Rensselaer
why not change the world?®



OpenIPSL model → FMI 2.0 Model Exchange → PowerFactory RMS

Deploying OpenIPSL Models into DIgSILENT PowerFactory via FMI Model Exchange

American Modelica & FMI Conference 2026, October 12–14 2026, Atlanta, GA, USA

Hao Chang and L. Vanfretti, ECSE Dept., RPI, Troy, NY, United States

E-mail: changh7@rpi.edu, vanfrl@rpi.edu | Web: https://alsetlab.github.io/

Code, FMUs, PowerFactory projects and the 45-page tutorial

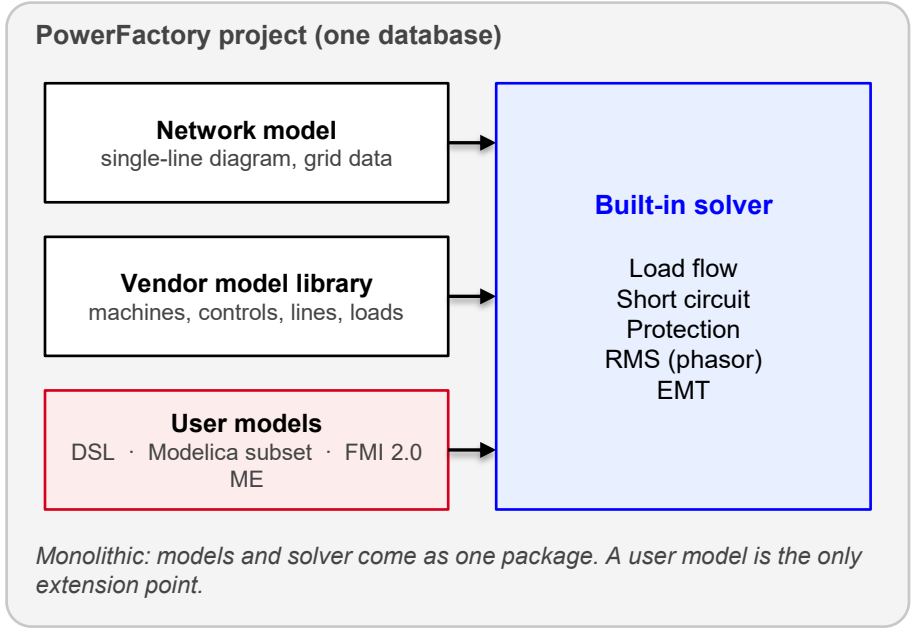
The Import Target: What PowerFactory Is

A commercial power-system simulation tool (DIgSILENT GmbH)

- **One project database, many studies:**
  - load flow, short circuit, protection
  - RMS (phasor-domain, our target) and EMT (waveform) dynamic simulation
- **Every dynamic study starts from a load flow:**
  - the built-in solver computes the operating point and initialises every dynamic model from it

Monolithic, like every power-system tool

- **Model library and solver ship as one package:** the solver is fixed; models come from the vendor library
- **User models are the only way in;** three ways matter here:
  - DSL (its own block language), a Modelica subset, FMI 2.0 Model Exchange



Monolithic: models and solver come as one package. A user model is the only extension point.

Every RMS or EMT study is initialised from a converged load flow — remember this for the initialisation handshake.

Why Bother Deploying Into PowerFactory?

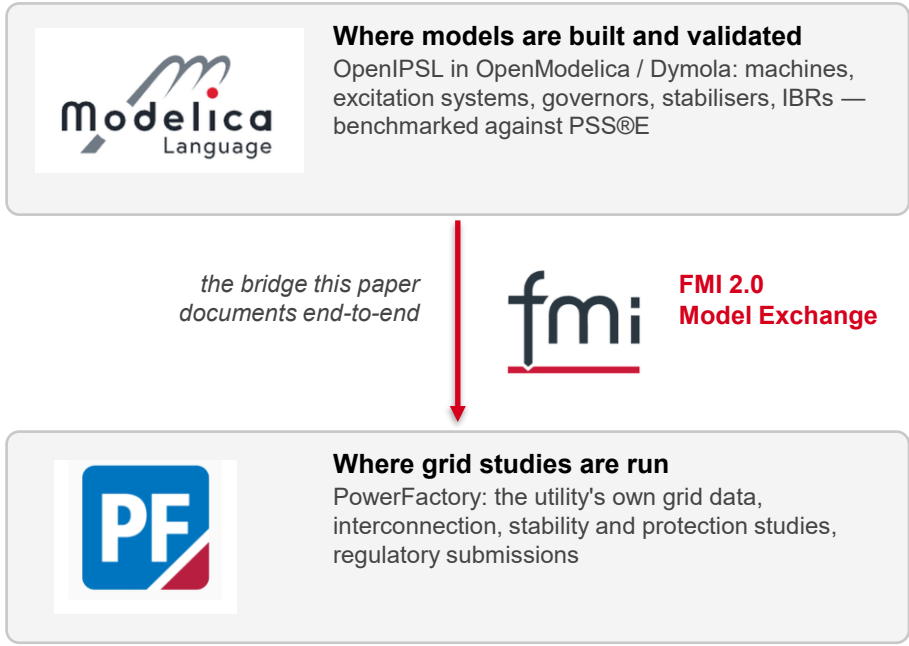
ALSETlab

It is where the grid studies actually happen

- **Standard tool of many utilities and system operators:**
  - across much of Latin America (Chile, Colombia, Ecuador, …) and Europe; little used in the U.S.
- **Often a regulatory deliverable:** several jurisdictions require the dynamic model in PowerFactory format
- **The grid models already live there:** nobody rebuilds a national grid in a research tool

What is missing there

- **Models the vendor library does not have:**
  - machine and control models we already implemented and validated in OpenIPSL for utility studies
- **So the question is the reverse of the usual one:** not “bring the grid to Modelica”, but “bring the validated Modelica model to the grid”



A validated open model that cannot reach PowerFactory cannot enter those workflows at all, whatever its technical merit.

Why FMI, and Not PowerFactory's Native Modelica?

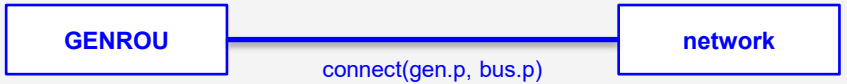
Its native Modelica is not the one you know

- **A documented subset:** no acausal connectors, no Modelica Standard Library in the published feature set — OpenIPSL cannot run there
- **Write once, run only there:** it breaks “implement once, run everywhere”
- **Continuous-time models only since 2025** (2022: discrete-time only)

FMI Model Exchange is the documented route

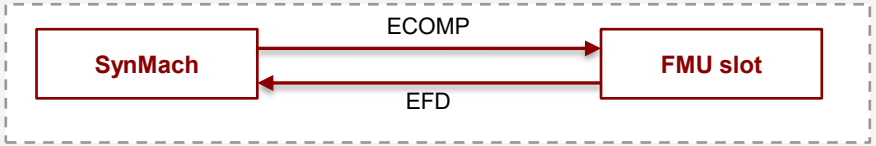
- **Tool-independent:** export from OpenModelica, import into PowerFactory 2022+
- **Model Exchange:** PowerFactory's own integrator advances the FMU's states
- **The price:** OpenIPSL is acausal, PowerFactory exchanges directed signals only — so we convert before export; **how much goes inside the FMU** is the design decision

OpenIPSL / Modelica — acausal



PwPin: vr, vi (potential) · flow ir, ii ⇒ sum of flows = network current balance

PowerFactory — directed ✗ **no documented direct route**



Composite Model Frame: directed signals only. SynMach = PowerFactory's native machine.

FMI 2.0 Model Exchange → Modelica Model Type

2022: FMI ME + discrete-time Modelica | 2025: continuous-time (“hybrid”) Modelica

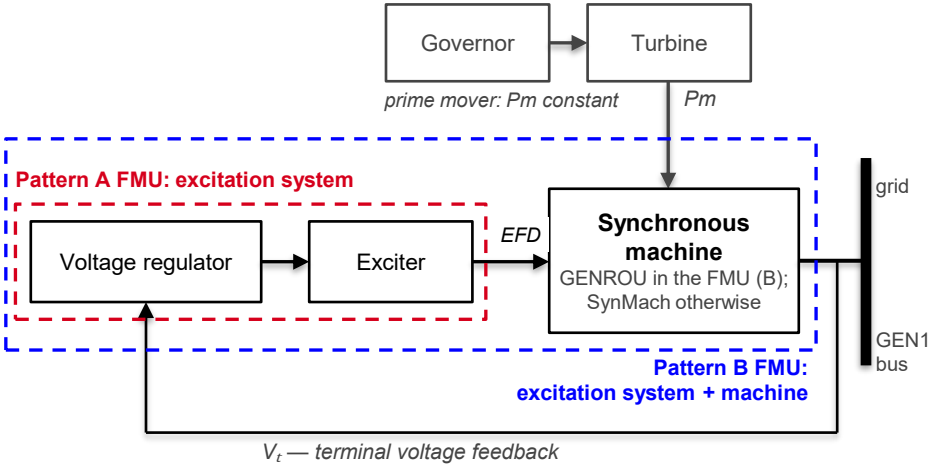
What Goes Inside the FMU: Anatomy of a Generator Unit

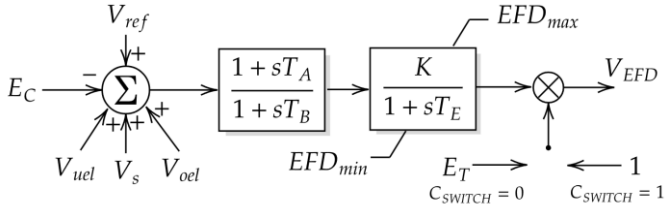
A generator unit is several subsystems

- **Prime mover:** turbine + governor set the mechanical power (held constant here)
- **Excitation system:** voltage regulator + exciter; sets the field voltage `EFD` to hold the terminal voltage
  - here the **SCRX**, a standard PSS®E-library model that every grid tool re-implements — a subsystem of the unit, not a stand-alone controller
- **Synchronous machine:** turns field and mechanical power into the current injected into the grid
  - here the PSS®E **GENROU** round-rotor model

Two variants of what we put in the FMU

- **Pattern A — excitation system only:** wired into the control loop of PowerFactory's own machine
- **Pattern B — full machine:** excitation system + machine; PowerFactory only sees the terminal





SCRX: a static excitation system — lead-lag regulator, gain K, output limiter.

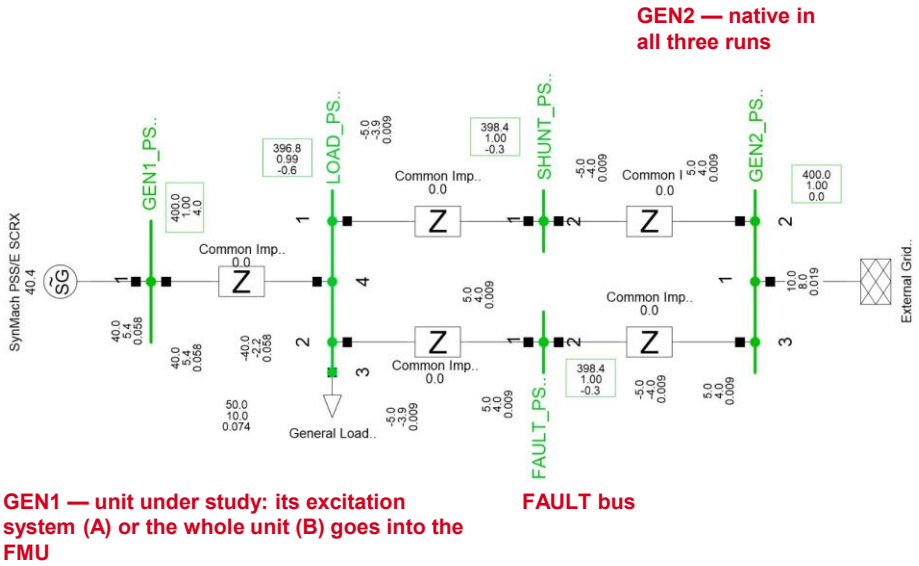
The Benchmark: One Grid, One Operating Point, One Fault

Where the grid comes from

- **Originally an OpenIPSL test system:** a small two-machine network, rebuilt one-to-one in PowerFactory so both tools simulate the same grid
- **Elements:** GEN1 (the unit under study), GEN2 (always native), a load, a shunt, a FAULT bus, an external-grid equivalent

Operating point and disturbance

- **One load flow sets the operating point:** GEN1 at rated P, terminal voltage 1.00 p.u. → `EFD` ≈ 1.434 p.u.
  - a no-disturbance run first confirms the steady state holds (smoke test)
- **Three-phase fault at the FAULT bus:** applied at t = 2 s, cleared at t = 2.15 s (150 ms)
- **Run −1 s to 10 s, 10 ms maximum step;** recorded: `EFD` and the GEN1 terminal voltage



The benchmark grid in PowerFactory's single-line diagram, with the load-flow results that set the operating point.

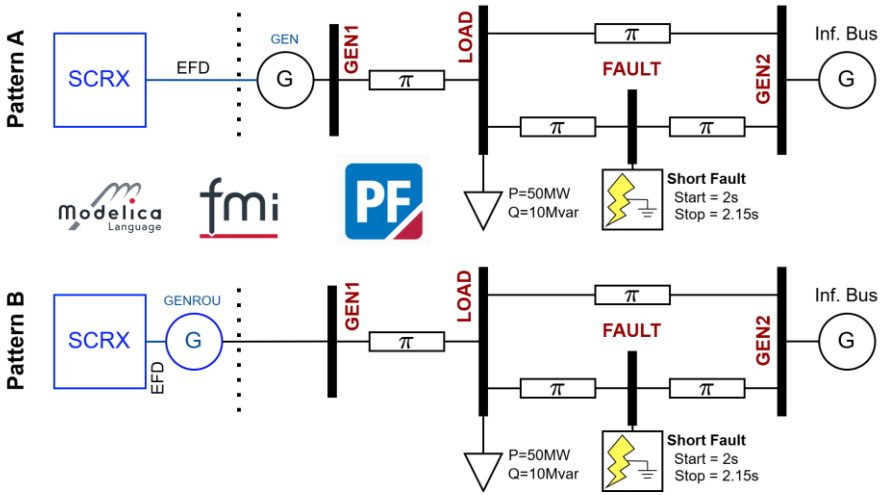
Three Configurations, One Reference

The reference comes first

- **All-native PowerFactory run:** PowerFactory's own SynMach machine with its PSS®E-compatible SCRX
  - this is the reference — every other run is scored against it by RMS error

The two FMI configurations

- **Pattern A — excitation system only:** FMU = OpenIPSL SCRX; machine and grid stay native
  - four scalars cross: `ECOMP` (measured terminal voltage), `ECOMP0` and `EFD0` (its load-flow value and the load-flow field voltage) in; `EFD` (field voltage) out
- **Pattern B — full machine:** FMU = GENROU + SCRX + PwPin2PF + 2 filters; GEN1 is replaced by the FMU plus a voltage sensor and a P/Q injector
  - four scalars cross: `vr`, `vi` in; `Pext`, `Qext` out
- **Both are Model Exchange:** one host integrator advances the FMU's states



Both patterns on the same grid; the dotted line marks the FMI boundary.

Configuration	Native in PowerFactory	Inside the FMU
Reference	GEN1 (SynMach + SCRX), GEN2, grid	—
Pattern A	GEN1 (SynMach), GEN2, grid	SCRX
Pattern B	GEN2, grid, sensor + P/Q slot	GENROU + SCRX + PwPin2PF + filters

grid = lines, load, shunt and external-grid equivalent

Initialisation: Why the FMU Needs Load-Flow Values, and How

Why: the OpenIPSL model must find its equilibrium

- **SCRX computes its own steady state from two numbers:** `EFD0` and `ECOMP0` at the operating point (they fix the regulator state and the voltage reference)
- **Only the load flow knows them:** they must be mapped into the FMU before the run

How: PowerFactory's "Initialisation" field

- **Per variable, Local or External:** the slot signals `EFD`, `ECOMP` are External — their start values come from the load flow
- **Local definition of input start values:**
  `EFD0 := EFD; ECOMP0 := ECOMP;`
  - right-hand sides = load-flow values, not the FMU's output
- **Not an FMI concept:** FMI 2.0 has six causalities; "External" is none of them — on the FMI side these are plain inputs or parameters

```
// SCRX.mo (repository): the PSS®E SCRX with its init values turned into inputs
RealInput EFD0   "Initial value of field voltage";
RealInput ECOMP0 "Initial value of the exciter input";
initial equation
  VR0   = EFD0/ECOMP0;
  V_REF = VR0/K + ECOMP0;
```

The equilibrium depends on the operating point — which only PowerFactory's load flow knows.



Variable Declarations: Initialisation = External on the slot-wired signal (here the output EFD)

Local definition of input start values

```
EFD0:=EFD;
ECOMP0:=ECOMP;
```

PowerFactory "Initialisation" ≠ FMI 2.0 causality

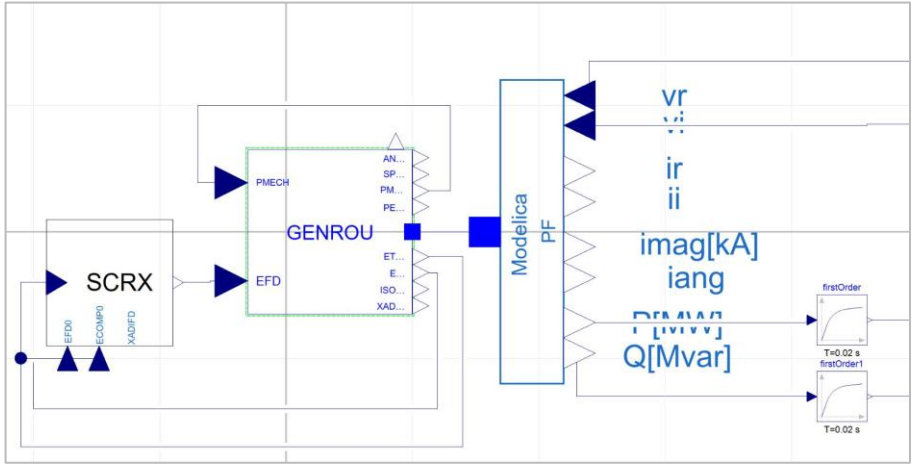
Inside the Pattern B FMU

What the FMU contains

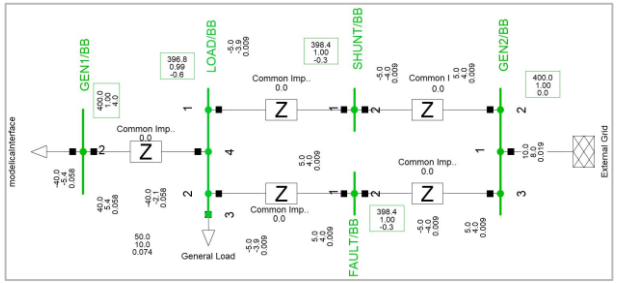
- **The same SCRX as Pattern A:** `EFD` drives GENROU and the terminal voltage returns as `ECOMP` — all inside the FMU
- **OpenIPSL GENROU:** round-rotor machine, six states, with saturation
- **A boundary block, PwPin2PF:** turns the acausal `PwPin` into directed signals, because PowerFactory imposes a causal interface
  - `vr`, `vi` in → P, Q out, scaled to MW and Mvar

What the grid sees

- **A controlled P/Q injector at the GEN1 bus**
- **Export lesson:** only top-level connectors become FMI inputs/outputs; internal variables stay internal — but the host can still record them, which is how we get the `EFD` trace



*The FMU in OMEdit: SCRX + GENROU + PwPin2PF ("Modelica PF" block) + two first-order filters (far right — **see slide 12**)*



Benchmark grid:
the FMU appears as "modelicainterface" at the GEN1 bus (far left)

The Boundary: Power Injection, Not Current Injection

What we expected to work

- **The textbook interface:** read the bus voltage, inject the machine current
  - the model has that connector — `genInterfaceCurrent` ships in the repository
- **It failed in PowerFactory:** the terminal voltage sagged to ≈ 0.8 p.u. for no physical reason and the run went unstable
- **We do not know why — our best guess:** the FMU interface was designed for user-defined controls inside a native machine, not for a user-defined source

What works

- **Read voltage, inject power:** `vr`, `vi` in → P, Q out into a load slot, sign flipped to injection convention
- **Counter-intuitive, but stable:** inside the FMU the machine is still a current source; only the exchange with the host is in P/Q

(a) Current injection — what we tried first



✗ V_t sags to ≈ 0.8 p.u. with no physical cause; the run goes unstable

(b) Power injection — what works



✓ Stable — the configuration used for every Pattern B result

Why? Unknown.
Our reading, not a documented fact: the FMU interface was built for user-defined controls bound to native machines; a user-defined source is outside what it was designed for. We report what worked, not why.

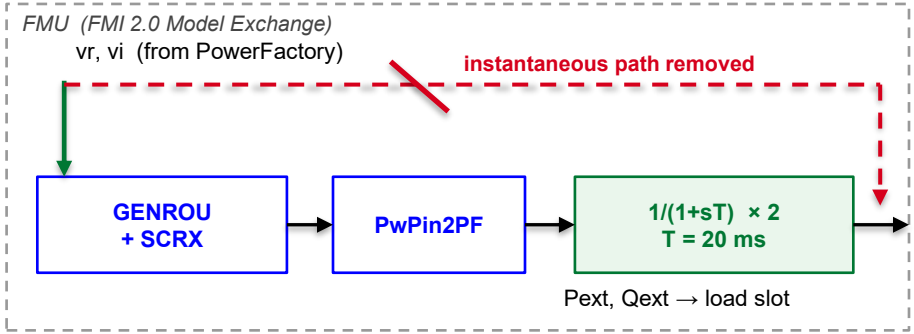
Why the Boundary Filters, and Why T = 20 ms

The problem: direct feedthrough

- **P and Q follow `vr`, `vi` instantly:** closing that path through the network equations puts an algebraic loop across the FMI boundary
- **A loop is allowed, but unverifiable here:** FMI 2.0 lets an importer iterate, PowerFactory solves DAEs — but nothing documents that its FMU importer does

What we did instead

- **Two first-order lags on P and Q only:** T = 20 ms — the outputs depend on filter states, not on the inputs
- **Choosing T:** above the 10 ms host step; well below the ≈ 1 Hz electromechanical band (≈ 7° of phase)
  - a few shorter values were tried: closer to the reference, but the solver went unstable — no systematic sweep
  - inside the loop the lag is a structural change to the model, not smoothing



green arrow: the state derivatives still depend on vr and vi

```
FirstOrder firstOrder(k=1, T=0.02,
           initType=Init.SteadyState);
<Outputs>  p ← firstOrder.y
           q ← firstOrder1.y
```

Top: the filter declaration (Modelica). Bottom: the <Outputs> dependencies in the exported modelDescription.xml — p and q (the frame's Pext, Qext) depend on the two filter states, and on neither input.

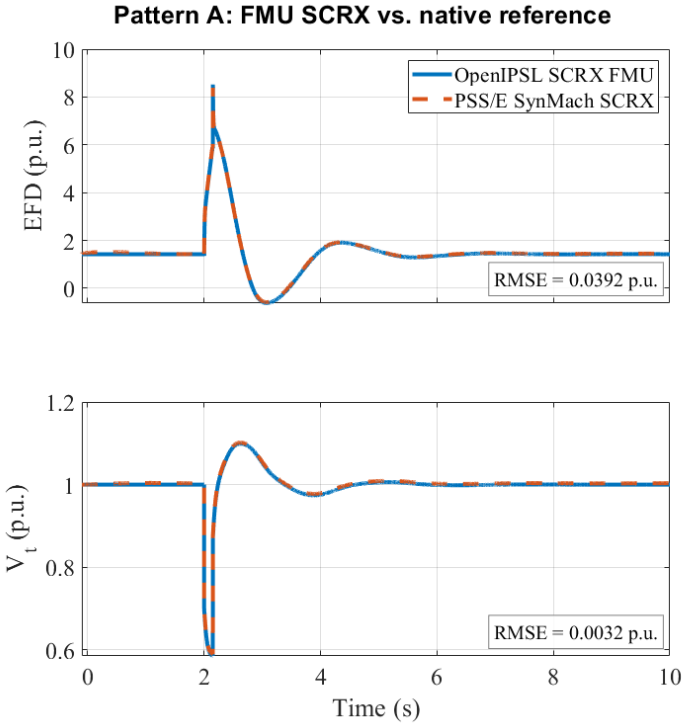
Pattern A Results: A Clean Swap of the Excitation System

What the overlay shows

- **Same starting point:** EFD = 1.434 p.u., V_t = 1.00 p.u. — the initialisation handshake works
- **Post-fault RMSE 0.039 p.u. on EFD, 0.003 p.u. on V_t:** the traces are indistinguishable except for the brief peak at t ≈ 2.1 s

What this tells us

- **The two SCRX implementations agree:** OpenIPSL's and PowerFactory's built-in one
- **The FMI import adds no visible error:** machine and grid were identical in both runs, so FMI was the only other thing that could differ
- **For Pattern B this means:** any difference there must come from the machine model or from the boundary — not from the excitation system, not from FMI



OpenIPSL SCRX FMU (solid) vs. the all-native reference (dashed).

Pattern B Results: Same Equilibrium, Different Transient

Steady state: as clean as Pattern A

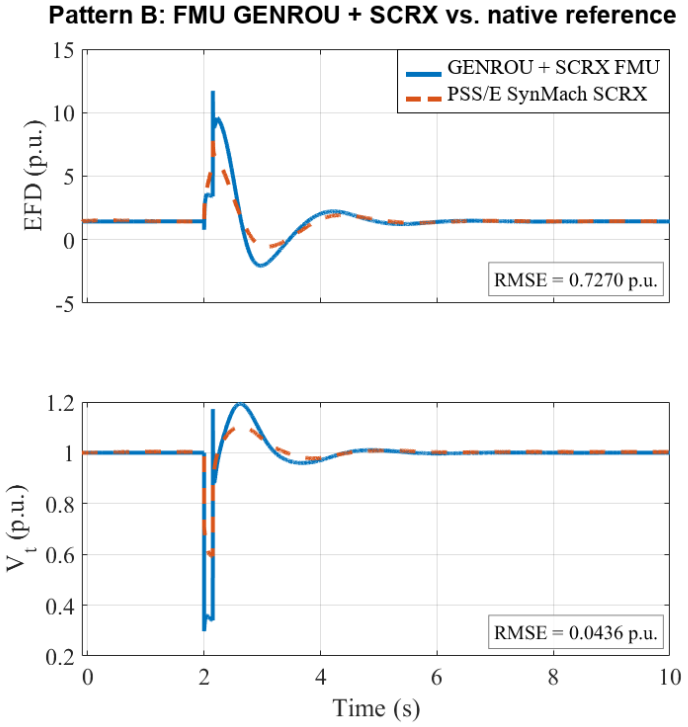
- **Same start, same finish:** both runs begin at `EFD` = 1.434 p.u., V_t = 1.00 p.u. and settle to the same post-fault steady state

Transient: it does not agree

- **RMSE 0.727 p.u. on `EFD`, 0.044 p.u. on V_t:** about 19× and 14× Pattern A
- **Larger `EFD` peak, deeper voltage dip:** ≈ 11 vs. ≈ 6.5 p.u.; V_t ≈ 0.3 vs. ≈ 0.6 p.u.

Two reasons

- **The machine inside the FMU is not PowerFactory's machine:** OpenIPSL's GENROU and PowerFactory's SynMach model the generator differently
- **The boundary adds dynamics:** the 20 ms filters and the P/Q representation change what the grid sees during the fault



Same grid, same operating point, same 150 ms fault — note the rescaled EFD and V_t axes.

What the Coupling Costs: Runtime

ALSETlab

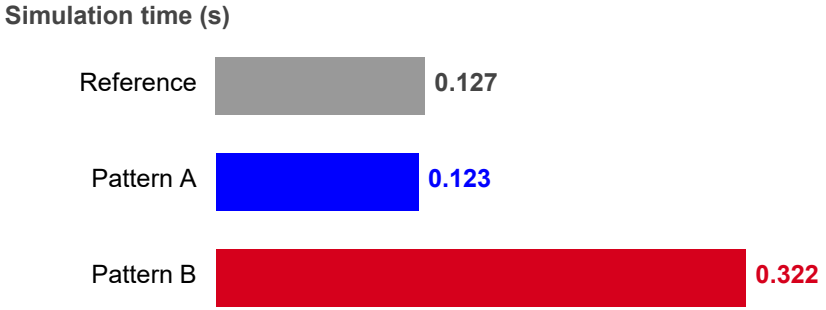
11 s simulated, 10 ms step, mean of 3 runs

- **Pattern A is free:** 0.133 s total, identical to the all-native run — one excitation system across the boundary costs nothing measurable
- **Pattern B costs ≈ 2.5×:** 0.322 s vs. 0.127 s simulation time; the extra is in simulation, not in initialisation

Where the extra time goes

- **Not a Newton loop inside the FMU:** checked in the exported code — its two nonlinear systems are initialisation-only; each step solves two small linear systems
- **A full machine through the FMI call interface every step:** six GENROU states plus the SCRX, plus the finite-difference boundary Jacobian the implicit host builds
- **Two extra filter states and the lag they add:** a filter is a delay; a different T could move the cost either way

Configuration	Init. (s)	Sim. (s)	Total (s)
Reference (all-native PowerFactory)	0.007	0.127	0.133
Pattern A	0.010	0.123	0.133
Pattern B	0.015	**0.322**	0.337



Absolute cost is well under a second — but Pattern B scales with the number of FMU-resident machines. Compile time (seconds for SCRX, tens of seconds for GENROU + SCRX) is a design cost, not a runtime one.

Which Pattern to Use

Aspect	Pattern A — excitation system	Pattern B — full machine
Interface surface	ECOMP, ECOMP0, EFD0, EFD	vr, vi, Pext, Qext
FMU contents	SCRX only	GENROU + SCRX + PwPin2PF + 2 filters
Initialisation	load-flow values map to External signals	two-stage: FMU back-solves its states
Algebraic coupling	loose	tight (mitigated by the 1st-order P/Q filters)
Step-size sensitivity	low	moderate
Runtime cost	none measurable (1.0×)	≈ 2.5×
Debuggability	high: excitation system isolable	lower: machine and grid entangled
Recommended use	novel controls	novel machine models

Choose Pattern A when…
a new control law (excitation system, governor, stabiliser) is under test against a trusted machine and grid.

Choose Pattern B when…
the machine model itself, or the machine–control interaction, is the thing under test.

Either way: the same three PowerFactory objects, the same load-flow initialisation — and check the steady state first.

- **Neither is uniformly better — the choice is study-driven.** Pattern A exposes only the excitation system; Pattern B exercises the full electromechanical dynamics.

Conclusions & GitHub Repository

The repository

- **github.com/ALSETLab/Modelica-FMI-PowerFactory-ModelImport-Tutorial** — BSD 3-Clause, archived on Zenodo
- **A 45-page screenshot tutorial:** both patterns click by click — export, frame, Model Type, initialisation handshake, slot wiring, RMS run

What you can run

- **Modelica source, pre-compiled FMUs, both PowerFactory projects** — including the PwPin2PF boundary block and the current-injection variant
- **Requirements:** PowerFactory 2022+ to run it; OpenModelica 1.22+ with OpenIPSL 3.1.0-dev and MSL 4.0.0 only to rebuild the FMUs

Scan → GitHub repository
Zenodo DOI 10.5281/zenodo.19859079

Further work

- **Next:** sweeps over fault severity, T and step size; the current-injection boundary; then governors, wind, PV and HVDC models, and multi-machine FMUs.

The route works

OpenIPSL → FMI 2.0 Model Exchange → PowerFactory's Composite Model Frame, initialised from the load flow — at both granularities.

Excitation system only: deployable today

RMSE 0.039 / 0.003 p.u. on EFD / V_t, and a runtime identical to the all-native run.

Full machine: steady state matches, the transient does not

RMSE 0.727 / 0.044 p.u. and ≈ 2.5× runtime. The machine model and the boundary differ from PowerFactory's own; FMI itself is not the cause.

What sets the fidelity is how much you put in the FMU — not FMI

The limits we hit were the exporter's defaults and our own boundary (P/Q plus filters). We do not claim that Pattern B is validated, and we do not blame the FMI standard.

References

- **Baudette, M. et al. (2018).** “OpenIPSL: Open-Instance Power System Library — Update 1.5 to “iTesla Power Systems Library (iPSL): A Modelica Library for Phasor Time-Domain Simulations””. SoftwareX 7, pp. 34-36.
- **Vanfretti, L. et al. (2016).** “iTesla Power Systems Library (iPSL): A Modelica Library for Phasor Time-Domain Simulations”. SoftwareX 5, pp. 84-88.
- **Fritzson, P. et al. (2018).** “The OpenModelica Integrated Modeling, Simulation and Optimization Environment”. 1st American Modelica Conf.
- **Blochwitz, T. et al. (2011).** “The Functional Mockup Interface for Tool Independent Exchange of Simulation Models”. 8th Int. Modelica Conf.
- **Modelica Association Project FMI (2022).** “Functional Mock-up Interface for Model Exchange and Co-Simulation”, v2.0.4 — source for the six causality values and the Set/Get loop-resolution provision (slides 9, 12).
- **Junghanns, A. et al. (2021).** “The Functional Mock-up Interface 3.0 — New Features Enabling New Applications”. 14th Int. Modelica Conf. — cited in the paper (FMI 3.0 event indicators and terminals).
- **DIgSILENT GmbH.** “Overview: Modelling in PowerFactory, Model Import and Model Export” (product documentation) — source for the 2022 FMI / discrete-time and 2025 continuous-time Modelica timeline (slide 4).
- **van der Meer, A. A. et al. (2019).** “Towards Scalable FMI-based Co-simulation of Wind Energy Systems Using PowerFactory”. IEEE PES ISGT-Europe.
- **Basumallik, S. et al. (2025).** “Enhancing Large-Scale Power Systems Simulations through Functional Mock-up Unit-based Grid-Forming Inverter Models”. 16th Int. Modelica & FMI Conf.
- **Islam, M. S. et al. (2023).** “Modeling Specialized Electric Power Generators, Excitation Systems and Prime Movers Used by North American Utilities”. 15th Int. Modelica Conf.
- **Weber, J. (2015).** “Description of Machine Models GENROU, GENSAL, GENTPF and GENTPJ” — source for the per-unit-speed stator-flux and saturation conventions (slide 17).
- **NERC (2018).** “Reliability Guideline: Power Plant Model Verification and Testing for Synchronous Machines” — “use of numerical thresholds for sufficient match are not recommended” (slide 17).
- **Bastian, J. et al. (2011); Gomes, C. et al. (2018).** “Master for Co-Simulation Using FMI”; “Co-Simulation: A Survey” — cited in the paper only to record that macro-step co-simulation results do not transfer to Model Exchange.
- **Chang, H. and L. Vanfretti (2025).** Modelica-FMI-PowerFactory-ModelImport-Tutorial. Zenodo, DOI 10.5281/zenodo.19859079.

Thank You — Questions?

Coupling granularity, not the FMI standard, sets the fidelity you get.

Rensselaer

why not change the world?®

github.com/ALSETLab/
Modelica-FMI-PowerFactory-ModelImport-Tutorial
Zenodo DOI 10.5281/zenodo.19859079