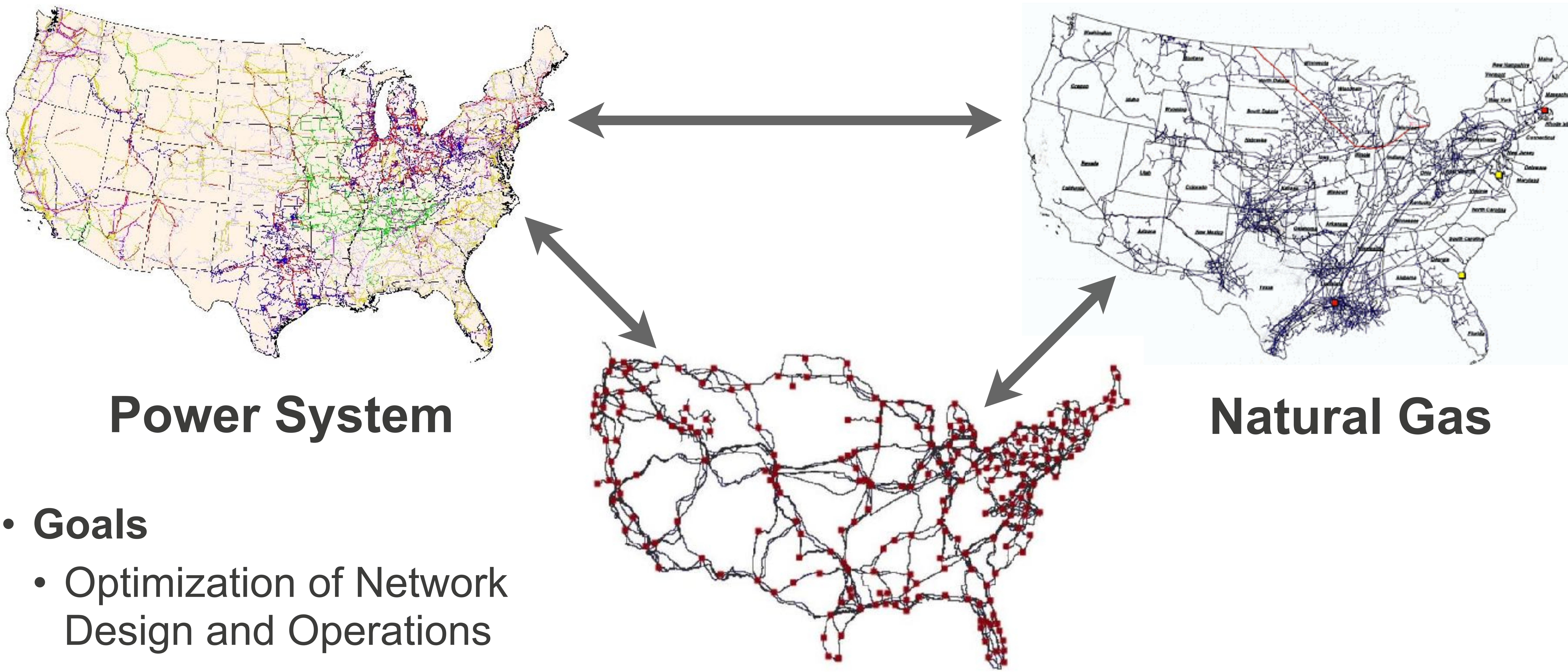
Infrastructure Optimization in Julia

Carleton Coffrin, et. al.
Los Alamos National Laboratory
Advanced Network Science Initiative

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA

LA-UR-18-31741

National Infrastructure Networks



Power System

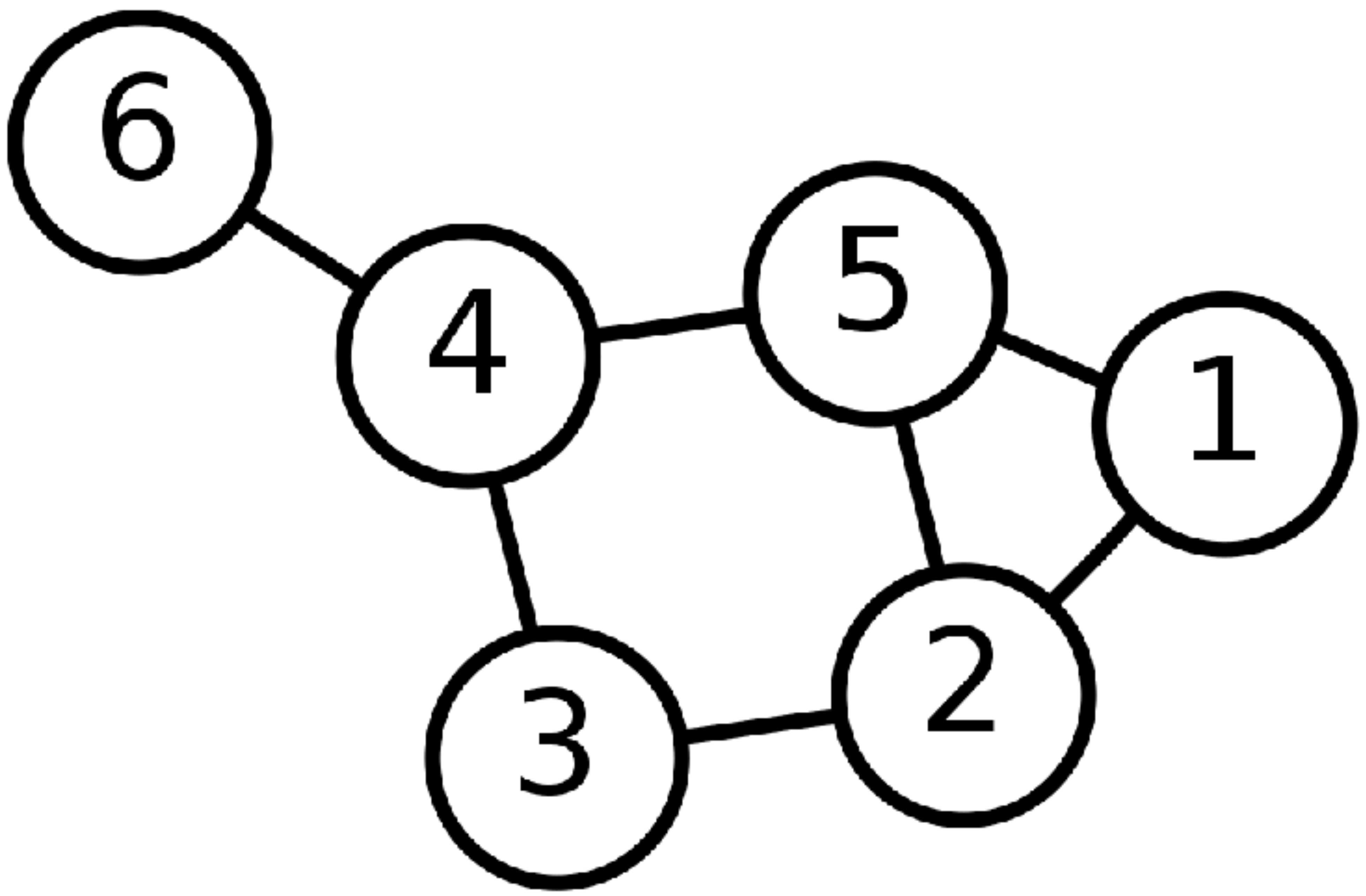
Natural Gas

Communications (Fiber)

- **Goals**
  - Optimization of Network Design and Operations
  - Understand Interdependencies

Common Threads

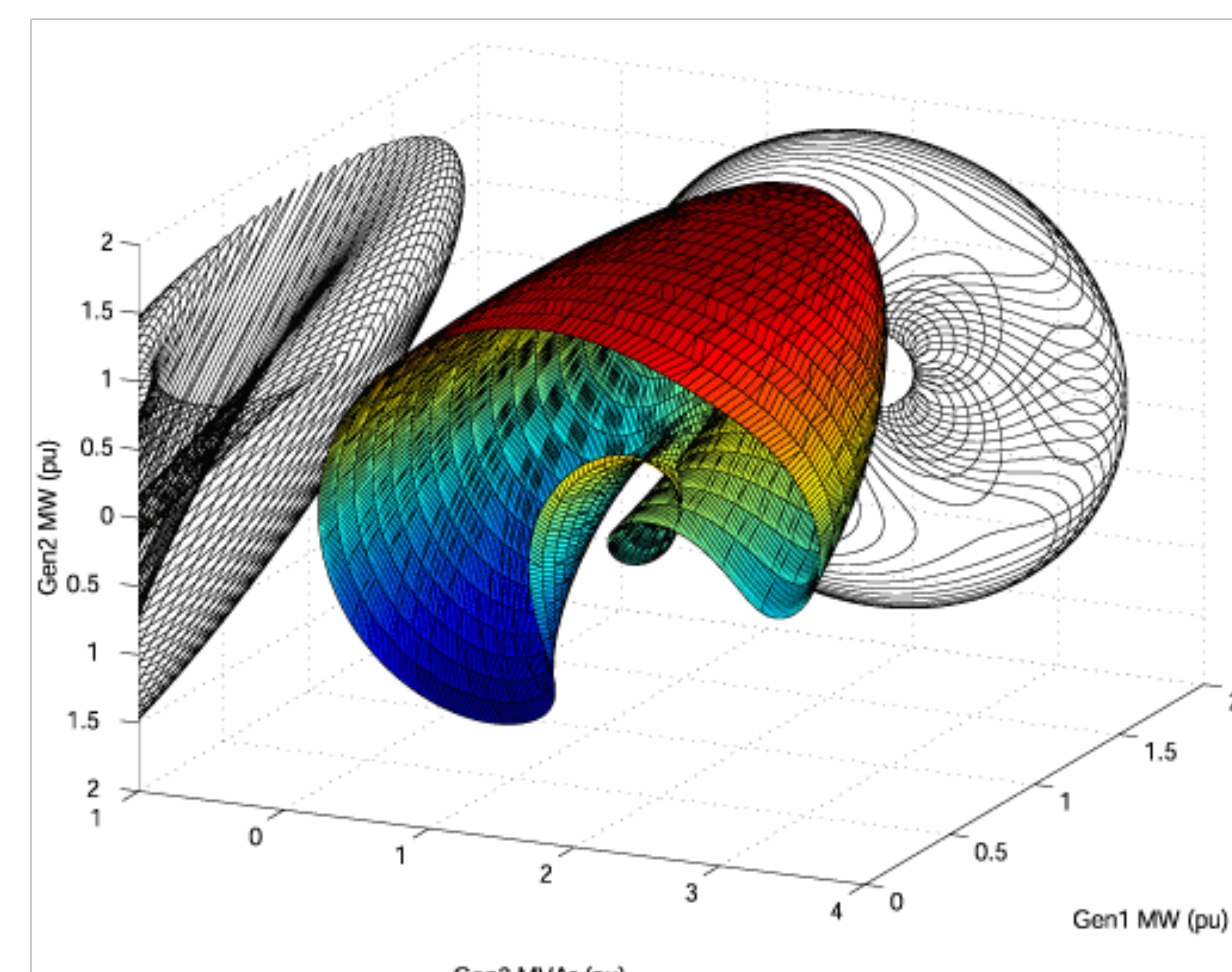
Graph Structure



Physics Equations

$$S_{ij} = \boldsymbol{Y}_{ij}^* V_i V_i^* - \boldsymbol{Y}_{ij}^* V_i V_j^*$$

$$S_i^g - \boldsymbol{S}_i^d = \sum S_{ij}$$

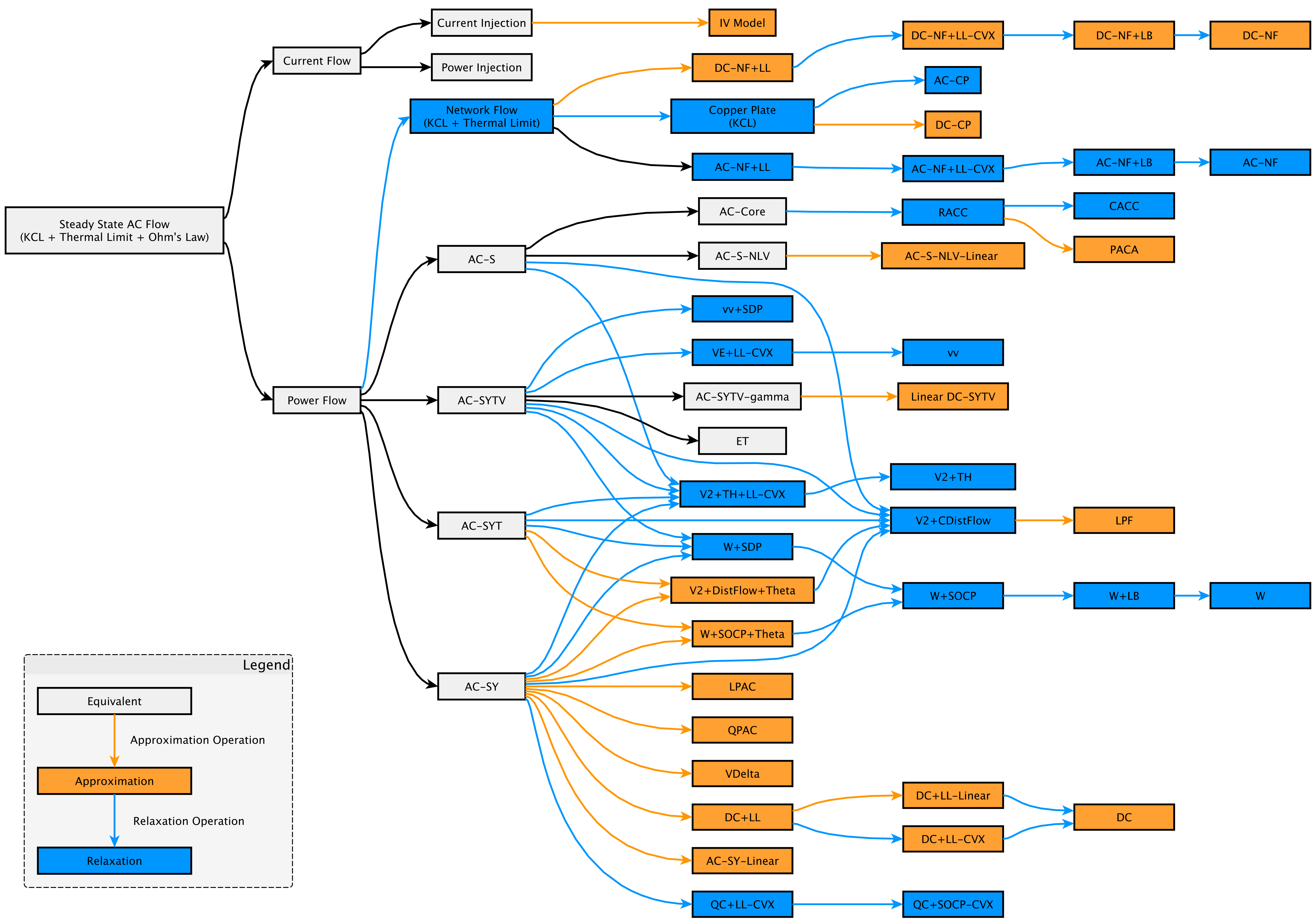


I.A. Hiskens and R.J. Davy, **"Exploring the power flow solution space boundary"**
IEEE Transactions on Power Systems, Vol. 16, No. 3, August 2001, pp. 389-395

Challenges

- **Explosion of physics formulations in recent years**

Taxonomy of Power Flow Formulations (2014)



Also see:
D.K. Molzahn and I.A. Hiskens, **"A Survey of Relaxations and Approximations of the Power Flow Equations,"** to appear in Foundation and Trends in Electric Power Systems, 2018.

Challenges

- **Extremely Wide Range of Optimization Problem Classes**
- **NLP: nonlinear physics in operations**
- **Mixed-Integer: network design decisions**
- **QP / SOC / SDP: convex relaxations**
- **LP: linear approximations**

Three Core Ideas

- **Develop a Infrastructure Network Optimization modeling layer**
  - all formulations share a common mathematical program specification
  - switching formulations (and problem class) is effortless

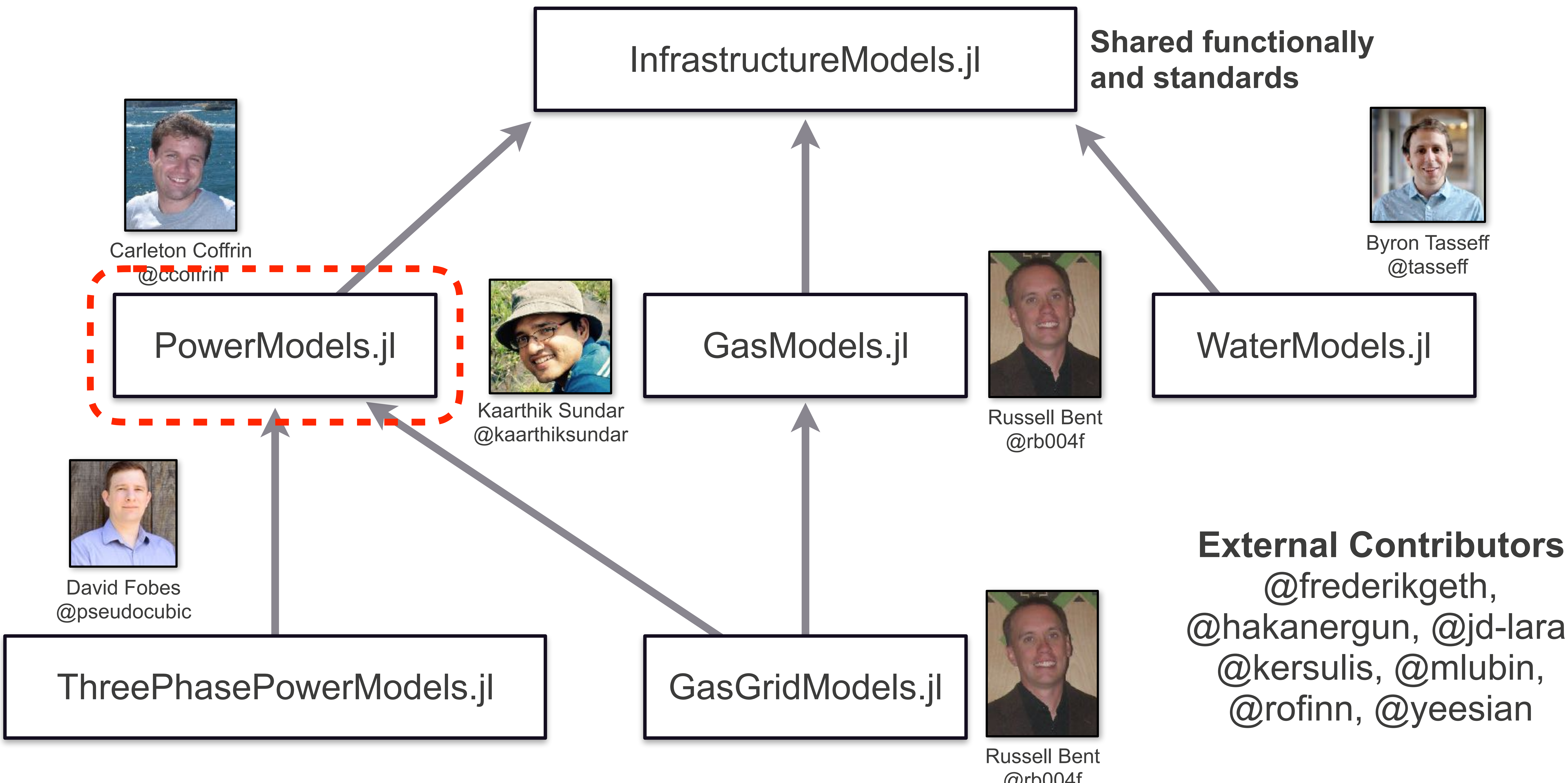
- **Open-source and community driven**
  - formulation authors contribute their implementation
  - most well suited to provide the best version of their formulation idea

- **Common implementation platform**
  - shared problem specification ensure same equations and constraints
  - side-by-side comparisons (e.g. benchmarking) are much more accurate

InfrastructureModels.jl

Open-source Julia packages for Infrastructure Network Optimization

Infrastructure Models Packages



Theme of InfrastructureModels Packages

- Infrastructure-aware JuMP-like modeling layer

- Minimalist Matlab data formats for R&D

- Parsers for Established Data Files
  - e.g. Matpower, PSSE v33, OpenDSS, EPANET

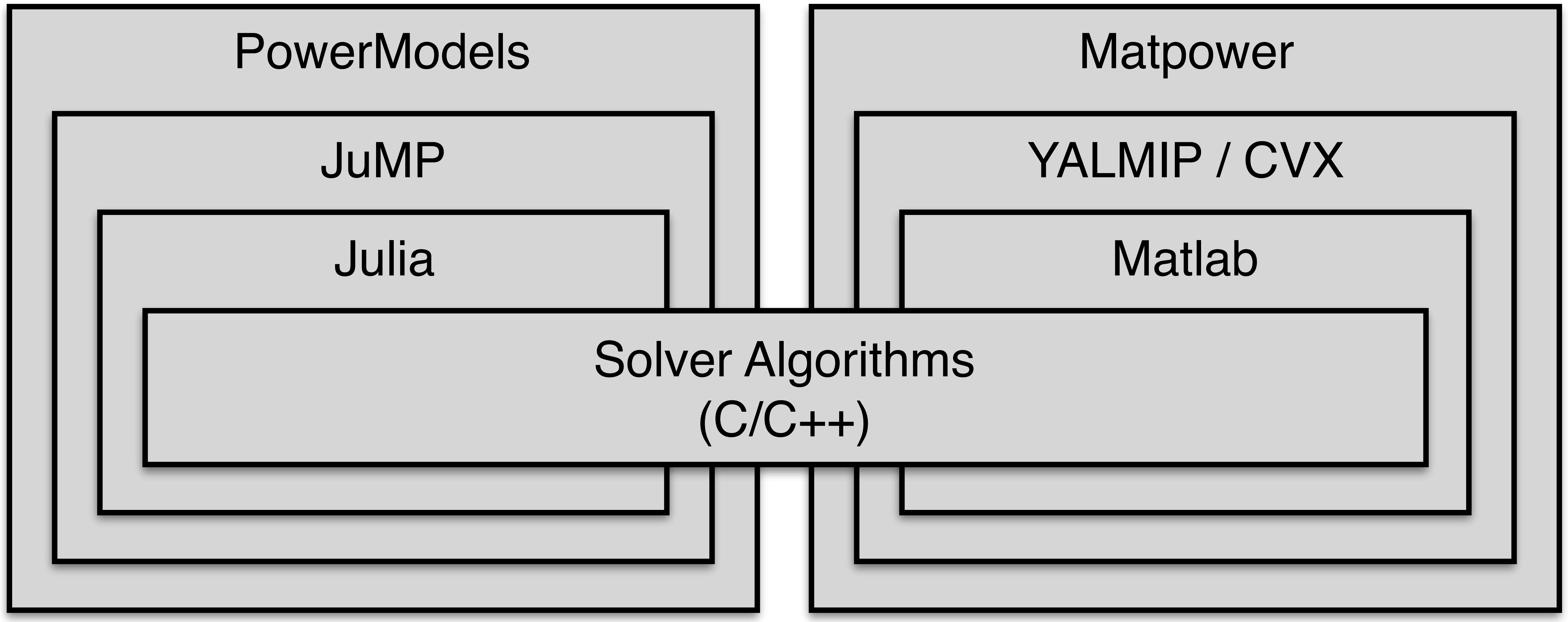
- Cross Infrastructure Data Model Standards
  - Lists of components, organized by component type
  - Universal fields
    - “index” - unique integer id for component
    - “status” - makes components inactive
    - “name” - human readable name

- Goal: Usage of all InfrastructureModels packages “feels” similar

PowerModels.jl

What is PowerModels?

- Data Parsers
  - Matpower (.m)
  - PSSE v33 (.raw)
- Problem Specifications
  - PF, OPF, OTS, TNEP, …
- Formulations
  - AC-Polar, AC-Rect, DC, DC+LL, SDP, SOC, QC, …
- Documentation
  - https://lanl-ansi.github.io/PowerModels.jl/stable/
- Extensions
  - PowerModelsAnnex.jl, ThreePhasePowerModels.jl, PowerModelsReliability.jl, …



Component Model Mapping

Matpower

- Bus+Load+Shunt
- Generator
- Branch (Pi-Model)
  - AC Line
  - Transformer
- Simple DC Line
- Generator Cost
- DC Line Cost

PowerModels

- Bus
- Load
- Shunt
- Generator
- Branch (Pi-Model)
  - AC Line
  - Transformer
- Simple DC Line
- Generator Cost
- DC Line Cost

PSSE (Subset)

- Bus
- ZIP Load
- Fixed Bus Shunt
- Switched Bus Shunt
- Generator
- AC Line (Pi-Model)
- Transformer (T-Model)
  - Two Winding
  - Three Winding
- DC Line
  - Two-Terminal
  - VSC

Want to know more?

https://youtu.be/j7r4onyiNRQ

Example - Reading Data

```
using PowerModels

network_data = PowerModels.parse_file(“pglib_opf_case5_pjm.m”)
```

```
function mpc = pglib_opf_case5_pjm
mpc.version = '2';
mpc.baseMVA = 100.0;

mpc.bus = [
  1  2    0.0    0.00  0.0  0.0  1  1.00000  0.00000  230.0  1  1.10000  0.90000;
  2  1  300.0   98.61  0.0  0.0  1  1.00000  0.00000  230.0  1  1.10000  0.90000;
  3  2  300.0   98.61  0.0  0.0  1  1.00000  0.00000  230.0  1  1.10000  0.90000;
  4  3  400.0  131.47  0.0  0.0  1  1.00000  0.00000  230.0  1  1.10000  0.90000;
  5  2    0.0    0.00  0.0  0.0  1  1.00000  0.00000  230.0  1  1.10000  0.90000;
];

mpc.branch = [
  1  2  0.00281  0.0281  0.00712  400  400  400  0.0  0.0  1  -30.0  30.0;
  1  4  0.00304  0.0304  0.00658  426  426  426  0.0  0.0  1  -30.0  30.0;
  1  5  0.00064  0.0064  0.03126  426  426  426  0.0  0.0  1  -30.0  30.0;
  2  3  0.00108  0.0108  0.01852  426  426  426  0.0  0.0  1  -30.0  30.0;
  3  4  0.00297  0.0297  0.00674  426  426  426  0.0  0.0  1  -30.0  30.0;
  4  5  0.00297  0.0297  0.00674  240  240  240  0.0  0.0  1  -30.0  30.0;
];

...
```

```
Dict{String,Any} with 13 entries:
  "source_type"    => "matpower"
  "name"           => "pglib_opf_case5_pjm"
  "source_version" => v"2.0.0"
  "baseMVA"        => 100.0
  "per_unit"       => true
  "bus"            => Dict{String,Any}(...)
  "branch"         => Dict{String,Any}(...)
  "dcline"         => Dict{String,Any}(...)
  "gen"            => Dict{String,Any}(...)
  "load"           => Dict{String,Any}(...)
  "shunt"          => Dict{String,Any}(...)
  "storage"        => Dict{String,Any}(...)
```

```
network_data["bus"]["1"]["vm"] ... 1.0
network_data["branch"]["3"]["br_x"] ... 0.0064
```

Example - Varying the Problem Formulation

```
using PowerModels; using Ipopt
solver = IpoptSolver()

result = run_ac_opf(network_data, solver)

result = run_dc_opf(network_data, solver)


run_opf(network_data, ACPPowerModel, solver)

run_opf(network_data, DCPPowerModel, solver)

run_opf(network_data, SOCWRPowerModel, solver)
```

Non-Convex Form

Linear Approximation

Convex Relaxation

Shared Problem Specification

Example - Varying the Problem Class

Problem Class

Non-Convex Formulation

Linear Approximation

Convex Relaxation

```
using PowerModels; using Ipopt
solver = IpoptSolver()

# Base Non-Convex Model
run_pf("pglib_opf_case5_pjm.m", ACPPowerModel, solver)
run_opf("pglib_opf_case5_pjm.m", ACPPowerModel, solver)
run_ots("pglib_opf_case5_pjm.m", ACPPowerModel, solver)

# Linear Approximation
run_pf("pglib_opf_case5_pjm.m", DCPPowerModel, solver)
run_opf("pglib_opf_case5_pjm.m", DCPPowerModel, solver)
run_ots("pglib_opf_case5_pjm.m", DCPPowerModel, solver)

# Convex Relaxation
run_pf("pglib_opf_case5_pjm.m", SOCWRPowerModel, solver)
run_opf("pglib_opf_case5_pjm.m", SOCWRPowerModel, solver)
run_ots("pglib_opf_case5_pjm.m", SOCWRPowerModel, solver)
```

PowerModels Mathematical Modeling Layer

```
function post_opf(pm::GenericPowerModel)

    variable_voltage(pm)
    variable_generation(pm)
    variable_branch_flow(pm)

    objective_min_fuel_cost(pm)

    constraint_voltage(pm)

    for i in ids(pm, :ref_buses)
        constraint_theta_ref(pm, i)
    end

    for i in ids(pm, :bus)
        constraint_kcl_shunt(pm, i)
    end

    for i in ids(pm, :branch)
        constraint_ohms_yt_from(pm, i)
        constraint_ohms_yt_to(pm, i)

        constraint_voltage_angle_difference(pm, i)

        constraint_thermal_limit_from(pm, i)
        constraint_thermal_limit_to(pm, i)
    end
end
```

Infrastructure Aware JuMP-like Modeling

Variable Declaration

Objective Function

Component-wise Constraints

Constraints

On to the Jupiter notebook!

https://github.com/lanl-ansi/tutorial-grid-science-2019