

PowerIO: Compiler infrastructure for power systems

Translating power system data formats with a shared, typed library

Samuel Talkington
Harvard SEAS

September 11, 2026, PowerUp 2026, Boulder
Workshop of the IEEE PES Task Force on Benchmarking Multiconductor OPF

Everyone wants the parser, nobody wants the rest of the stack

- `parse_file("case118.m")` loads JuMP and its whole dependency tree.
- PowerModels.jl #894 asks for a standalone parser. PMD #398 asked to parse `.dss` through OpenDSSDirect.jl instead.
- Re-implemented readers break on real cases: PowerModels 0.21.6, pandapower 3.2.2, PMD 0.16.

PowerModels.jl #894

"Sometimes I want to read a matpower file, but don't want to load/precompile JuMP and all the rest of the package dependencies of PowerModels."

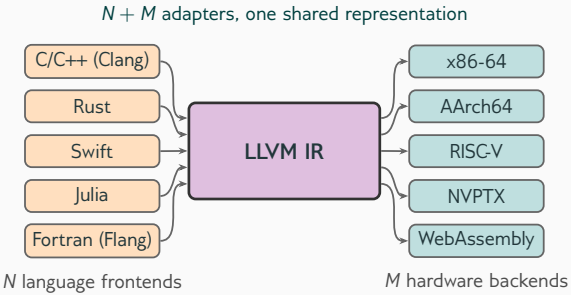
Julia PowerModels, PMD	**Python** pandapower, PyPSA
MATLAB MATPOWER	**Java, Delphi** PowSybl, OpenDSS

four ecosystems, four private parsers

PowerModels.jl #894 (2023, open); PowerModelsDistribution.jl #398 (2022, closed); failures reproduced in eigenergy/powerio `evals`.

LLVM shares one intermediate representation

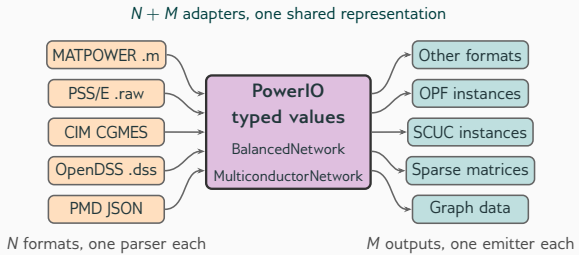
- LLVM (UIUC, 2000): reusable compiler libraries built around one typed IR.
- Frontends emit the IR, backends consume it. A new language or target is one new piece.
- Clang, Rust, Swift, and Julia all compile through it.



llvm.org. C. Lattner and V. Adve, "LLVM: A Compilation Framework for Lifelong Program Analysis and Transformation," CGO 2004. C. Lattner et al., "MLIR," CGO 2021.

PowerIO does the same for power system data

- A parse returns typed values plus diagnostics, not a dictionary.
- Two network types, balanced and multiconductor. Each format needs one parser and one emitter.
- Borrowed: a small core, diagnostics, explicit transformations. Left out: low-level compiler machinery. A bus stays a bus.



eigenergy/powerio documentation, “LLVM and MLIR lessons” and “Architecture,” eigenergy.github.io/powerio.

PowerIO never solves: it reads, checks, and emits

Three rules

1. Same format out: the source bytes, exactly.
2. Another format out: what fits, with every loss reported.
3. Changing network type: explicit, never silent.

- 12 balanced and 3 multiconductor formats, including OpenDSS and BMOPF draft JSON.
- Rust, Python, Julia, C, and WebAssembly; 1,800+ tests, 14 fuzz targets.
- case9241pegase parse: 10 ms vs. 0.79 s in PowerModels.jl.
- Downstream: tellegen, ExaModelsPower, BMOPFTools.

MIT or Apache-2.0. Timing: `evals/performance/RESULTS.md`, Rust and Julia bindings, one machine; the PowerModels.jl 0.21.6 parse also validates.

One command: OpenDSS in, schema-valid BMOPF JSON out


```
$ powerio convert ieee34Mod1.dss -to bmopf-json -o ieee34.bmopf.json
```

ieee34Mod1.dss and IEEELineCodes.DSS (excerpt)

```
New linecode.300 nphases=3 basefreq=60 units=kft
~ rmatrix = [0.253181818 | 0.039791667
  0.250719697 | ...]
New Line.L1 Phases=3 Bus1=800.1.2.3
  Bus2=802.1.2.3 LineCode=300 Length=2.58
  units=kft
New Load.S860 Bus1=860 Phases=3 Conn=Wye Model=1
  kV= 24.900 kW= 60.0 kVAR= 48.0
```

ieee34.bmopf.json (BMOPF draft schema 0.1.0)

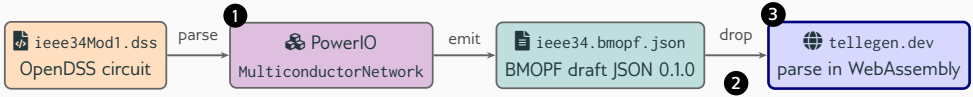
```
"line": { "L1": { "bus_from": "800",
  "bus_to": "802", "length": 786.384,
  "linecode": "300", ... } },
"linecode": { "300": { ...,
  "R_series_1_1": 0.00083064..., ... } },
"load": { "S860": { "bus": "860", ...,
  "model": "CONSTANT_POWER",
  "p_nom": [20000.0, 20000.0, 20000.0], ... } }
```

Emitted `.dss` re-solves in OpenDSS within 10^{-8} pu (IEEE 13, 34, 123).

Validates against draft schema 0.1.0 and reparses stably; 306 losses reported, e.g. `vminpu` and RegControl settings.

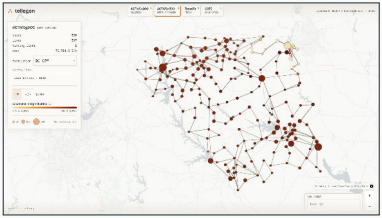
Live: drop the BMOPF JSON on tellegen.dev, parsed in the browser

DEMO



1. Convert with the command on the previous slide.
2. Drop `ieee34.bmopf.json` on tellegen.dev. It is parsed in the browser.
3. Today: balanced cases solve, multiconductor cases display. AC OPF is in progress.

tellegen.dev



tellegen.dev today: DC OPF and SOCWR in WebAssembly.

eigenergy/tellegen: case parsing uses powerio; local case files never leave the browser.

Questions?

samuel_talkington@seas.harvard.edu

github.com/eigenergy/powerio

eigenergy.github.io/powerio

tellegen.dev

```
cargo add powerio   pip install powerio   Pkg.add("PowerIO")
```





eigenergy