** — the main path (`imLimitedLeadLag2.y`) connected directly to the regulator gain (`K_r.u`), bypassing any select logic. The UEL and OEL lead-lag chains computed outputs that went nowhere.

Files created:

File	Description
`resources/PSSE/ST5B_corrected.mo`	Corrected ST5B model with all three fixes applied. Adds a `SelectLogic3` instance to route the three lead-lag path outputs to the regulator.
`resources/PSSE/SelectLogic3.mo`	New `OpenIPSL.NonElectrical.Logical.SelectLogic3` block implementing the three-way selection logic from the IEEE/PSS®E reference. OEL active › V3, UEL active › V2, otherwise › V1.

No Dyad files or generated code were modified.

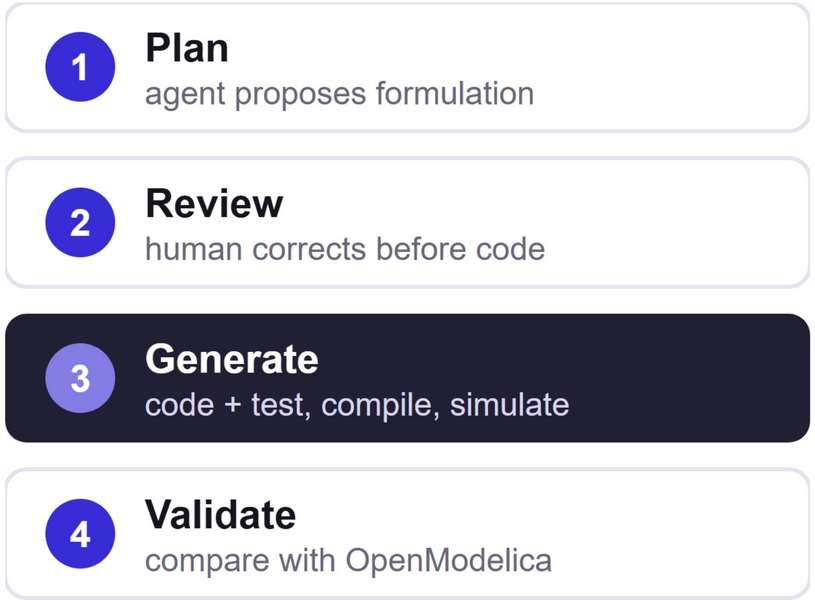
Prompt and agent summary: three ST5B bugs fixed, new SelectLogic3 block added

Both code bases stay in sync; automated pull requests are feasible

AI INSIGHTS

Plan first, then generate: lessons from the translation

- **Review formulations** — fixes before code save time and tokens
- **Harness guarantee** — no model left uncompilable for humans
- **Physics-aware** — agent checks conservation and sane values
- **Next time** — a reusable translation skill for consistency
- **Verdict** — far slower, perhaps infeasible, without the agent

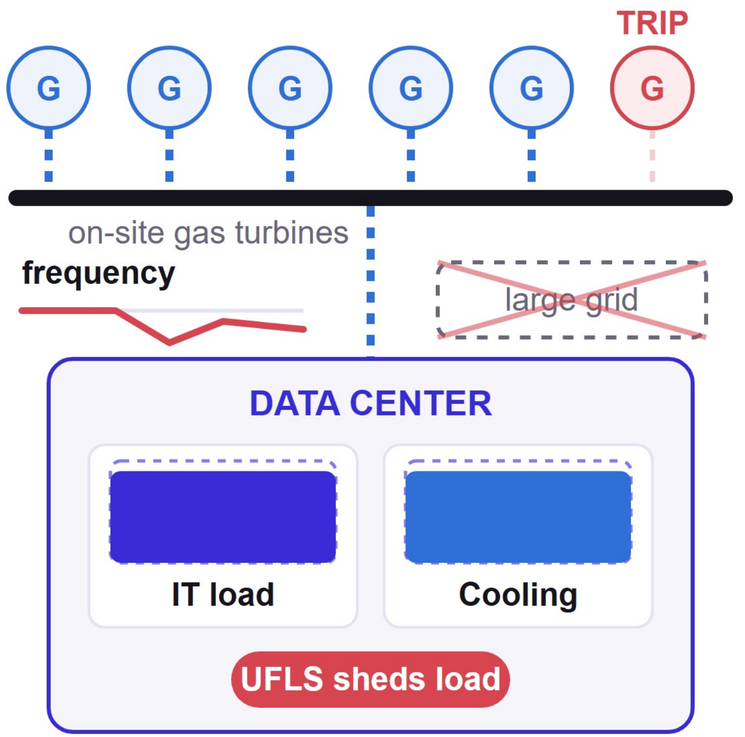


Fixes before code save time and tokens

DATA CENTER

Islanded data centers are acutely sensitive to generator trips

- **Load** — campuses routinely exceed 100 MW
- **Supply** — on-site gas turbines when the grid is limited
- **No grid support** — survivors absorb the full deficit
- **Risk** — under-frequency load shedding hits IT availability



DATA CENTER

New protection components in Dyad

- **ProtectionRelay** — delayed pickup, latching
- **ProtectionBreaker** — ANSI 27/32/50/51/59 modes
- **FrequencyRelay** — wrapping-free bus frequency
- **DataCenterLoad** — IT plus cooling, tiered shedding

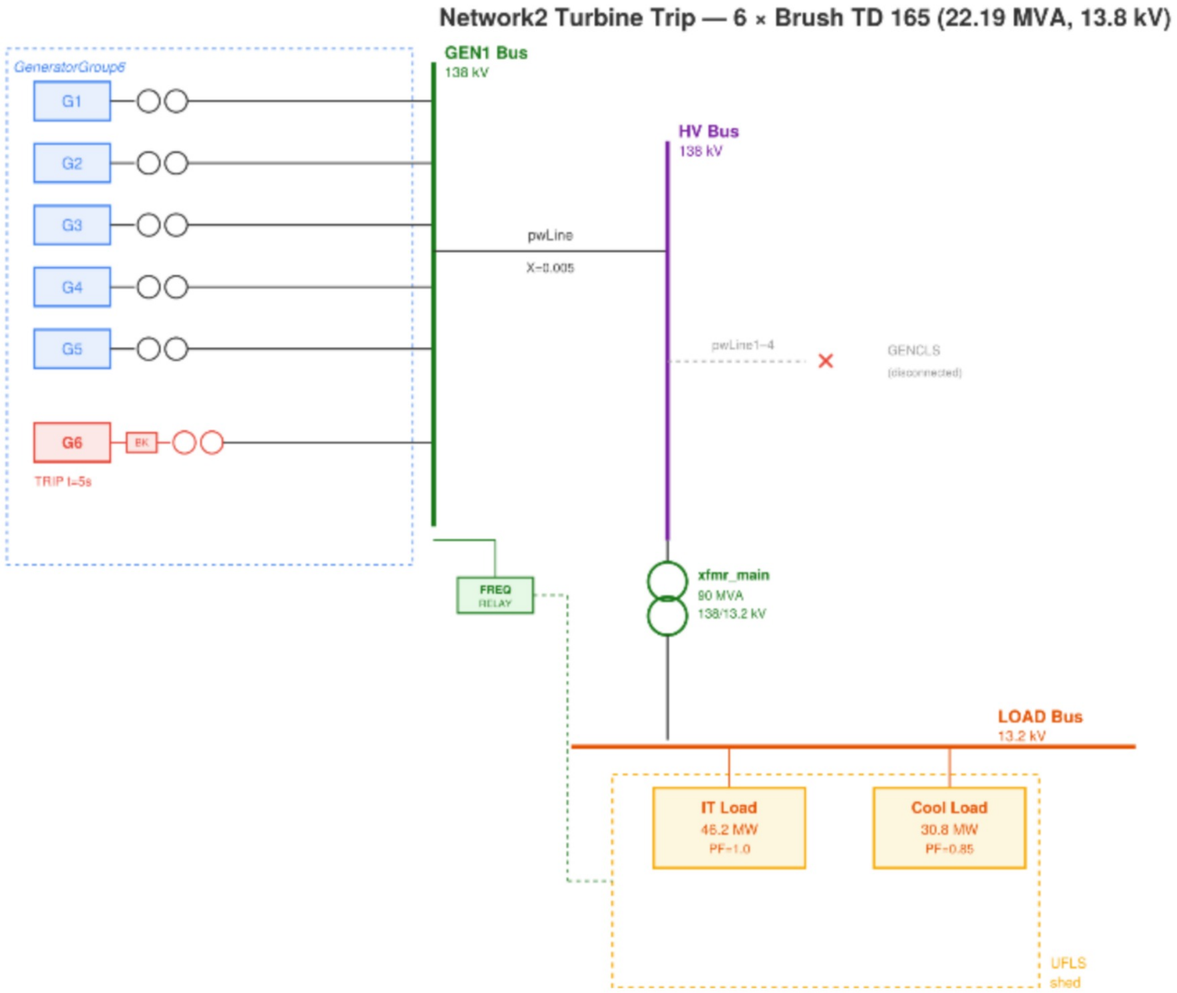
```
switch mode
  case ReversePower
    monitor = -(s.vr * s.ir + s.vi * s.ii)
  case OverCurrent
    monitor = sqrt(s.ir^2 + s.ii^2)
  case OverVoltage
    monitor = sqrt(s.vr^2 + s.vi^2)
  case UnderVoltage
    monitor = 2.0 * pickup - sqrt(s.vr^2 + s.vi^2)
end
```

Monitor signal selected by an enum and switch/case

DATA CENTER

Six GTs feed a data center via one transformer

- **Generators** — 6 GTs: GENROU, GAST, ST5B
- **Load** — 77 MW: constant-power IT, ZIP cooling
- **UFLS** — sheds IT and cooling fractions
- **ANSI 32** — reverse-power breaker on G6

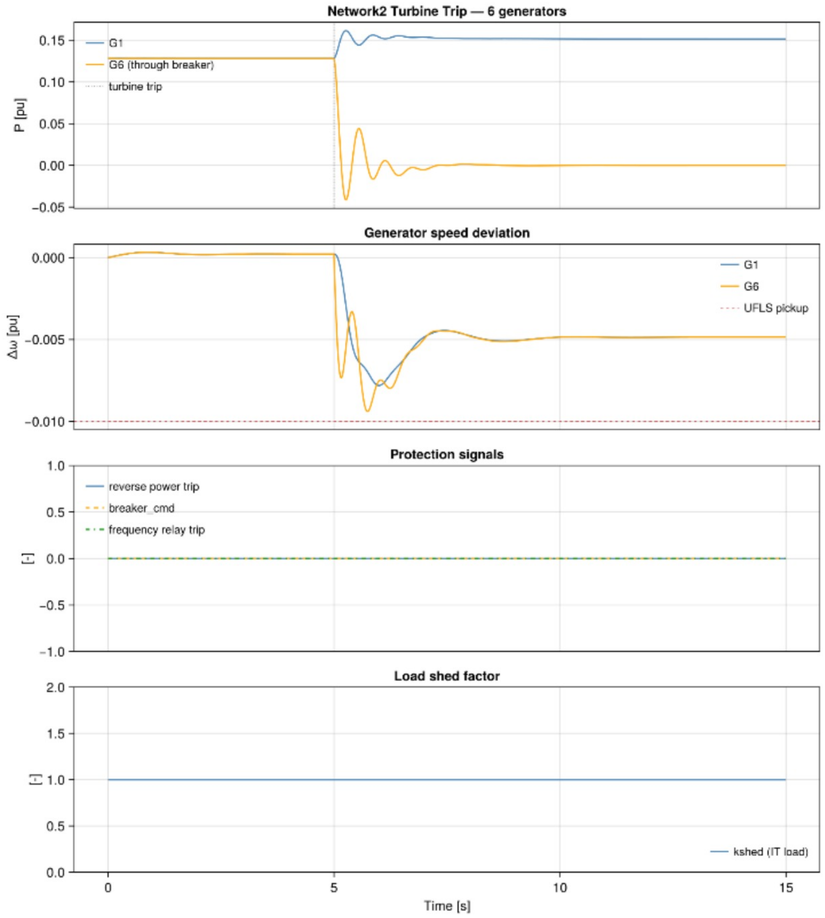


Turbine trip model schematic

RESULTS

With six units online, the trip rides through without shedding

- **Trip** — G6 lost at 5 s; 12.8 MW deficit
- **Nadir** — 59.53 Hz, above the 59.4 Hz UFLS pickup
- **Settles** — 59.71 Hz on 5% droop
- **Load shed** — none



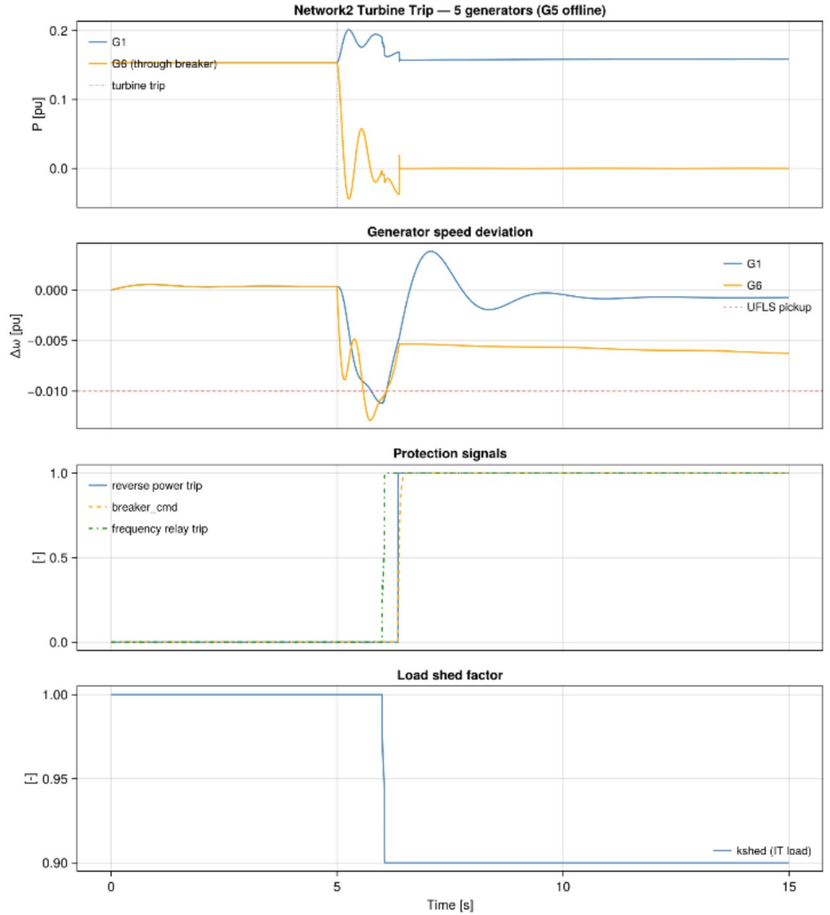
Six generators online; G6 trips at 5 s

JuliaHub

RESULTS

With five units online, the same trip sheds load

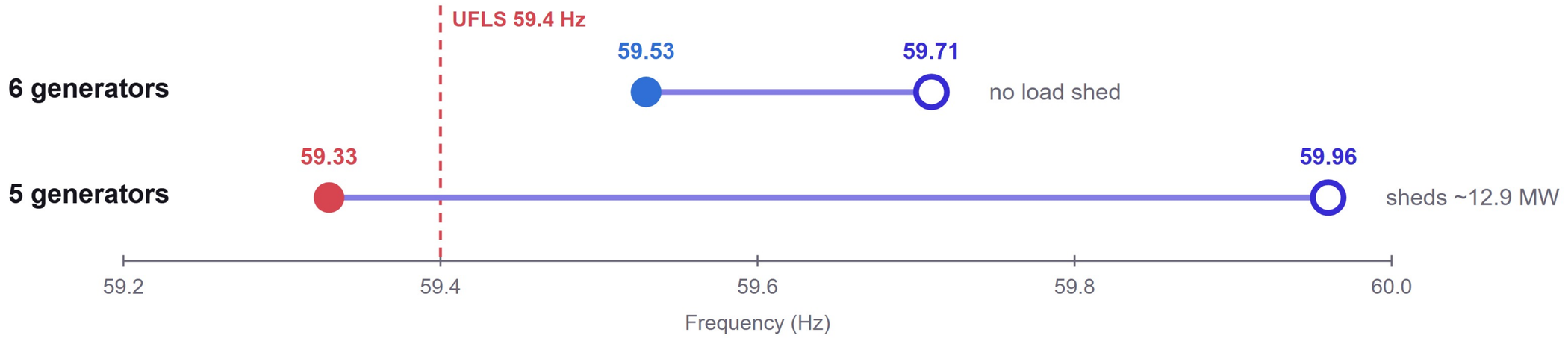
- **Nadir** — 59.33 Hz, breaches the UFLS pickup
- **Sequence** — frequency relay, then reverse-power breaker
- **Shed** — 10% of IT and cooling, about 12.9 MW
- **Settles** — 59.96 Hz after shedding



Five generators online; G6 trips at 5 s

RESULTS

Redundancy rides through; without it, load is shed



frequency nadir post-event steady state after G6 trips at 5 s

A stress test: no UPS ride-through, BESS or advanced frequency control

CONCLUSIONS

OpenIPSL now runs in Dyad

- **Translated** — 70 components, validated against OpenModelica
- **AI agent** — made translation and validation practical
- **Both communities** — bugs found, fixes back-ported to Modelica
- **Data centers** — new protection and load models, GT-trip study

DONE

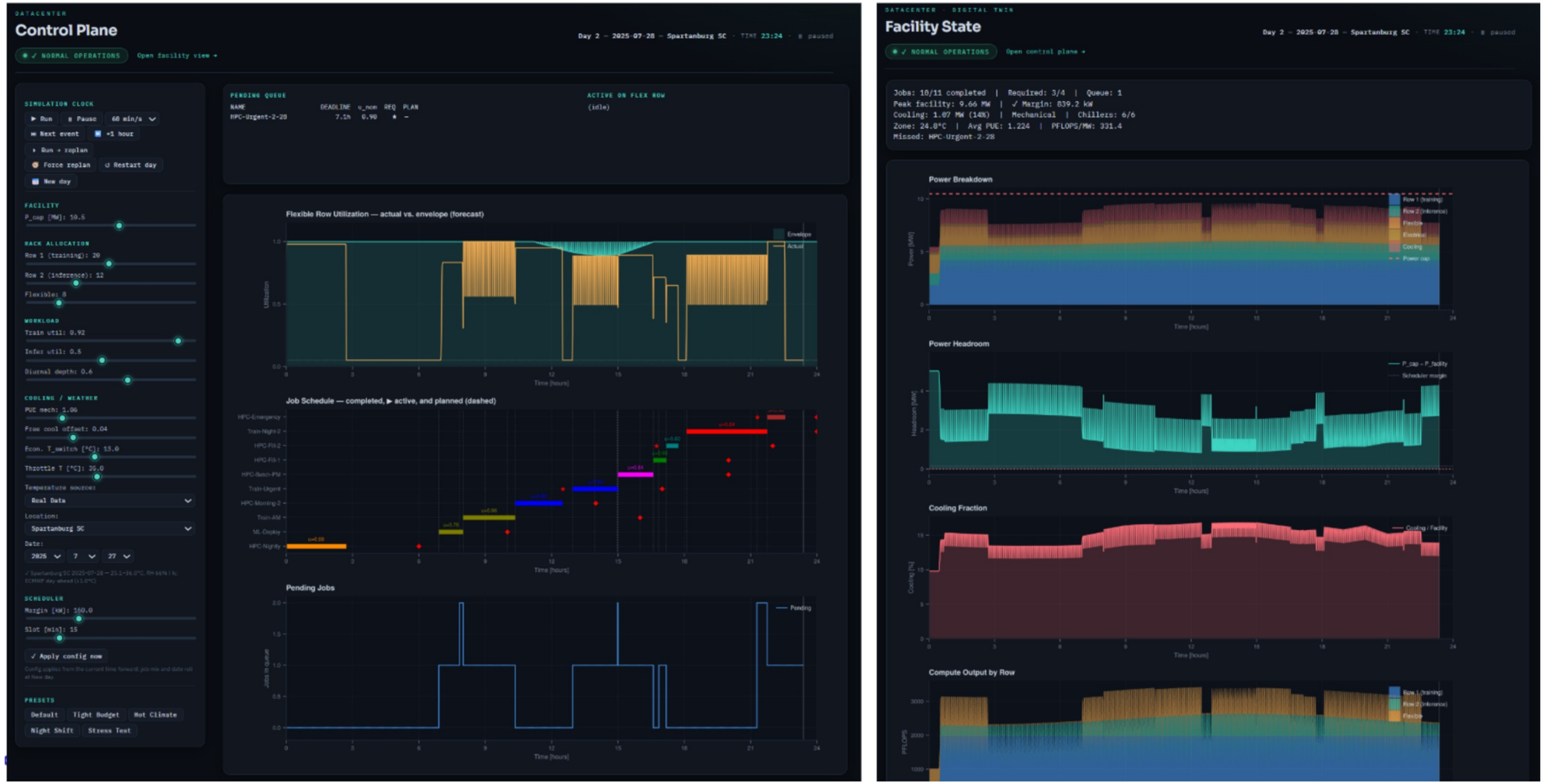
- ✓ 70 components translated
- ✓ Validated vs OpenModelica
- ✓ Bugs found and back-ported
- ✓ Data center GT-trip study

NEXT

- → Complete the library
- → UPS and BESS models
- → More data center controls

DIGITAL TWIN

Data center digital twin



Control plane (left) and facility state (right)

JuliaHub