Open Source Tools for System Operators

ESIG & G-PST Pillar 5 webinar
2022-12-08

Clayton Barrows
Juha Kiviluoma

System operator viewpoints

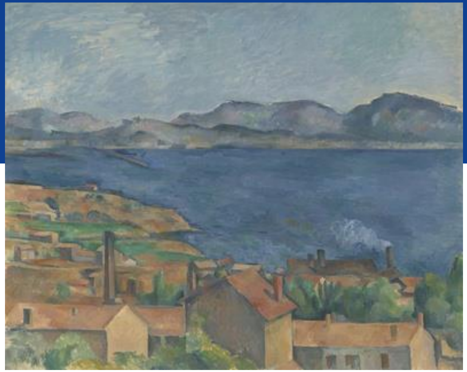
- Motivation to move to open-source (from RTE and EnergiNet presentations)
  - New tools needed to improve speed and cost-efficiency – utilize software innovations
  - Customization not dependent on specific vendor
  - Improved modularity, interoperability, evolutivity and shorter release time cycles
  - Sharing efforts instead of duplicating
  - Adds transparency
  - Resource of open-source communities: diverse skills and viewpoints
  - A platform for collaboration and co-creation, thinking differently and continuous learning
- Open-source (and other digitalization efforts) requires competence and culture
- Requirements
  - Support
  - Validation and benchmarking
  - Modularity and interoperability
  - Security: screen all code changes, keep power system security vulnerabilities confidential

For starters

- What are the steps that are required before open-source tools can become adopted by professionals in the field
  1. Trust in the process
  2. Validated results
  3. Available support
  4. Usability
  5. Documentation
  6. Compatibility and interoperability
  7. Computational efficiency
- Level of maturity

Paintings: Art Institute of Chicago (CC0)

Step 1: Trust in the process



- The trivial – open code allows code review
  - The actual model can be scrutinized
  - Challenge: code readability
  - Challenge: correspondence with possible mathematical formulation
  - Challenge: modern software relies on a large stack of code
- Has someone reviewed the whole code stack
  - Are there potential risks – if there are, how they are minimized
    - e.g. using only package versions that have been around for a while
  - How the introduction of new packages or new package versions is managed
- How new model code is managed
  - What kind of merge process
  - Who have the rights
  - Are there tests, how comprehensive they are

Step 2a: Validation of the tools

- Validation against reality
  - Can be good for simpler systems
  - For big systems, not a good measure – reality is too messy
  - Never a perfect match and hard to know why
- Validation against other models
  - Gives some confidence, but
    - Are the other model/models correct?
    - Large system models
  - Pillar 5 activities on benchmarking!
- Correct functioning
  - Unit tests
    - Can be comprehensive
    - Can be measured
  - System tests
    - How features interact
    - Probably cannot be comprehensive

Step 2b: Open certification process

- EU project Mopo aims to develop open certification process for planning tools (starting from SpineOpt.jl)
- System tests could be a shared effort between different modelling efforts working the same domain
- A common database of system tests and correct results
  - For different modelling purposes
- Requires an ontology
- Can allow performance comparisons
- Can allow feature comparisons

Step 3: Support

- For operational tools: 24/7
- For planning tools: ~within working day
- Ways to achieve
  - Developer(s) perform also support function
  - Separate company/companies
  - Part of the support can be between SOs
    - Peer support can be useful especially for planning timescale and for more complex features
    - Reduces reliance on third parties

Step 4: Usability

- Many open-source tools have roots in research
- Usability may have been a secondary concern
- Ways to improve
  - Wrap in an external GUI
  - Develop own GUI
  - In any case: a clear API helps (where does your model start and where does it end)
  - Make a translation layer between user-friendly input and model code (and back)
- Avenues to get there
  - Research funding (can be difficult)
  - G-PST funding
  - In-kind collaboration
  - Investment (requires sales)

Step 5: Documentation

- Tutorials to get acquinted
- Explainers for complex but essential features
- Demo systems to show how to address specific modelling needs
- Reference section to cover everything (through documented code)
- For mature tools: (online) courses (link to G-PST Pillar 2 and 3)

- Maintenance of documentation

Step 6: Compatibility and interoperability

- TSOs have legacy systems that need to work
- Modularity of new tools
- Interoperability
  - APIs and conformance with data specifications
  - Flexible data structures
- Compatibility over time (support and version control)

Step 7: Computational efficiency

- High computational efficiency not required in all use cases
- But it is always very nice
- All regular software best-practices apply
- Model formulation is an additional wrinkle
- A large developer community more likely to find good solutions than a small community

How SOs (and others) could participate

- Each SO needs to ensure the trustworthiness of their tools in any case
  - Open-source allows to become partners in building trust
  - Sharing responsibility together with other SOs in the trust ensurance
  - In practice: Participation in the code review and code merging process
  - E.g. knowing that three other SOs have verified a package let's you focus on verifying something else
- Provide user feedback – what would be important to improve from practical perspective
- Increased visibility to what tools have been verified and in what way
- Grow the snowball: SO involvement brings credibility and indication of usefulness, this helps with further funding efforts

What is success?

Industry usage will define ultimate success for open-source software that supports the power system transformation.

Industrial adoption will require **robust user and development communities** to ensure validity, reliability, and longevity of open-source projects, and to provide the required support services.

Areas where we have seen success

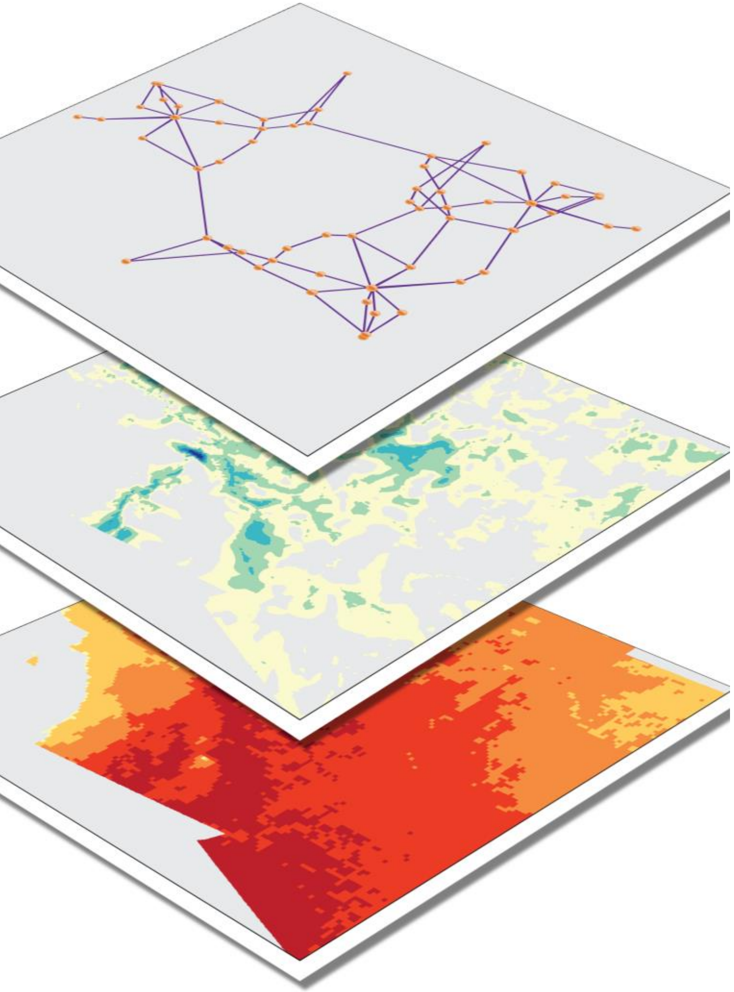
- Applications that require highly customizable solutions:
  - Analytics (Pandapower + Pandas + Python)
  - Research/emerging methods (PowerModels.jl)
- Applications that enable interoperability and/or data transfer:
  - Co-simulation (Helics)
  - Workflow orchestration and data management
  - Data standards/specifications (CIM)

G-PST – LF-Energy Benchmarking Exercises

- Event 1 (March 2022): Power Flow Benchmarking
  - Multiple open-source packages
  - Common problem and dataset
  - Match results
  - Compare performance and workflow

Contributors

- G-PST
  - Hannele Holtenen – Recognis
  - Juha Kiviluoma – VTT
- PyPSA
  - Max Parzen – University of Edinburgh
- pandapower
  - Roman Bolgaryn – Fraunhofer IEE
  - Sadia Ferdous Snigdha – Fraunhofer IEE & TU Ilmenau
- PowSybl
  - Benoit Jeanson – RTE
  - Nicolas Omont – Artelys
- PowerSimulations.jl
  - Clayton Barrows – NREL
  - Dheepak Krishnamurthy – NREL



Data

- RTS-GMLC (github.com/gridmod/rts-gmlc)
  - 73 buses, 120 branches, and 115 generators
- PEGASE-9241 (matpower.org/docs/ref/matpower6.0/case9241pegase.html)
  - 9,241 buses, 16,049 branches, and 1,445 generators

Problem and Workflow Specification

- Power flow:
  - Static bus control (no PV-PQ transitions)
  - Single slack bus definition
- Workflow
  - Load data from MATPOWER case file (.m or .mat)
  - Calculate power flow
  - Export results to csv file

Value Proposition

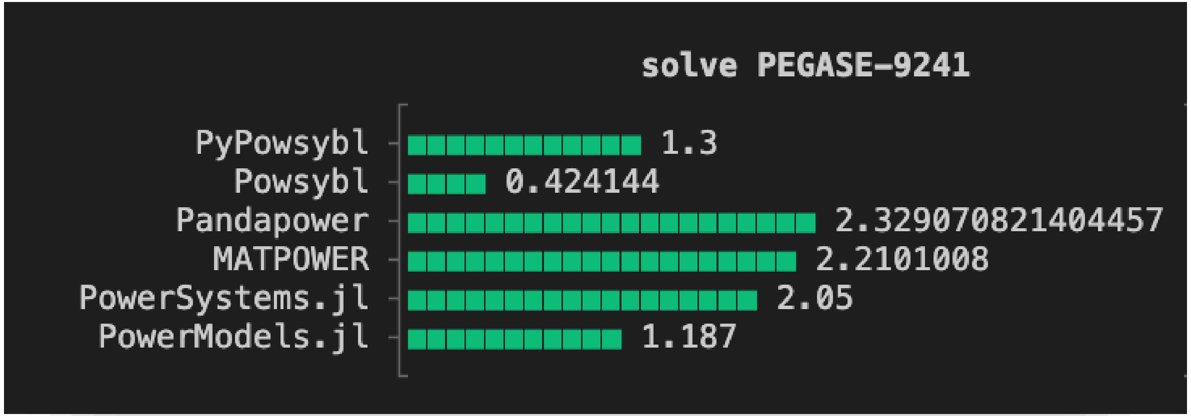
- Benchmarking:
  - understand the strengths/weaknesses of different tools, validation, identify opportunities for improvement

- Coordination:
  - standardize problem formulations and data specifications, enhance/build tool interoperability

PEGASE-9241 Benchmark comparison

- Benchmark Notes:
  - All times in seconds
  - pandapower: benchmark time displayed includes compilation (example 1st execution required 1.66 s vs. 0.13 s for 2nd execution)
  - PowerSystems.jl & PowerModels.jl: times exclude compilation

Challenges
- Data errors
- Slack bus
- 3-winding transformers
- Generator/bus voltage setting disagreement



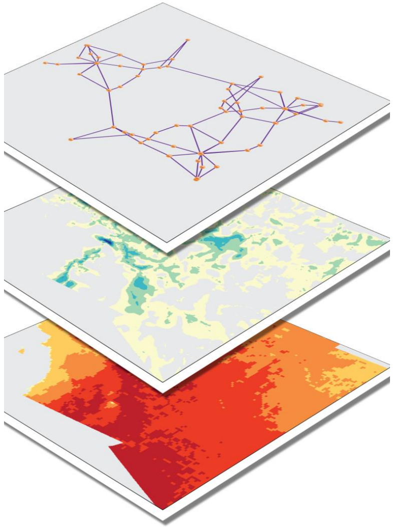
G-PST – LF-Energy Benchmarking Exercises

- Event 2 (June 2022): Capacity Expansion Coordination
  - Multiple open-source packages
  - Coordinated workflow with distinct contributions from each tool
  - Demonstrate interoperability and capabilities

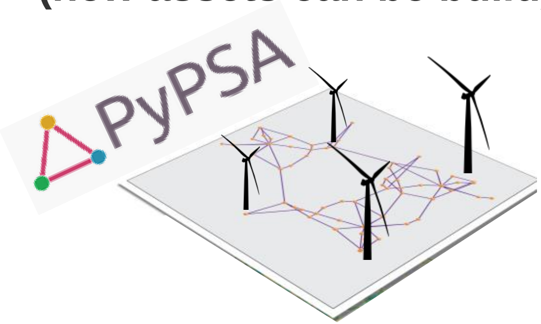
Problem and Workflow Specification

- Event 2: System expansion planning and analysis

Existing RTS-GMLC



New **Capacity expansion (new assets can be build)**



- Perform invest and dispatch *co-optimization*
- Add investment costs
- Add constraints (i.e.CO2)
- Add load scenario

Operational analysis (PyPSA results valid?)

pandapower

LF ENERGY POWSYBL

PowerSystems.jl
PowerSimulations.jl

Need for better data specifications and formats

- Common Information Model (CIM) designed for operational data exchange
- Model specific (MATPOWER, PSS/e .raw) formats are incomplete
- Is there a need for something in-between?
  - RTS-GMLC (not well defined, but could be a starting point)

Outcomes

- PyPSA2PowerSystems.jl
- PowerSystems.jl -> PowerModels.jl -> MATPOWER .m
- RTS-GMLC -> PowSyBl-Metrix
- RTS-GMLC -> PyPSA
- RTS-GMLC -> pandapower
- Improved pandapower -> PyPSA

G-PST – LF-Energy Benchmarking Exercises

- Event 3 (Spring 2023): Transient Stability Modeling
  - Multiple open-source packages
  - Common problem and dataset
  - Match results
  - Compare performance and workflow