# Separation of Data, Models and Algorithms

EPICS Tutorial – Part 2

30 Jan 2026 Newcastle

# optimisation problems

- minimise cost (function)
- subject to
  - mathematical models for entities
    - laws of physics
    - technological envelopes
    - financials
      - CAPEX
      - OPEX

objective

constraint set:
linear and nonlinear
equalities and inequalities

# why *mathematical* optimisation?

- without ***structure***, the best approach to solve an optimisation problem is to enumerate all candidate solutions
  - evaluate, score, compare, pick best
- with - mathematical - ***structure***, things can ‘simplify’ a lot

# mathematical structure

- linear
- polynomial
- set inclusion
- integrality
- logical conditions

$$Ax = b$$

$$x^2 + y^2 + z^2 - 2xyx - 1 = 0$$

$$x \in \mathbb{S}^n_+$$

$$x \in \{0, 1\}$$

$$(x \geq 2 \wedge y = 1) \vee (x \leq y \leq 2)$$

# mathematical optimisation

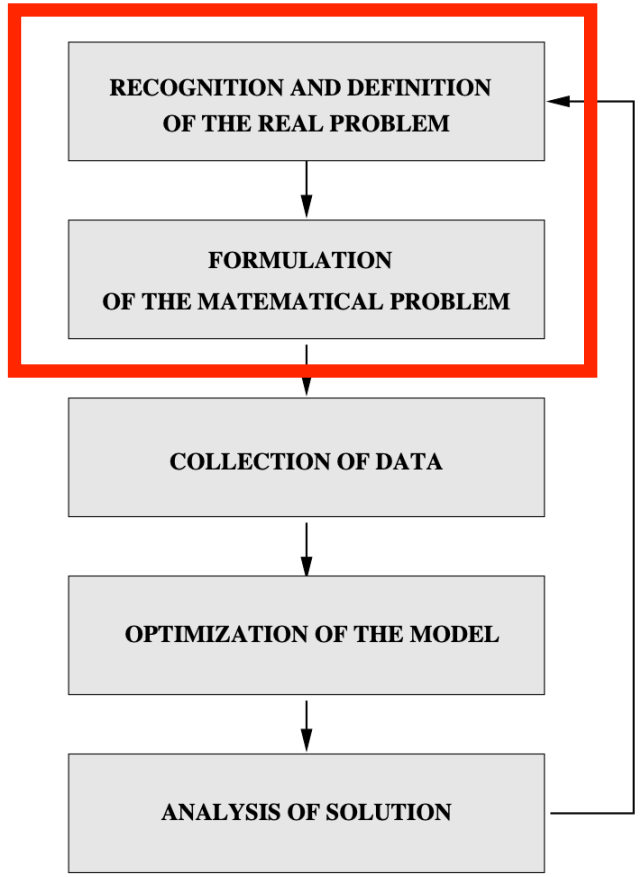
$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0 \\ & h_j(x) = 0 \end{aligned}$$

- decision variables (vector): $x \in \mathbb{R}^n$
- functions $f, g_i, h_j : \mathbb{R}^n \to \mathbb{R}$
  - what is their nature?
- # constraints and variables is typically finite $i \in \{1, \ldots, m\}, j \in \{1, \ldots, l\}$

# What is optimisation modelling?

# Optimisation problem modelling process

modeling task




good habit to separate

- model
- data
- solving

Fragni, E., & Gondzio, J. (1999). Optimization modeling languages.

# Mathematical model of a problem

- minimum cost flow problem
  - over a graph
    G = (V, E) = (vertices, edges)
  - V = {1,2, …n} (vertices)
  - source at vertex 1
  - sink at vertex n

$$\min_{x} \sum_{(i,j)\in E} c_{i,j}x_{i,j}$$

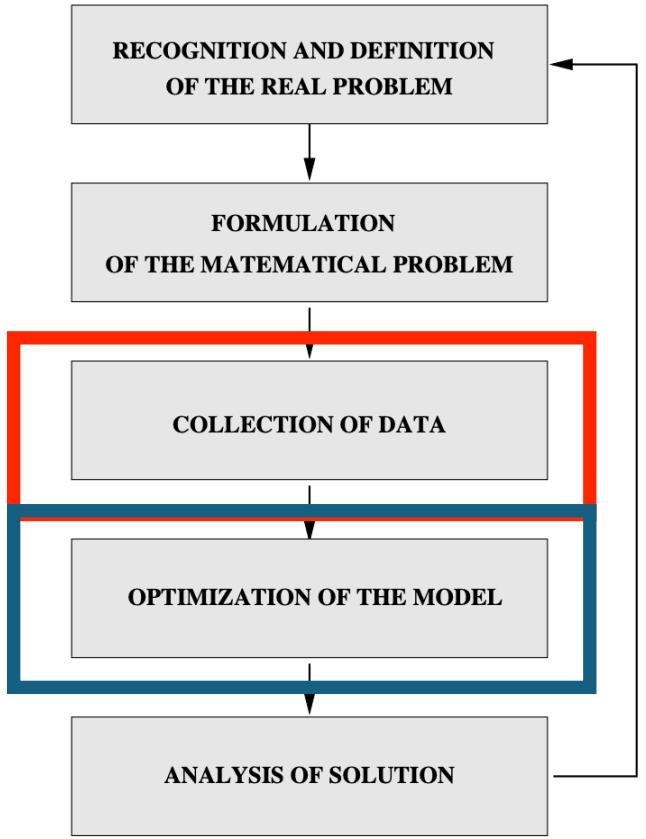
$$\text{s.t.} \sum_{(i,j)\in E} x_{i,j} = \sum_{(j,k)\in E} x_{j,k} \quad \forall j \in V \setminus \{1,n\}$$

$$\sum_{(i,n)\in E} x_{i,n} = 1$$

$$0 \le x_{i,j} \le C_{i,j} \quad \forall (i,j) \in E$$

Dunning, I., Huchette, J., & Lubin, M. (2015). JuMP: a modeling language for mathematical optimization, 1–21. Retrieved from http://arxiv.org/abs/1508.01982

# Optimisation problem modelling process




data interfacing

ask solver kindly

good habit to separate

- model
- data
- solving

Fragni, E., & Gondzio, J. (1999). Optimization modeling languages.

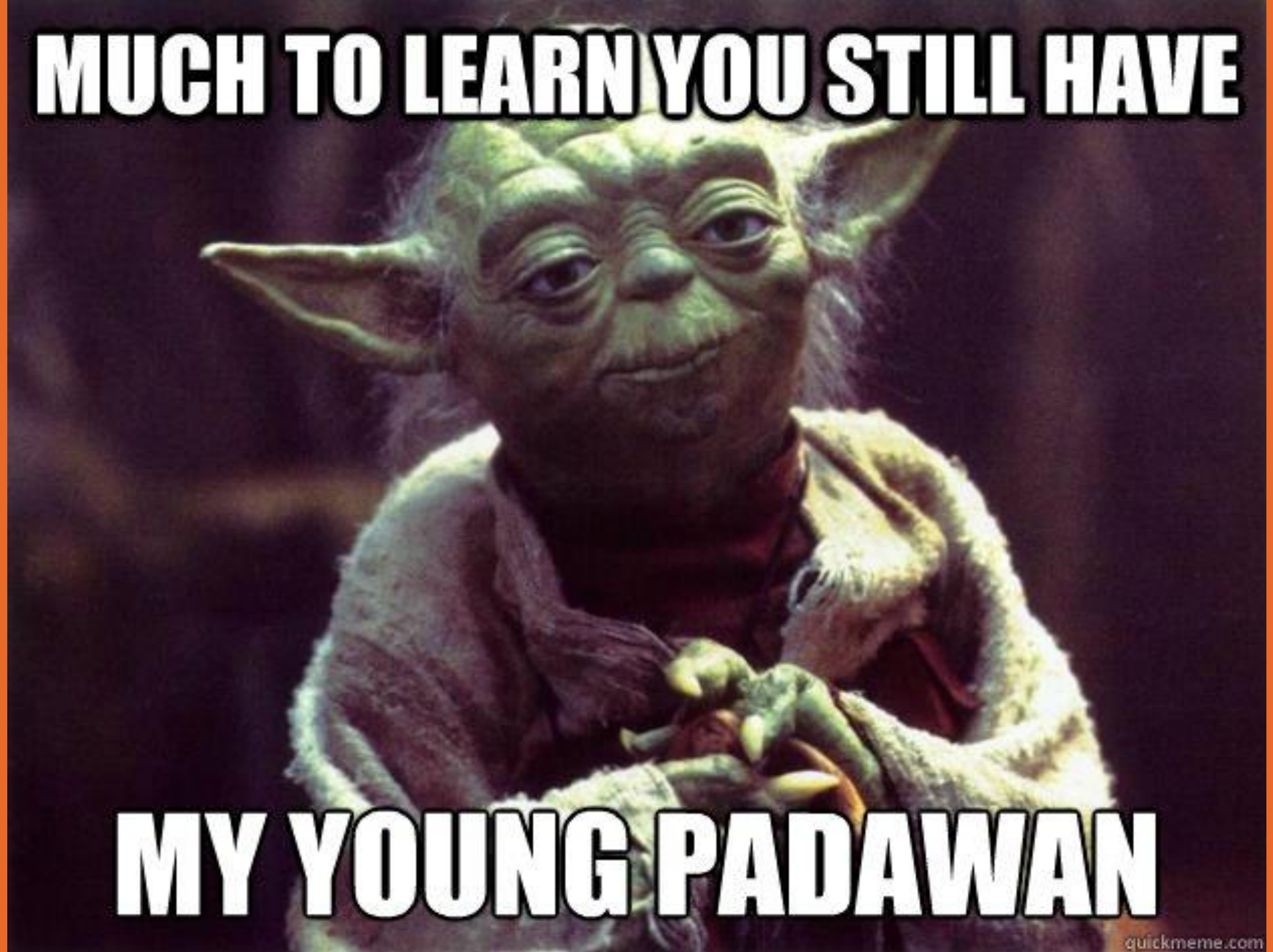
# declarative programming

- a style of building the structure and elements of computer programs
  - expresses the *logic of a computation* **without describing its control flow** (think: if then else)
  - describing what the program must accomplish in terms of the problem domain
  - **not**: describe how to accomplish it as a sequence of programming language primitives

# Learning JuMP by example


- James D. Foster

- Github: @jd-foster


MUCH TO LEARN YOU STILL HAVE
MY YOUNG PADAWAN
quickmeme.com

# A Modelling and Simulation Computation Process

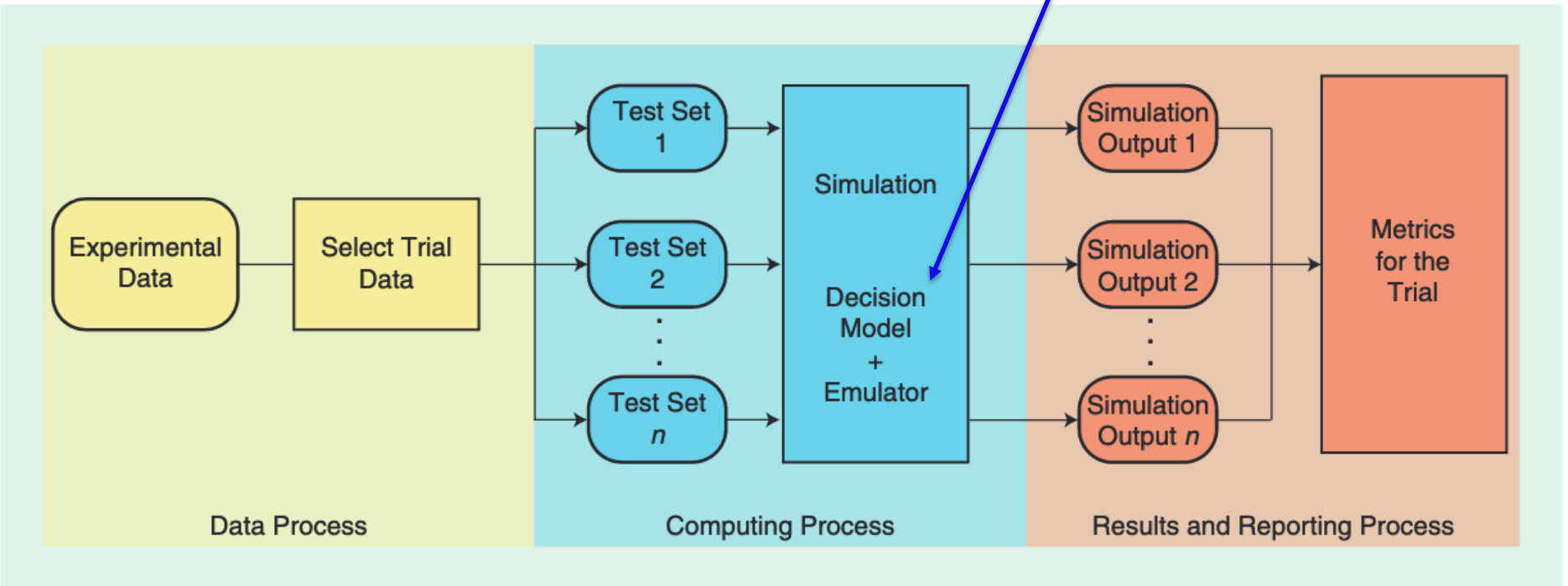
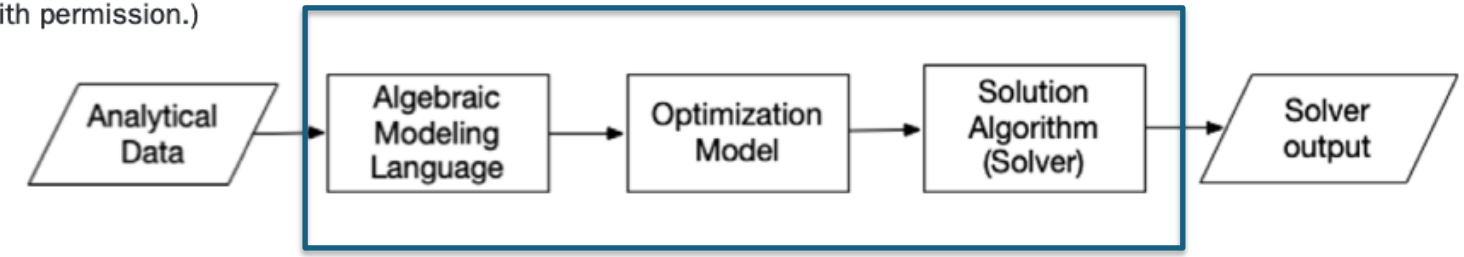



**Figure 1.** The computing steps required for conducting a single trial in a simulation experiment. These steps follow the best practices in scientific computing to achieve reproducibility and validation: data, computational modeling, and results subprocesses. (Adapted from Lara et al. 2020; used with permission.)



# What are Algebraic Modelling Languages

… such as JuMP?

- They are high-level **domain-specific languages** for succinctly describing decision models.
- Use algebra and mathematical functions to create ***software*** models that correspond well to your ***mental*** model.
- Their strength is that they separate the **descriptive** formulation of a model from the **process** of determining a solution or optimal decision variables.
- The **model formulation** stage involves defining decision variables and algebraic constraints on those variables.
- Good for decision-making with **simultaneous** (and conflicting) requirements.
- They mostly **automate** any data transformations that are necessary.
- They can be used to create **readable, reproducible and easily maintainable components** in a larger model or simulation.

# Algebraic Modelling Components

- Objective function: a “benefit” to maximise, or a “cost” to minimise
  - Maximise throughput, maximise expected value, minimise waste, minimise deviation

- Sets (for indexing or organising model components)
  - e.g. lists: {1, 2, 3, …}
  - key strings: {“Factory1”, “Retailer5”,…}

- Parameters (fixed data)
  - e.g. transfer matrix: A[i,j]
  - key-value stores (dictionaries): Capacity[“Factory1”,”Product3”].

- Decision variables (the “unknowns” to solve for)
  - e.g. Production levels: Produced[i]

- Constraints (equations and inequalities)
  - Implicitly describe the allowable “space” of decisions.

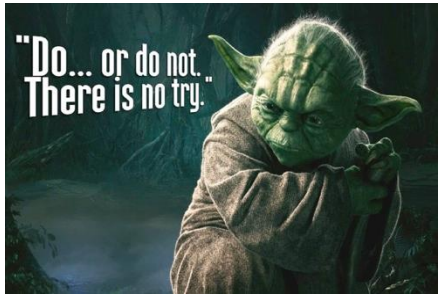
# The classic knapsack problem

$$\max \sum_{i=1}^{n} c_i x_i$$

$$\text{s.t.} \sum_{i=1}^{n} w_i x_i \leq b$$

$$x_i \in \{0, 1\} \quad \forall i = 1, \ldots, n.$$

maximise a linear combination of benefits (“profit”) *provided* the weighted sum is within budget *and* an item can be either included (1) or excluded (0).



# The classic knapsack problem

$$x_i \in \{0, 1\} \quad \forall i = 1, \ldots, n,$$

$$\max \sum_{i=1}^{n} c_i x_i$$

$$\text{s.t.} \sum_{i=1}^{n} w_i x_i \leq b$$

```
using JuMP   # modelling language package
import HiGHS  # solver interface

function example_knapsack(; verbose = true)
    profit = [5, 3, 2, 7, 4]
    weight = [2, 8, 4, 2, 5]
    capacity = 10

    model = Model(HiGHS.Optimizer)

    @variable(model, x[1:5], binary = true)

    # Objective: maximise profit
    @objective(model, Max, profit' * x)

    # Constraint: can carry all
    @constraint(model, weight' * x <= capacity)

    # Solve problem using MIP solver
    optimize!(model)
end
example_knapsack()
```

# Finding solutions (I)

- The software implementation of the solution algorithm is called a **solver**.
  - Delegation of responsibility for numerical implementation, precision, tolerances and so on.
  - Can be “tuned” to a particular use case by adjust many algorithmic parameters.
- The AML “compiles” a model down to the exact data inputs that a solver needs in order to run.

# Finding solutions (II)

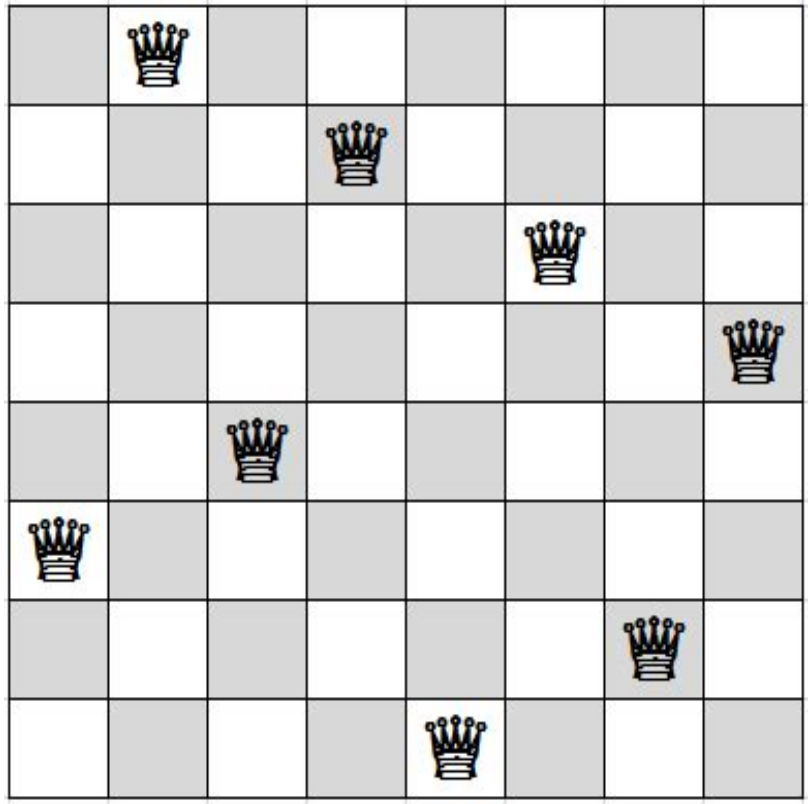
- Access powerful solution algorithms with **interfaces**
  - The user describes the data to the algorithm, but leaves the solution process to the algorithm.
  - A key reason of the abstraction for algebraic modelling languages is to be solver-independent: you can change to solver to a **different, more effective** solver if need be.
  - The translation interface between AML and solver must be specified.
  - There is a lot of “intelligence” applied to pre-processing the algebraic description:
  - Worry less about coming up with the “perfect” way to express the model
  - Work with what feels more natural.
- Writing modelling extensions
  - Two main approaches are new algebraic forms, or the user-defined functions derived within a general purpose programming language.
  - But, by *giving up* some of the full generality of a programming language function calls, you get highly targeted solution algorithms for your problem class.

# A tour of examples in the JuMP docs

- Problem types
  - Combinatorial
  - Continuous
- Applications
- Recent additions

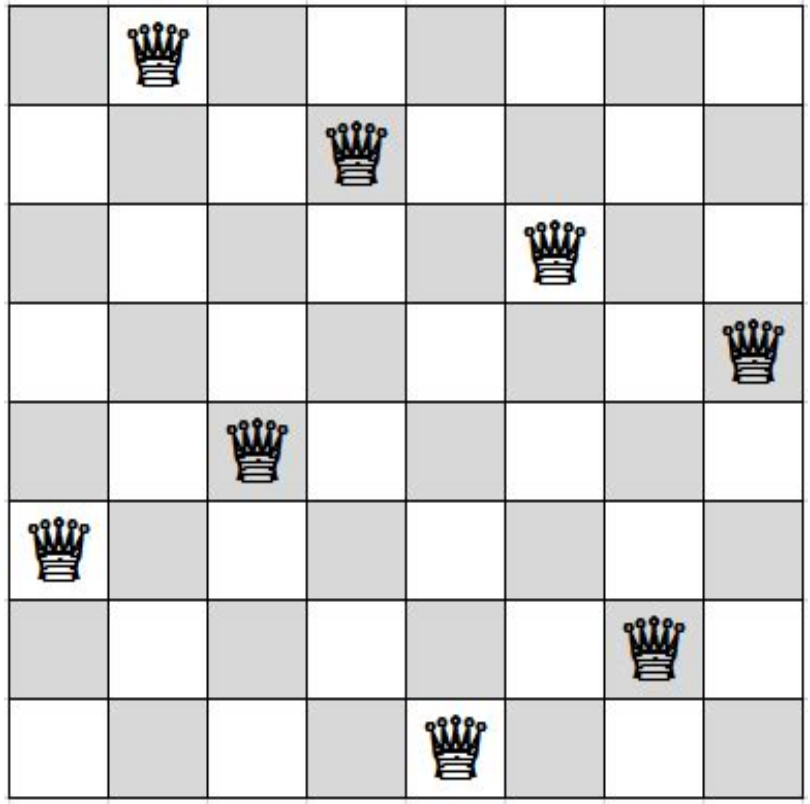
# N-Queens

```
using JuMP
import HiGHS
import LinearAlgebra

N = 8
model = Model(HiGHS.Optimizer)

@variable(model, x[1:N, 1:N], binary = true);
# There must be exactly one queen in a given row/column:
for i in 1:N
    @constraint(model, sum(x[i, :]) == 1)
    @constraint(model, sum(x[:, i]) == 1)
end
# There can only be one queen on any given diagonal:
for i in -(N - 1):(N-1)
    @constraint(model, sum(diag(x, i)) <= 1)
    @constraint(model, sum(diag(reverse(x; dims = 1), i)) <= 1)
end
optimize!(model)
```



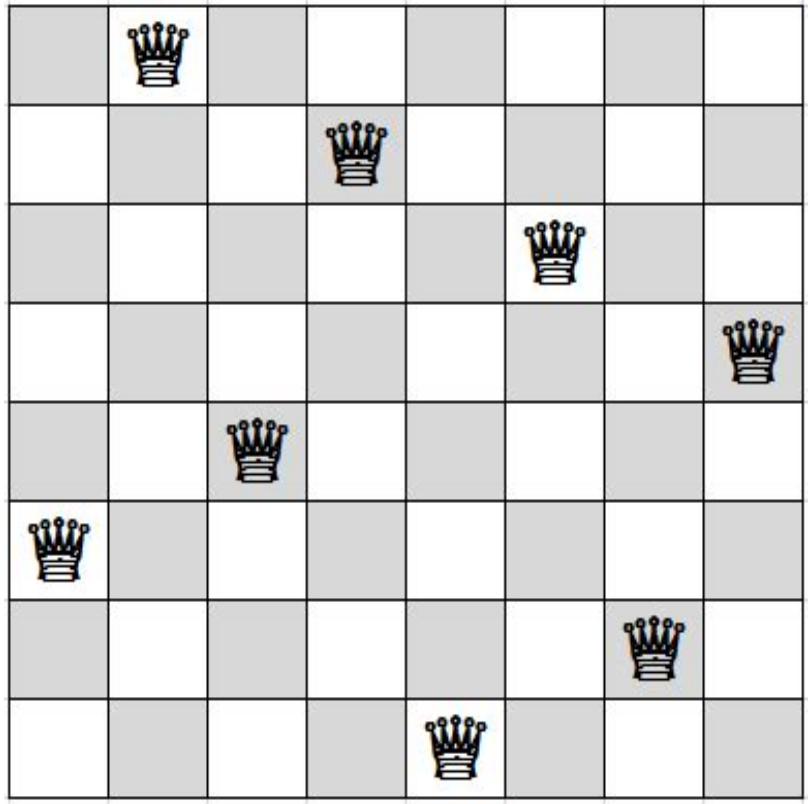
# N-Queens

```
julia> solution = round.(Int, value.(x))

8×8 Matrix{Int64}:
 0  0  0  0  1  0  0  0
 0  0  1  0  0  0  0  0
 0  0  0  0  0  0  0  1
 0  0  0  1  0  0  0  0
 0  0  0  0  0  0  1  0
 1  0  0  0  0  0  0  0
 0  0  0  0  0  1  0  0
 0  1  0  0  0  0  0  0
```



# N-Queens

```
julia> solution = round.(Int, value.(x))

8×8 Matrix{Int64}:
 0  0  0  0  1  0  0  0
 0  0  1  0  0  0  0  0
 0  0  0  0  0  0  0  1
 0  0  0  1  0  0  0  0
 0  0  0  0  0  0  1  0
 1  0  0  0  0  0  0  0
 0  0  0  0  0  1  0  0
 0  1  0  0  0  0  0  0
```



# Sudoku

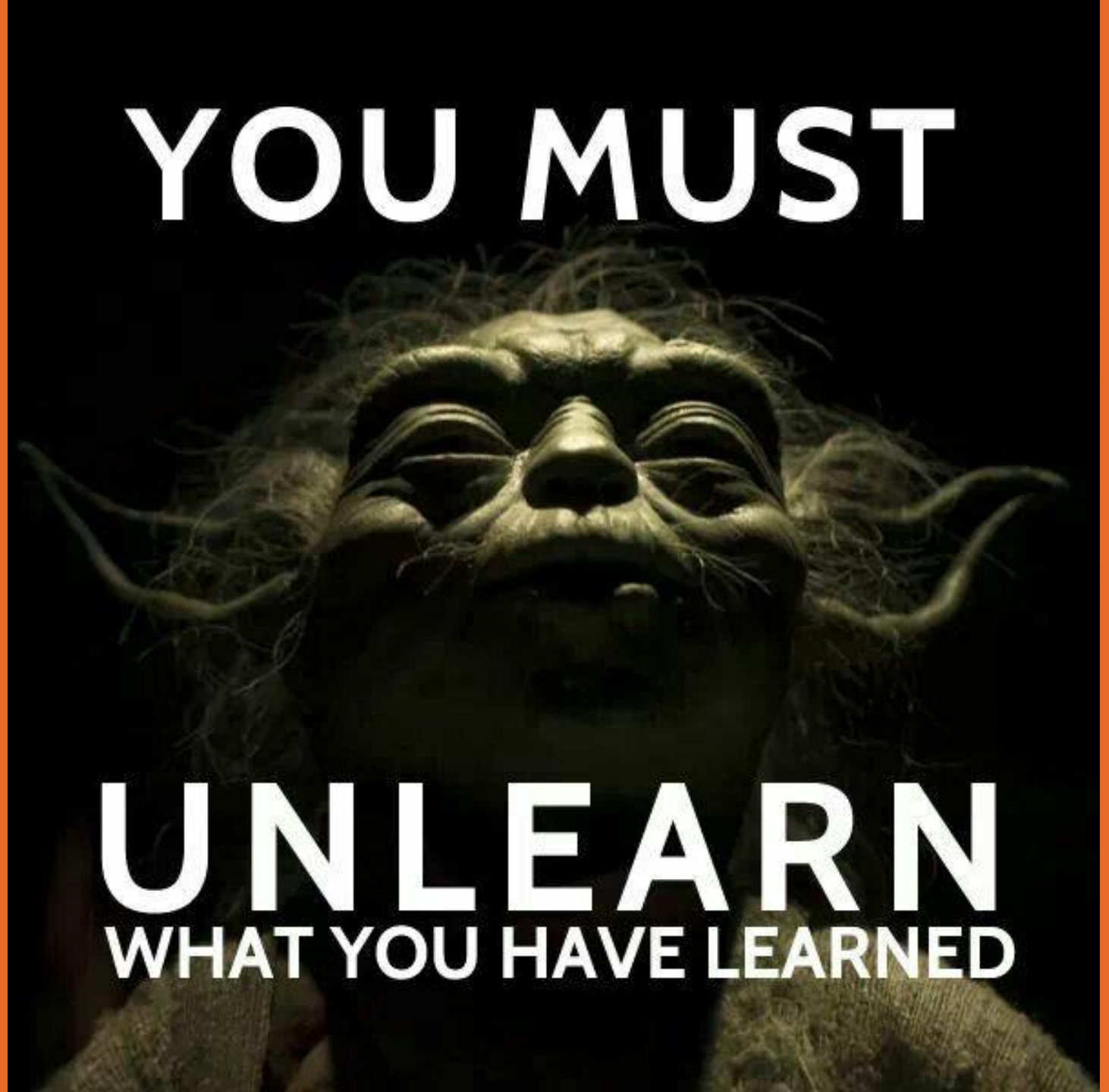
```
using JuMP
using HiGHS

sudoku = Model(HiGHS.Optimizer)
@variable(sudoku, x[i = 1:9, j = 1:9, k = 1:9], binary = true)

for ind in 1:9  # Each row, OR each column
    for k in 1:9  # Each digit
      # Sum across columns (j) - row constraint
      @constraint(sudoku, sum(x[ind, j, k] for j in 1:9) == 1)
      # Sum across rows (i) - column constraint
      @constraint(sudoku, sum(x[i, ind, k] for i in 1:9) == 1)
    end
end

# Enforce the constraint that each digit appears
# once in each of the nine 3x3 sub-grids

# i is the top left row, j is the top left column.
for i in 1:3:7
    for j in 1:3:7
        for k in 1:9
     # We'll sum from i to i+2, for example, i=4, r=4, 5, 6.
     @constraint(sudoku,
       sum(x[r, c, k] for r in i:(i+2), c in j:(j+2)) == 1)
        end
    end
end
```

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | | | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

# Sudoku

```
init_sol = [
    5 3 0 0 7 0 0 0 0
    6 0 0 1 9 5 0 0 0
    0 9 8 0 0 0 0 6 0
    8 0 0 0 6 0 0 0 3
    4 0 0 8 0 3 0 0 1
    7 0 0 0 2 0 0 0 6
    0 6 0 0 0 0 2 8 0
    0 0 0 4 1 9 0 0 5
    0 0 0 0 8 0 0 7 9
]

for i in 1:9
    for j in 1:9
        # If the space isn't empty
        if init_sol[i, j] != 0
# Then the corresponding variable for that digit and location
must be 1.
          fix(x[i, j, init_sol[i, j]], 1; force = true)
        end
    end
end

optimize!(sudoku)
```

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 4 | 6 | 7 | 8 | 9 | 1 | 2 |
| 6 | 7 | 2 | 1 | 9 | 5 | 3 | 4 | 8 |
| 1 | 9 | 8 | 3 | 4 | 2 | 5 | 6 | 7 |
| 8 | 5 | 9 | 7 | 6 | 1 | 4 | 2 | 3 |
| 4 | 2 | 6 | 8 | 5 | 3 | 7 | 9 | 1 |
| 7 | 1 | 3 | 9 | 2 | 4 | 8 | 5 | 6 |
| 9 | 6 | 1 | 5 | 3 | 7 | 2 | 8 | 4 |
| 2 | 8 | 7 | 4 | 1 | 9 | 6 | 3 | 5 |
| 3 | 4 | 5 | 2 | 8 | 6 | 1 | 7 | 9 |


YOU MUST
UNLEARN
WHAT YOU HAVE LEARNED

# Constraint programming toolkit

- AllDifferent
  - every element in a list takes a different integer value.
- BinPacking
  - divide up a set of items into different groups, such that the sum of their weights does not exceed the capacity of a bin.
- Circuit
  - construct a *tour* of a list of N variables
- CountAtLeast
  - at least n elements in a set of variables belong to a set of values.
- CountBelongs
  - count how many elements in a set of variables belong to a set of values.
- CountDistinct
  - count the number of *distinct* elements in a set of variables.
- CountGreaterThan
  - strictly upper-bound the number of distinct elements in a set of variables that have a value equal to another variable.
- Table
  - constrain a vector to exactly match one of the rows in a table.

# Sudoku:
# constraint programming and the *all-different* constraint

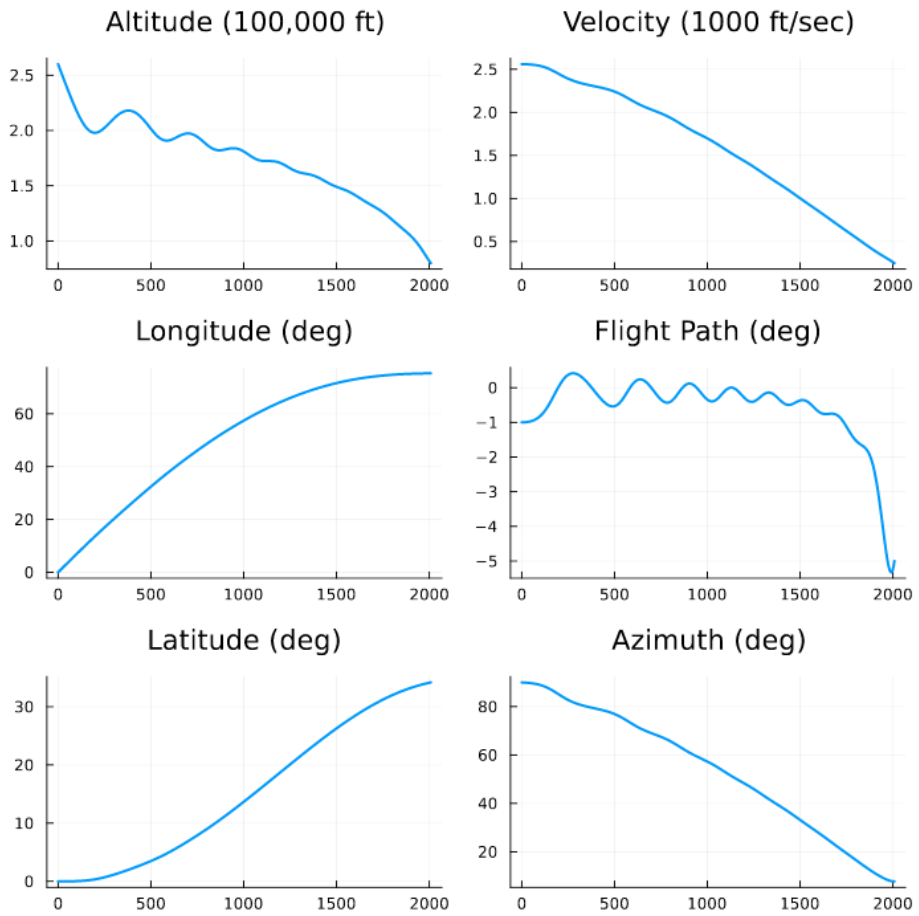
```
model = Model(HiGHS.Optimizer)

@variable(model, 1 <= x[1:9, 1:9] <= 9, Int);

@constraint(model, [i = 1:9],
                x[i, :] in MOI.AllDifferent(9));

@constraint(model, [j = 1:9],
                x[:, j] in MOI.AllDifferent(9));

for i in (0, 3, 6), j in (0, 3, 6)
    @constraint(model,
  vec(x[i.+(1:3), j.+(1:3)]) in MOI.AllDifferent(9))
end
```

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 4 | 6 | 7 | 8 | 9 | 1 | 2 |
| 6 | 7 | 2 | 1 | 9 | 5 | 3 | 4 | 8 |
| 1 | 9 | 8 | 3 | 4 | 2 | 5 | 6 | 7 |
| 8 | 5 | 9 | 7 | 6 | 1 | 4 | 2 | 3 |
| 4 | 2 | 6 | 8 | 5 | 3 | 7 | 9 | 1 |
| 7 | 1 | 3 | 9 | 2 | 4 | 8 | 5 | 6 |
| 9 | 6 | 1 | 5 | 3 | 7 | 2 | 8 | 4 |
| 2 | 8 | 7 | 4 | 1 | 9 | 6 | 3 | 5 |
| 3 | 4 | 5 | 2 | 8 | 6 | 1 | 7 | 9 |

Including automatic reformulation to the linear programming solver.

# Optimal control for a Space Shuttle reentry trajectory

```
using JuMP
import Interpolations
import Ipopt
…
# Aerodynamic and atmospheric forces on the vehicle
const ρ₀ = 0.002378
const hᵣ = 23800.0
const Rₑ = 20902900.0
const μ = 0.14076539e17
const S = 2690.0
const a₀ = -0.20704
…

model = Model(optimizer_with_attributes(Ipopt.Optimizer, user_options...))

@variables(model, begin
    0 ≤ scaled_h[1:n]                       # altitude (ft) / 1e5
    ϕ[1:n]                                  # longitude (rad)
    deg2rad(-89) ≤ θ[1:n] ≤ deg2rad(89)     # latitude (rad)
    1e-4 ≤ scaled_v[1:n]                    # velocity (ft/sec) / 1e4
    deg2rad(-89) ≤ γ[1:n] ≤ deg2rad(89)     # flight path angle (rad)
    ψ[1:n]                  # azimuth (rad)
    deg2rad(-90) ≤ α[1:n] ≤ deg2rad(90)     # angle of attack (rad)
    deg2rad(-89) ≤ β[1:n] ≤ deg2rad(1)      # bank angle (rad)
    #         3.5 ≤       Δt[1:n] ≤ 4.5     # time step (sec)
    Δt[1:n] == 4.0                          # time step (sec)
end);
…
```

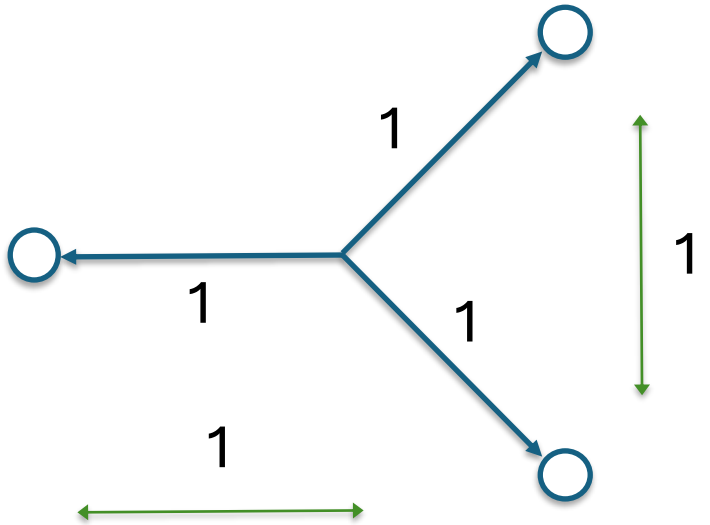


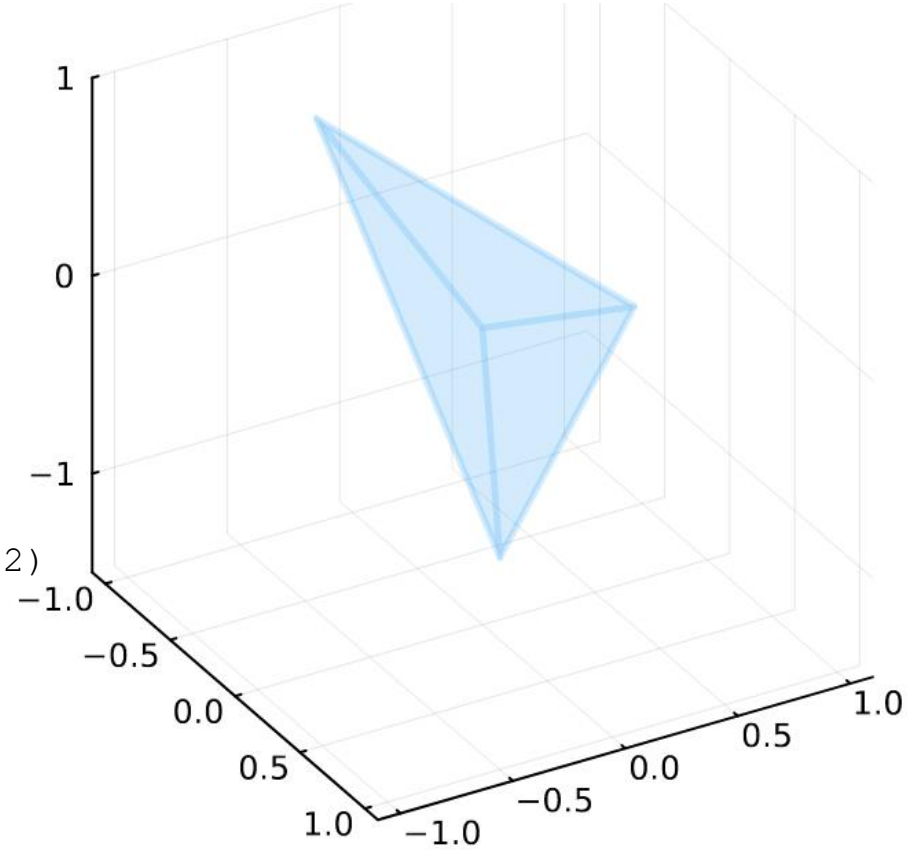
# Simple semidefinite programming (SDP) examples

- Maximum cut via SDP
- K-means clustering via SDP
- The correlation problem
- The minimum distortion problem
- Lovász numbers
- Robust uncertainty sets

# Simple semidefinite programming examples

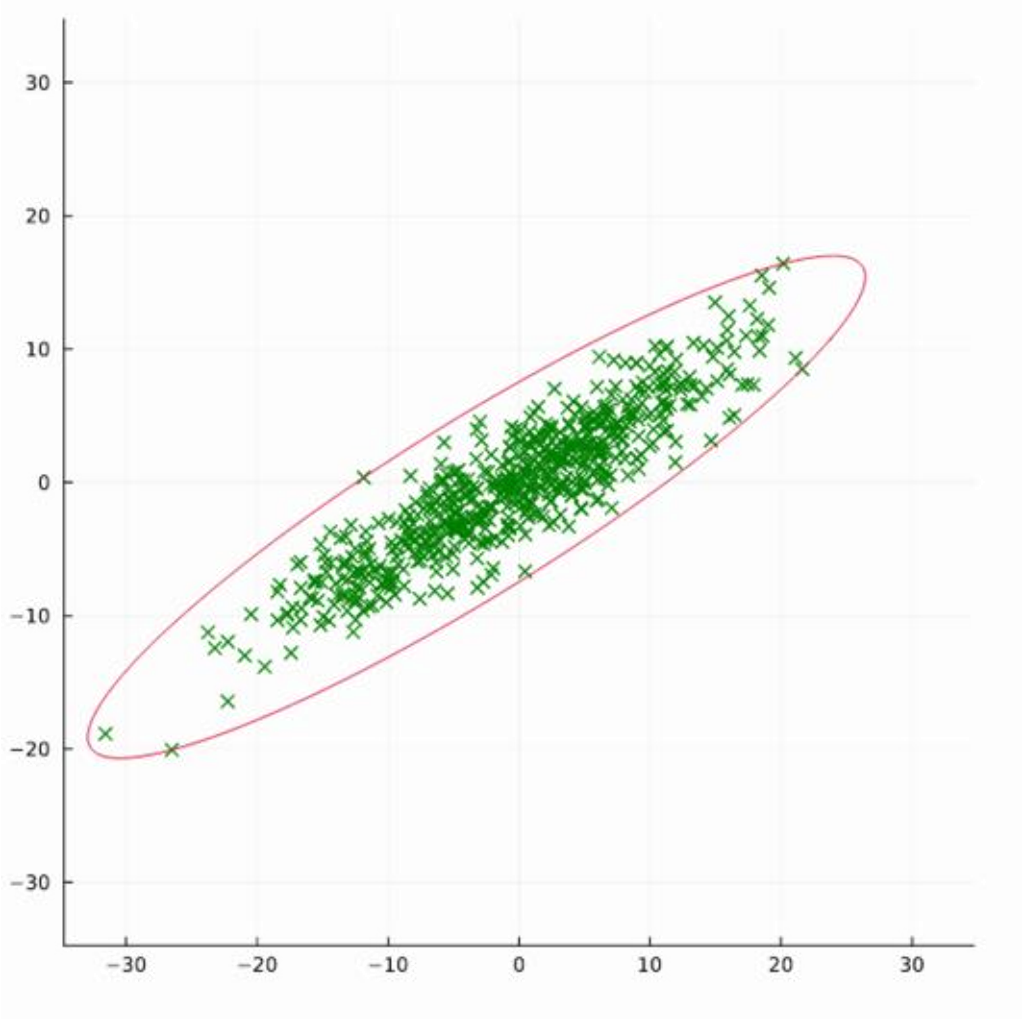
```
model = Model(SCS.Optimizer)

D = [
    0.0 1.0 1.0 1.0
    1.0 0.0 2.0 2.0
    1.0 2.0 0.0 2.0
    1.0 2.0 2.0 0.0
]

@variable(model, c² >= 1.0)
@variable(model, Q[1:4, 1:4], PSD)

for i in 1:4, j in (i+1):4
    @constraint(model, D[i, j]^2 <= Q[i, i] + Q[j, j] - 2 * Q[i, j])
    @constraint(model, Q[i, i] + Q[j, j] - 2 * Q[i, j] <= c² * D[i, j]^2)
end

fix(Q[1, 1], 0)
@objective(model, Min, c²)
optimize!(model)

Test.@test objective_value(model) ≈ 4 / 3 atol = 1e-4

# Recover the minimal distorted embedding:
X = [zeros(3) sqrt(value.(Q)[2:end, 2:end])]
```



# Ellipsoid approximation

```
model = Model(SCS.Optimizer)

S = generate_point_cloud(600);

m, n = size(S)

@variable(model, z[1:n])
@variable(model, Z[1:n, 1:n], PSD)
@variable(model, s)
@variable(model, t)

@constraint(model, [s z'; z Z] >= 0, PSDCone())
@constraint(
    model,
    [i in 1:m],
    S[i, :]' * Z * S[i, :] - 2 * S[i, :]' * z + s <= 1,
)
@constraint(model,
   [t; vec(Z)] in MOI.RootDetConeSquare(n))

@objective(model, Max, t)

optimize!(model)
```



# Solving the dual language problem

## Dualization

The purpose of this tutorial is to explain how to use Dualization.jl to improve the performance of some conic optimization models. There are two important takeaways:

1. JuMP reformulates problems to meet the input requirements of the solver, potentially increasing the problem size by adding slack variables and constraints.
2. Solving the dual of a conic model can be more efficient than solving the primal.

Dualization.jl is a package which fixes these problems, allowing you to solve the dual instead of the primal with a one-line change to your code.

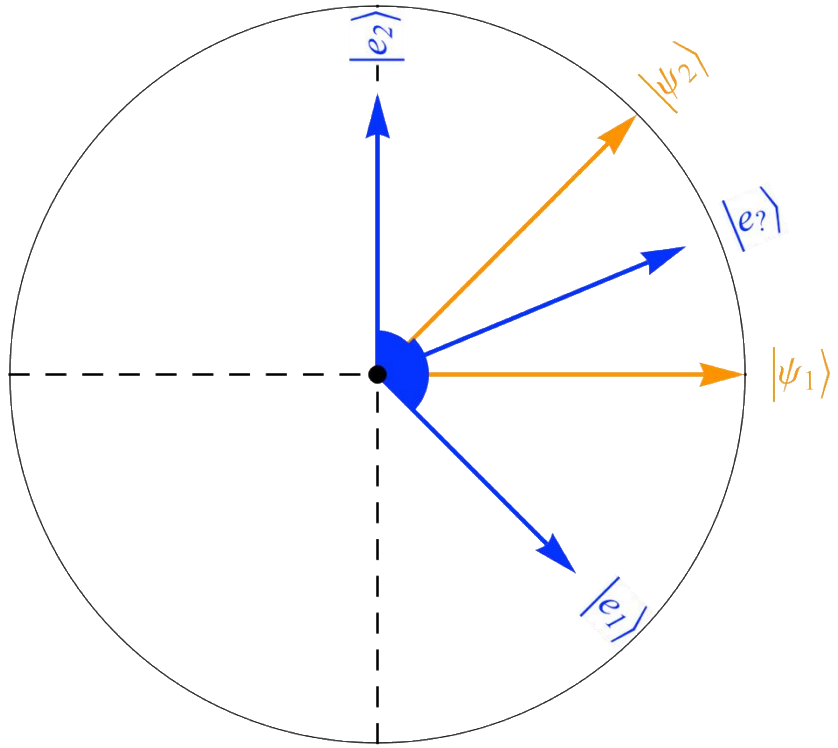
This tutorial uses the following packages

```
using JuMP
import Dualization
import SCS
```

| Model | #PrimalVars | #Cons | #DualVars | #DualCons |
|---|---|---|---|---|
| **Maximum cut** | 10 | 5 | 4 | 1 |
| **K-means** | 21 | 29 | 28 | 22 |
| **Correlation** | 6 | 8 | 7 | 5 |
| **Minimum distortion** | 11 | 15 | 14 | 15 |
| **Theta** | 15 | 7 | 6 | 1 |
| **Robust** | 12 | 4 | 24 | 10 |

# Quantum state discrimination

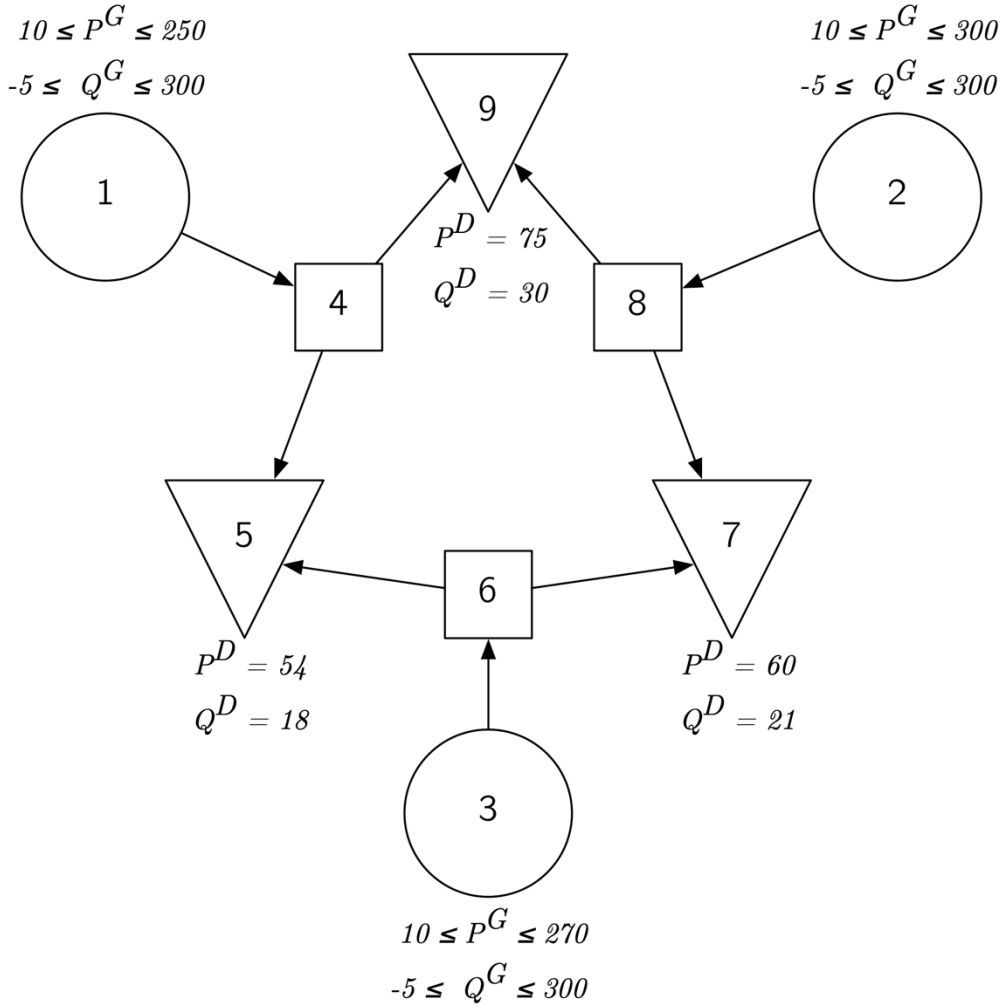
```
using JuMP
import LinearAlgebra
import SCS

function random_state(d)
    x = randn(ComplexF64, (d, d))
    y = x * x'
    return LinearAlgebra.Hermitian(y / LinearAlgebra.tr(y))
end

N, d = 2, 2

ρ = [random_state(d) for i in 1:N]

model = Model(SCS.Optimizer)
E = [@variable(model, [1:d, 1:d] in HermitianPSDCone()) for i in 1:N]
@constraint(model, sum(E) == LinearAlgebra.I)
@objective(
    model,
    Max,
    sum(real(LinearAlgebra.tr(ρ[i] * E[i])) for i in 1:N) / N,
)
optimize!(model)
```




https://commons.wikimedia.org/wiki/File:Unambiguous_quantum_state_discrimination_-_example.svg

# Optimal power flow

```
model = Model(Ipopt.Optimizer)

@variable(
    model,
    S_G[i in 1:N] in ComplexPlane(),
    lower_bound = P_Gen_lb[i] + Q_Gen_lb[i] * im,
    upper_bound = P_Gen_ub[i] + Q_Gen_ub[i] * im,
)

@variable(model,
  V[1:N] in ComplexPlane(), start = 1.0 + 0.0im)

@constraint(model, [i in 1:N], 0.9^2 <= real(V[i])^2 + imag(V[i])^2 <= 1.1^2)

@constraint(model, S_G - S_Demand .== V .* conj(Y * V))

P_G = real(S_G)
@objective(
    model,
    Min,
    (0.11 * P_G[1]^2 + 5 * P_G[1] + 150) +
    (0.085 * P_G[2]^2 + 1.2 * P_G[2] + 600) +
    (0.1225 * P_G[3]^2 + P_G[3] + 335),
);
optimize!(model)
```



# Nested optimization problems

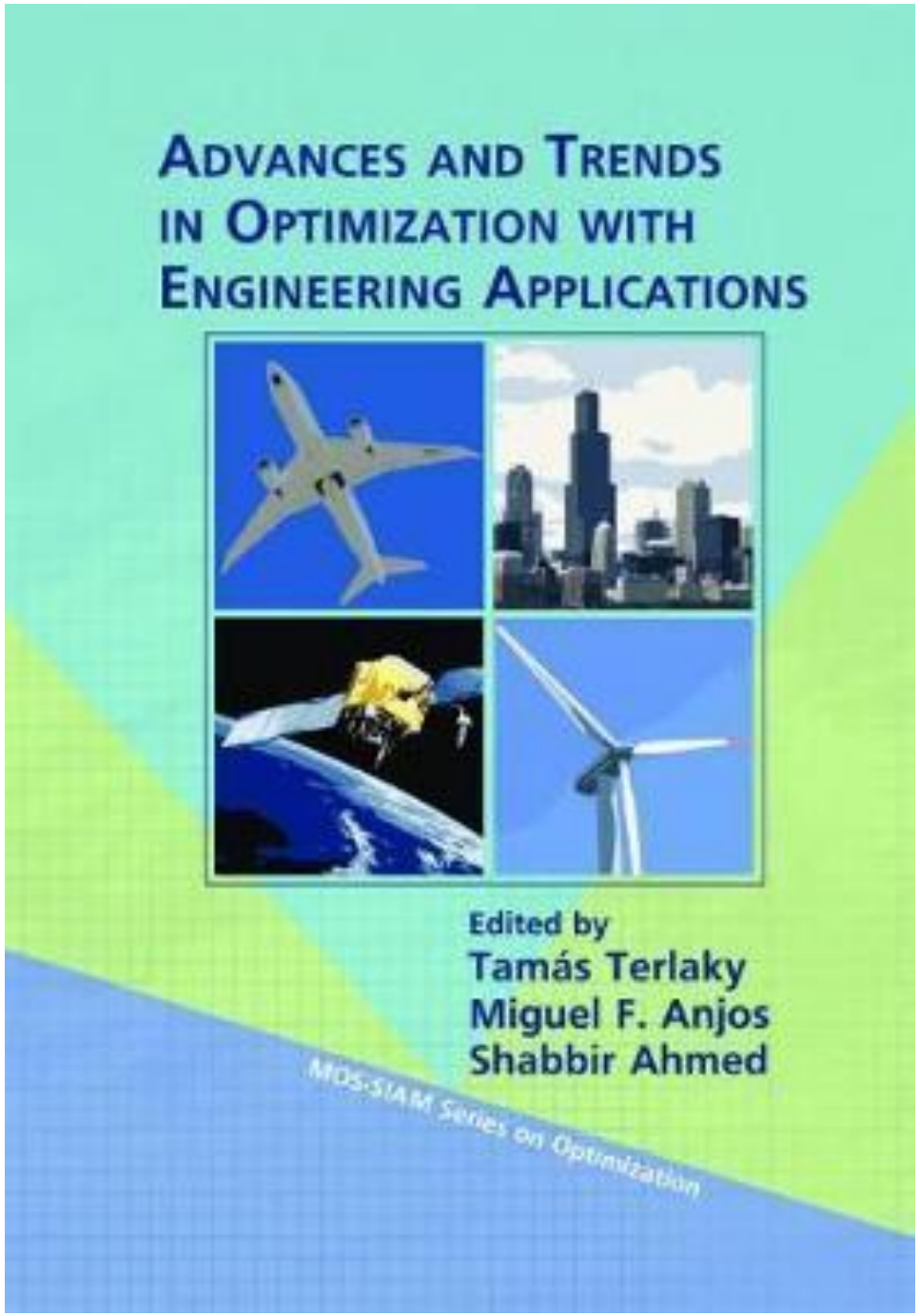
```
using JuMP
import Ipopt

function solve_lower_level(x...)
…
end

function V(x...)
    f, _ = solve_lower_level(x...)
    return f
end

function ∇V(g::AbstractVector, x...)
    _, y = solve_lower_level(x...)
    g[1] = 2 * x[1] * y[1] - y[1]^4
    g[2] = 2 * x[2] * y[2] - 2 * y[2]^4
    return
end


model = Model(Ipopt.Optimizer)

@variable(model, x[1:2] >= 0)

register(model, :V, 2, V, ∇V)

@NLobjective(model, Min, x[1]^2 + x[2]^2 + V(x[1], x[2]))
optimize!(model)
```

$$
\begin{aligned}
\min_{x,z} \quad & x_1^2 + x_2^2 + z \\
s.t. \quad & z = \max_{y} \quad x_1^2 y_1 + x_2^2 y_2 - x_1 y_1^4 - 2 x_2 y_2^4 \\
& \qquad s.t. \quad (y_1 - 10)^2 + (y_2 - 10)^2 \leq 25 \\
& x \geq 0.
\end{aligned}
$$

# Extensions

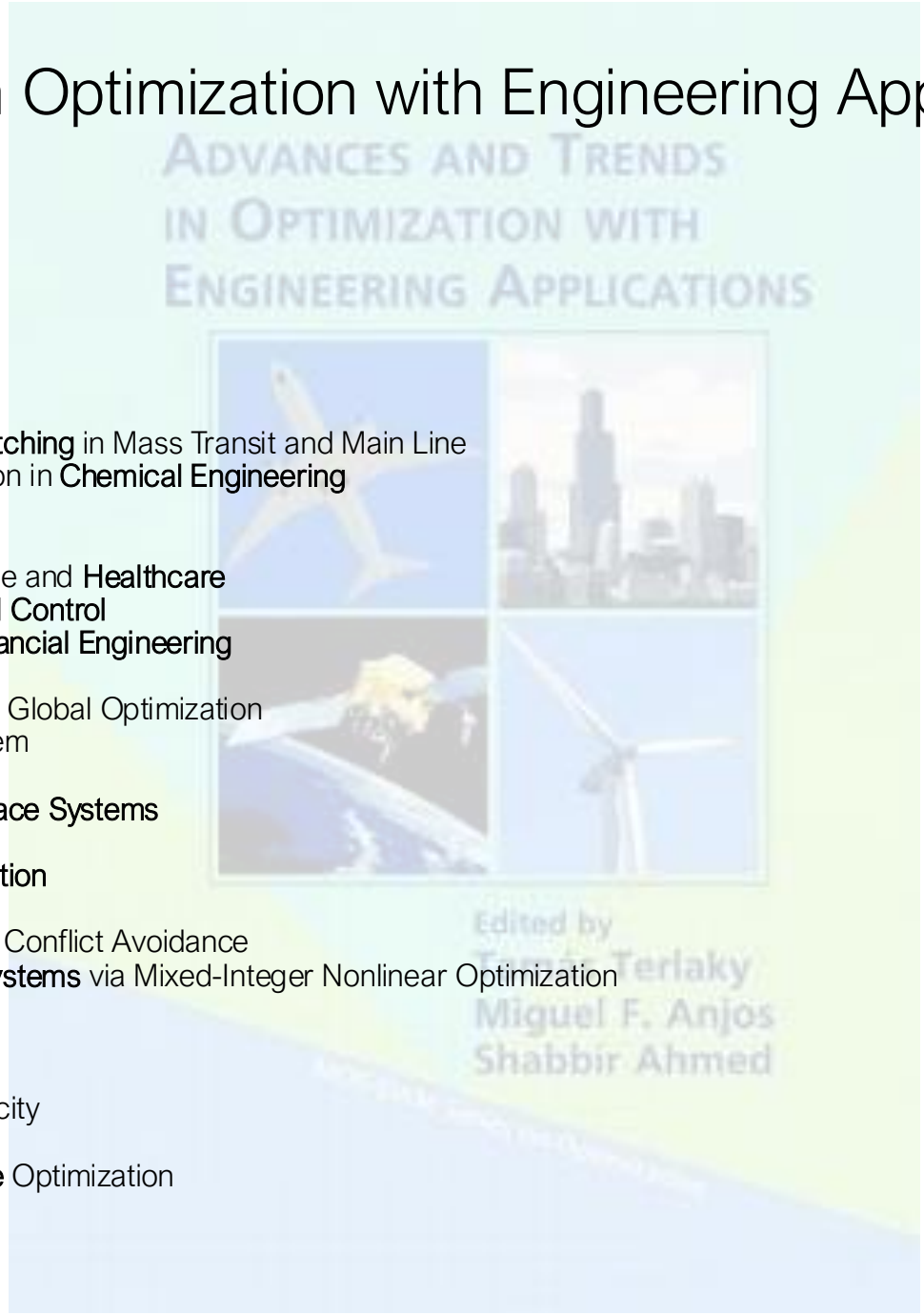
- MultiObjectiveAlgorithms.jl (MOA)
  - Algorithms for multi-objective optimisation.
- BilevelJuMP.jl
  - Model and solve bilevel optimisation problems.
- DiffOpt.jl
  - Differentiate through and then optimising over a set of structurally parameterised problems.
- ParametricOptInterface.jl
  - Efficiently modify key data and constants in an optimisation problem.
- PolyJuMP.jl
  - Formulate and solve problems involving polynomials as decision variables.
- SDDP.jl
  - A solver for multistage stochastic optimization problem.
- PiecewiseLinearOpt.jl
  - Model optimisation problems containing piecewise linear functions.

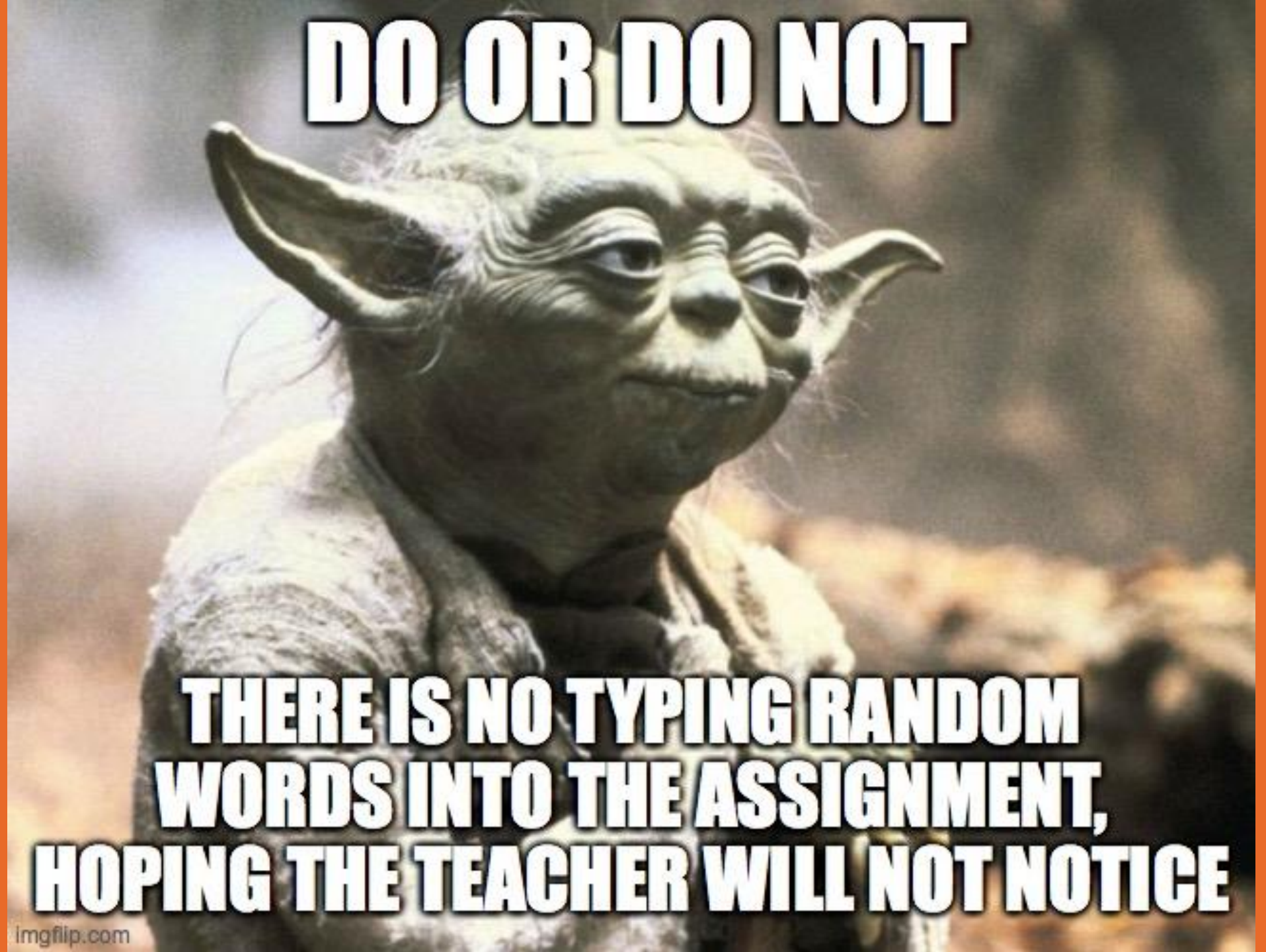
2017


ADVANCES AND TRENDS
IN OPTIMIZATION WITH
ENGINEERING APPLICATIONS
Edited by
Tamás Terlaky
Miguel F. Anjos
Shabbir Ahmed
MOS-SIAM Series on Optimization

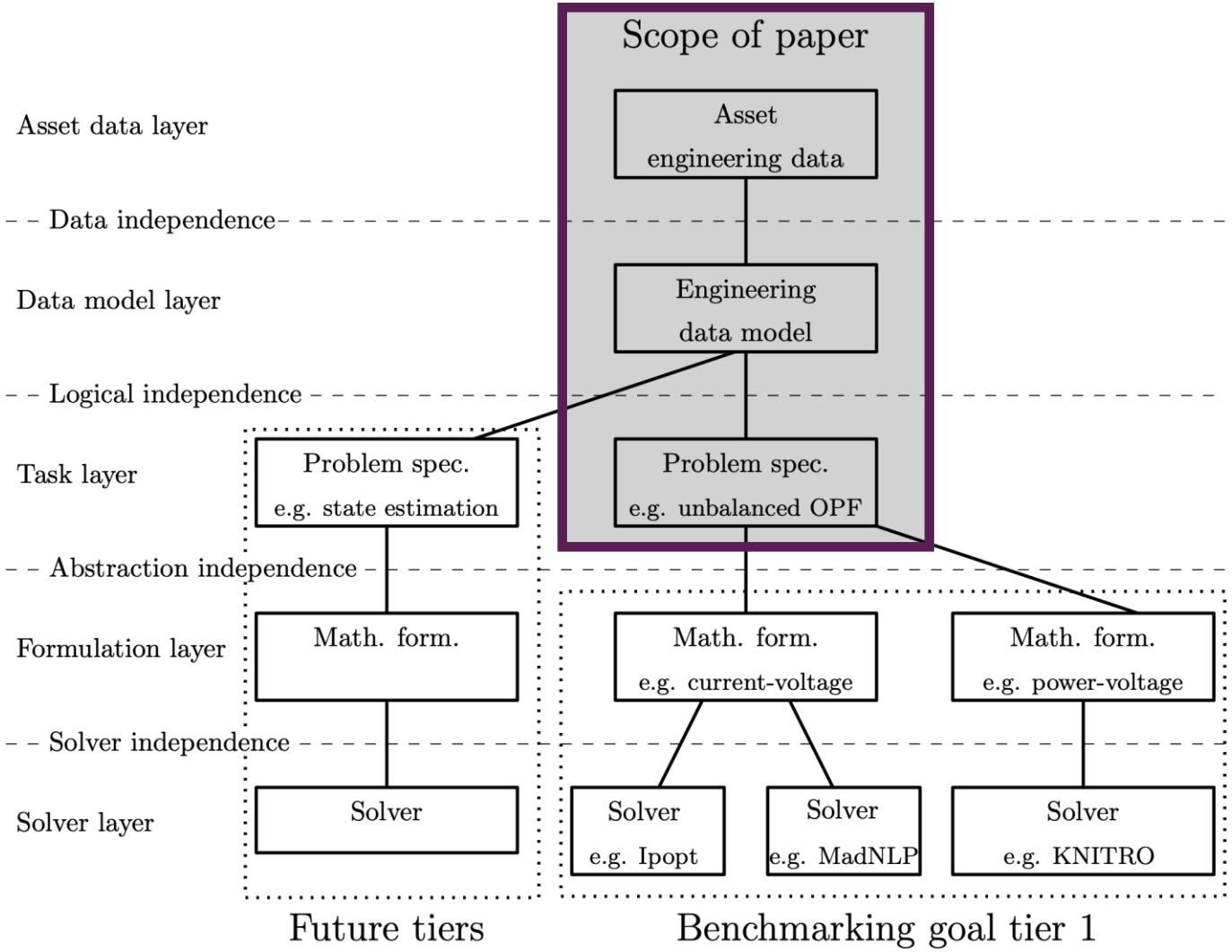
# Advances and Trends in Optimization with Engineering Applications

- **Truss Topology** Design by Linear Optimization
- Optimization in **Electrical Engineering**
  - Filter design
  - Pattern Classification
  - Information and Coding Theory
- Optimization in **Chemical Engineering**
  - Production Planning
  - Model Predictive Control
- Integer Optimization Techniques for **Train Dispatching** in Mass Transit and Main Line
- Applications of Mixed-Integer Linear Optimization in **Chemical Engineering**
  - LNG **Inventory Routing**
  - Chemical **Supply Network** Optimization
- Applications of Discrete Optimization in Medicine and **Healthcare**
- Conic Linear Optimization for Nonlinear **Optimal Control**
- Applications of Conic Linear Optimization in **Financial Engineering**
  - **Portfolio Optimization** Problems
- De Novo Design of **Protein-DNA Systems** Using Global Optimization
- Global Optimization: **Optimal Power Flow** Problem
- Optimization of **Distillation Systems**
- Multidisciplinary Design Optimization of **Aerospace Systems**
  - **Aerodynamic Shape** Optimization
  - Optimization of a **Satellite's Design and Operation**
- Nonlinear Optimization for **Building Automation**
- Optimization in **Air Traffic Management**: Aircraft Conflict Avoidance
- Short-Term Planning of **Cogeneration Energy Systems** via Mixed-Integer Nonlinear Optimization
- Robust Optimization in **Radiation Therapy**
  - Treatment Planning and Optimization
- Robust **Wind Farm Layout** Optimization
- On the **Marginal Value of Water** for Hydroelectricity
- Inventory and **Supply Chain** Optimization
- Methodologies and Software for **Derivative-Free** Optimization




DO OR DO NOT
THERE IS NO TYPING RANDOM WORDS INTO THE ASSIGNMENT, HOPING THE TEACHER WILL NOT NOTICE
imgflip.com

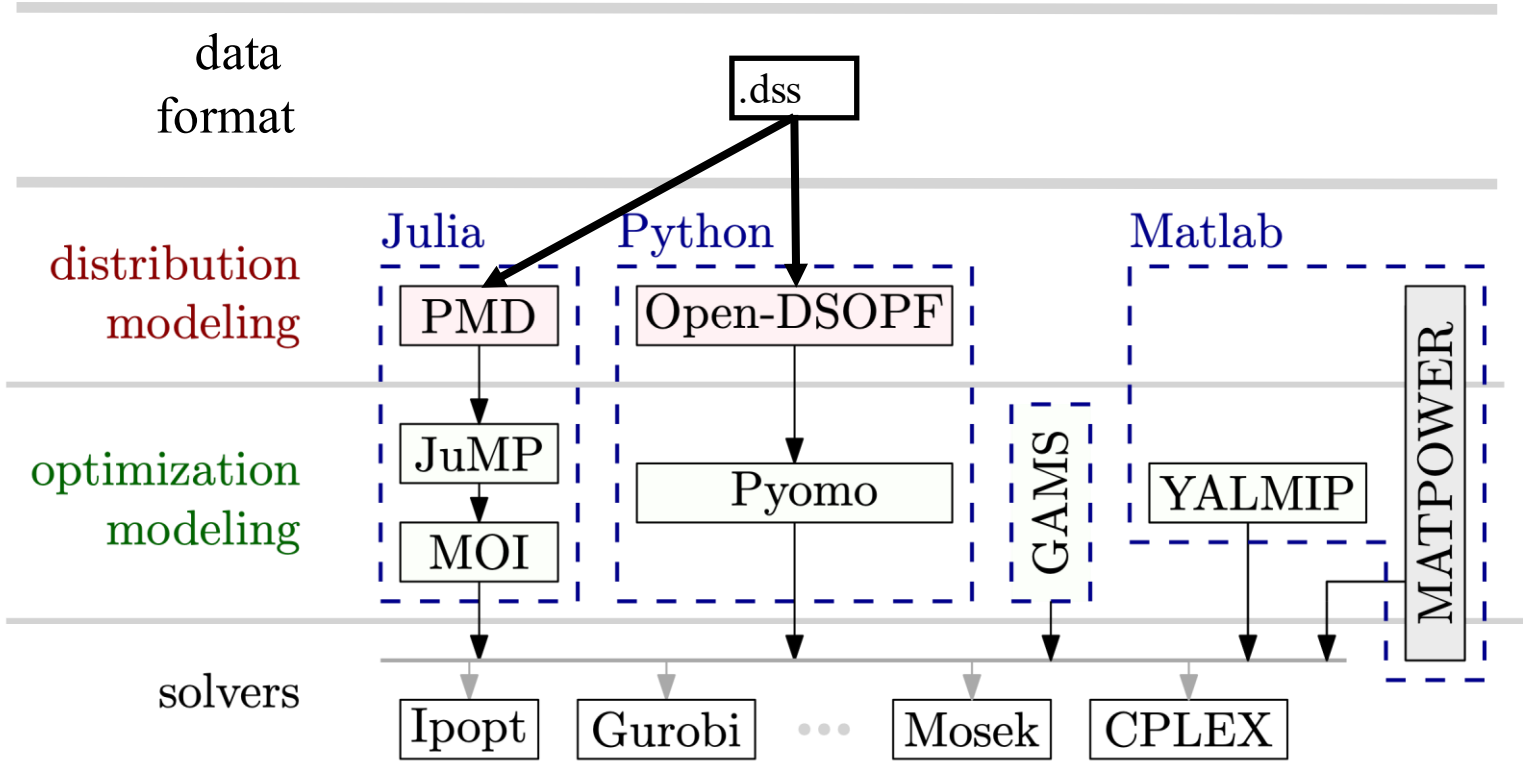
# Benchmarking unbalanced OPF




- Data *models* to represent electrical components in an unbalanced way
  - Modelling principles that avoid mistakes of the past
  - Easier to discuss properties of physical objects than math parameters
- Data *formats* that are easy to parse as a researcher

# Optimization-focused network data format

**.dss was a workaround**



# PowerModels OPF Problem Definition

- formulation of problem from academic paper

**variables:**

$$S_i^g \ \ \forall i \in N$$
$$V_i \ \ \forall i \in N$$

**minimize:**

$$\sum_{i \in N} f(S_i^g)$$

**subject to:**

$$(\boldsymbol{v_i^l})^2 \leq V_i V_i^* \leq (\boldsymbol{v_i^u})^2 \ \ \forall i \in N$$
$$\boldsymbol{S_i^{gl}} \leq S_i^g \leq \boldsymbol{S_i^{gu}} \ \ \forall i \in N$$
$$S_i^g - \boldsymbol{S_i^d} = \sum_{(i,j) \in E \cup E^R} S_{ij} \ \ \forall i \in N$$
$$S_{ij} = \boldsymbol{Y_{ij}^*} V_i V_i^* - \boldsymbol{Y_{ij}^*} V_i V_j^* \ \ (i,j) \in E \cup E^R$$
$$|S_{ij}|^2 \leq (\boldsymbol{s_{ij}^u})^2 \ \ \forall (i,j) \in E \cup E^R$$
$$-\boldsymbol{\theta_{ij}^\Delta} \leq \angle(V_i V_j^*) \leq \boldsymbol{\theta_{ij}^\Delta} \ \ \forall (i,j) \in E$$

# PowerModels OPF Problem Definition

```
function post_opf(pm::GenericPowerModel)
    variable_voltage(pm)
    variable_generation(pm)
    variable_line_flow(pm)

    objective_min_fuel_cost(pm)

    constraint_theta_ref(pm)
    constraint_voltage(pm)

    for (i,bus) in pm.ref[:bus]
        constraint_kcl_shunt(pm, bus)
    end

    for (i,branch) in pm.ref[:branch]
        constraint_ohms_yt_from(pm, branch)
        constraint_ohms_yt_to(pm, branch)

        constraint_phase_angle_difference(pm, branch)

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

Implicit

d

**variables:**

$S_i^g \;\; \forall i \in N$

$V_i \;\; \forall i \in N$

**minimize:**

$$\sum_{i \in N} f(S_i^g)$$

**subject to:**

$$(\boldsymbol{v_i^l})^2 \leq V_i V_i^* \leq (\boldsymbol{v_i^u})^2 \;\; \forall i \in N$$

$$\boldsymbol{S_i^{gl}} \leq S_i^g \leq \boldsymbol{S_i^{gu}} \;\; \forall i \in N$$

$$S_i^g - \boldsymbol{S_i^d} = \sum_{(i,j) \in E \cup E^R} S_{ij} \;\; \forall i \in N$$

$$S_{ij} = \boldsymbol{Y_{ij}^*} V_i V_i^* - \boldsymbol{Y_{ij}^*} V_i V_j^* \;\; (i,j) \in E \cup E^R$$

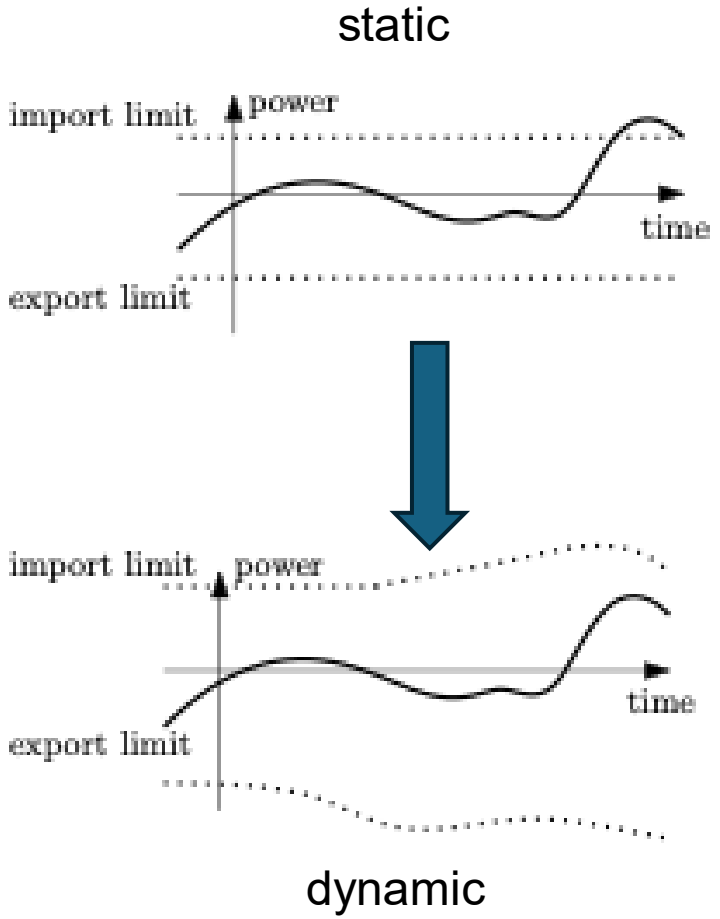
$$|S_{ij}|^2 \leq (\boldsymbol{s_{ij}^u})^2 \;\; \forall (i,j) \in E \cup E^R$$

$$-\boldsymbol{\theta_{ij}^\Delta} \leq \angle(V_i V_j^*) \leq \boldsymbol{\theta_{ij}^\Delta} \;\; \forall (i,j) \in E$$

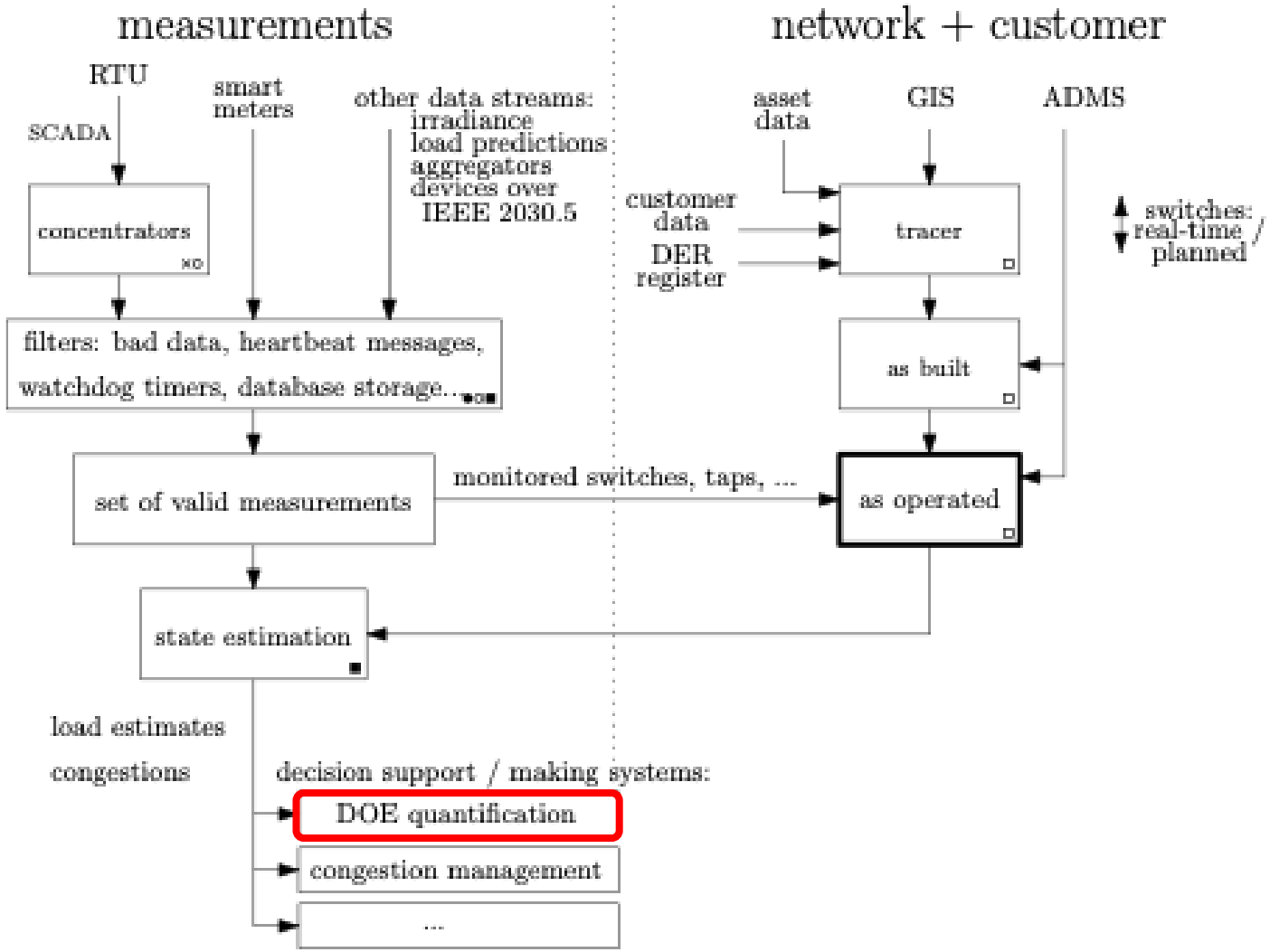
# Dynamic operating envelopes

# From static exports to DOEs

- In Australia, you are typically allowed to *install* rooftop PV up to 5 kW per phase
  - However, you may *not* be allowed to *export* power to the network, e.g. you can only inject 1.5 kW, or even 0.
  - These static limits are driven by voltage-based congestion
- In practice, even in congested networks, there is spare capacity a lot of the time, which is not accessible under these static export limits
- … make them dynamic?



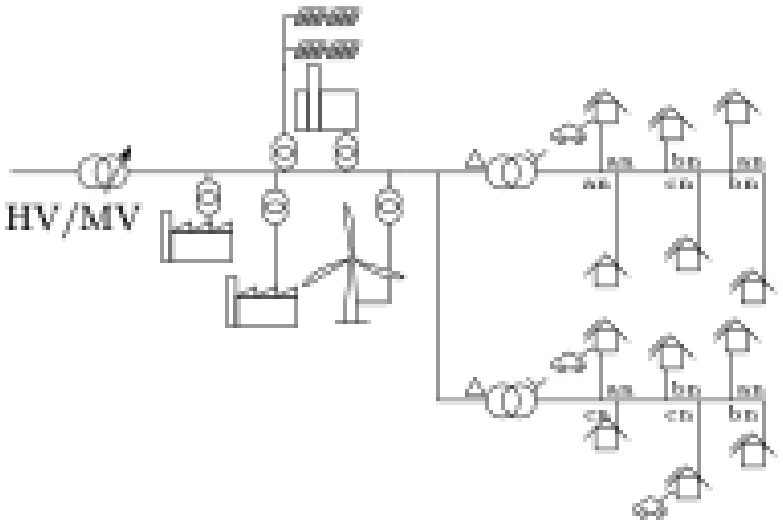
# Visibility to enable DOE

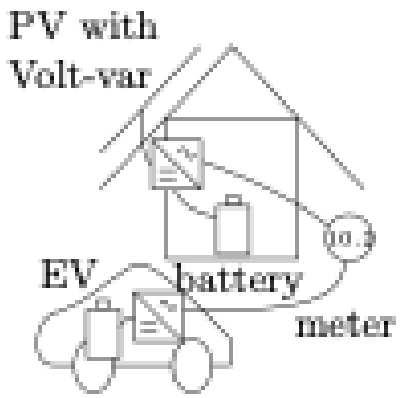


# Operating envelopes

- Maximum export values that can be realized simultaneously without violating the network envelopes for voltage magnitude and thermal limits
  - Optimisation is a natural framework for finding minima and maxima
- Approach: optimisation problems that include the distribution network physics
  - Including expected inverter response
  - Taking into account different objectives for equity and competitiveness

# DOE




OPF example

min. cost of dispatch
s.t. network physics
operating limits: S, V, I
devices: EV, PV, battery
inverter control

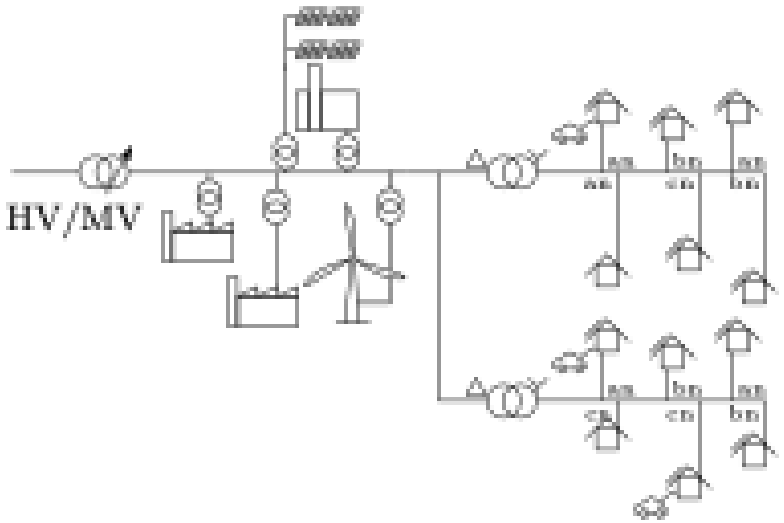
outcome:
feasible dispatch &
network within
operating limits

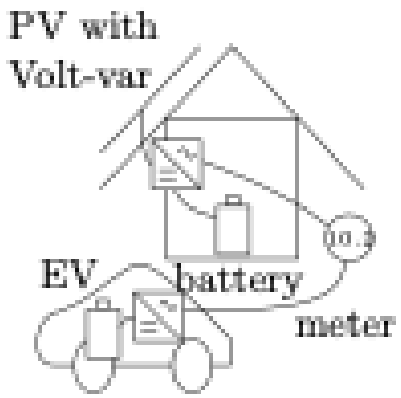
DOE example

max. aggregate exports
s.t. network physics
operating limits: S, V, I
devices: EV, PV, battery
inverter control

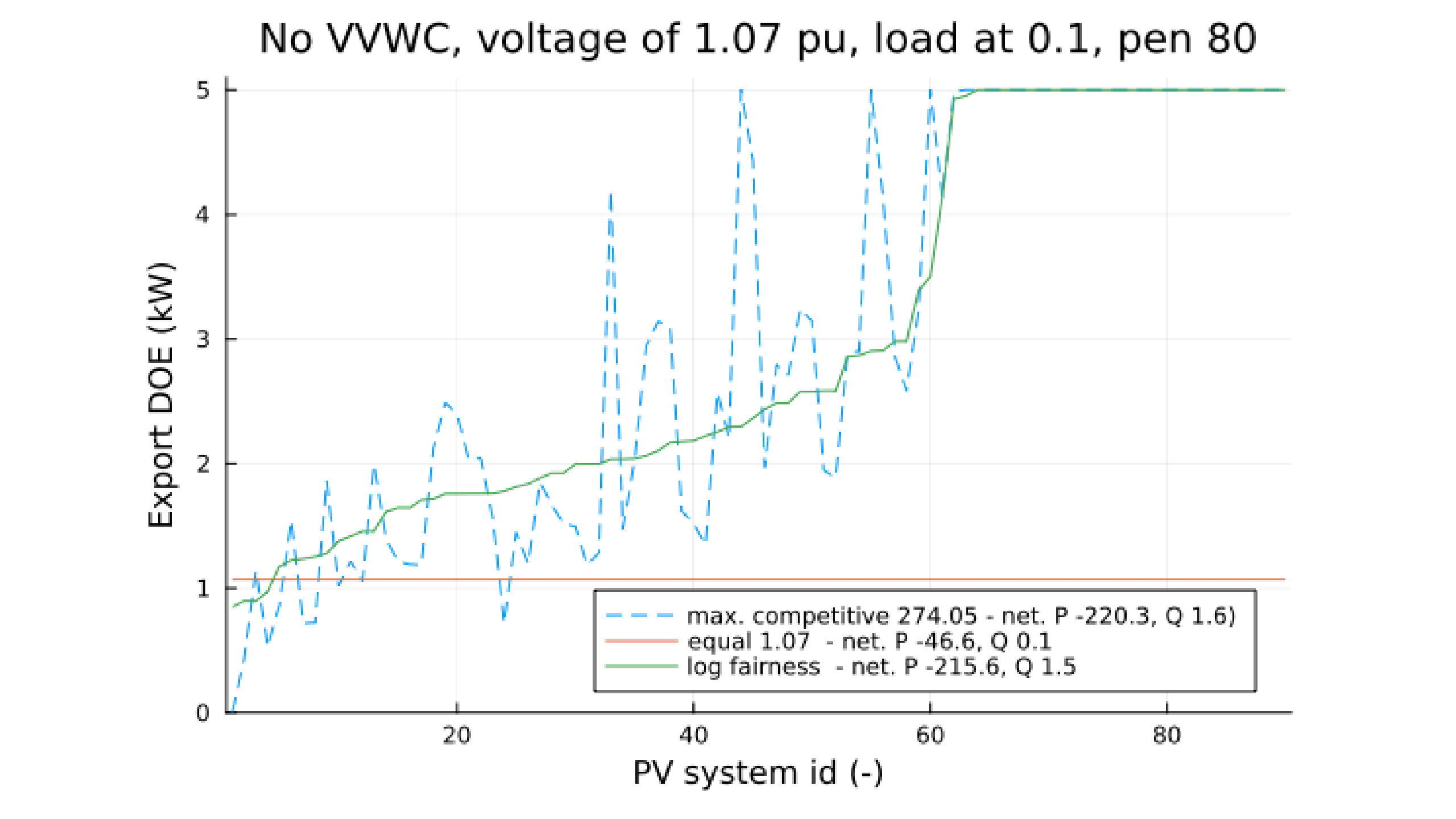
outcome:
feasible export ranges

# DOE

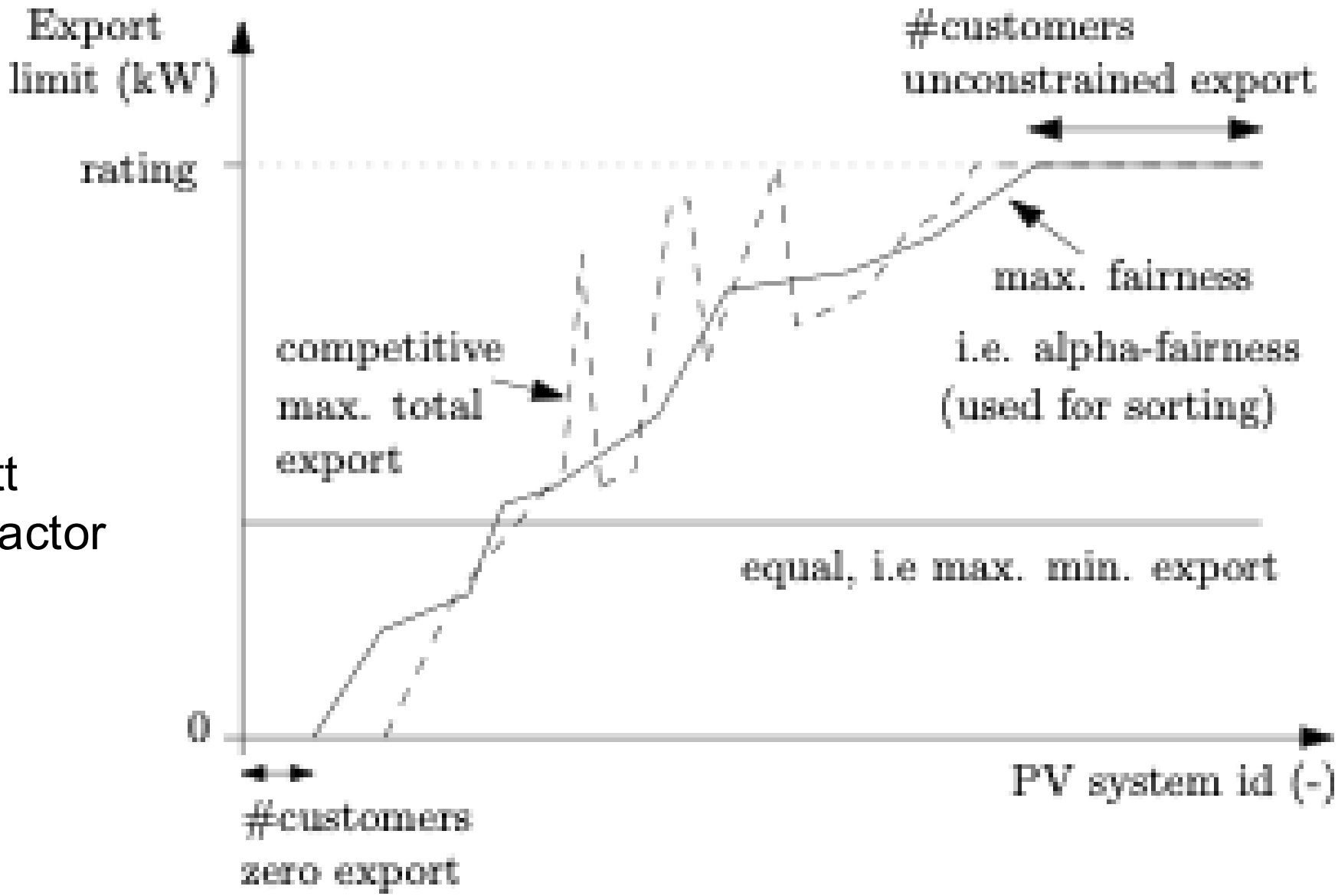


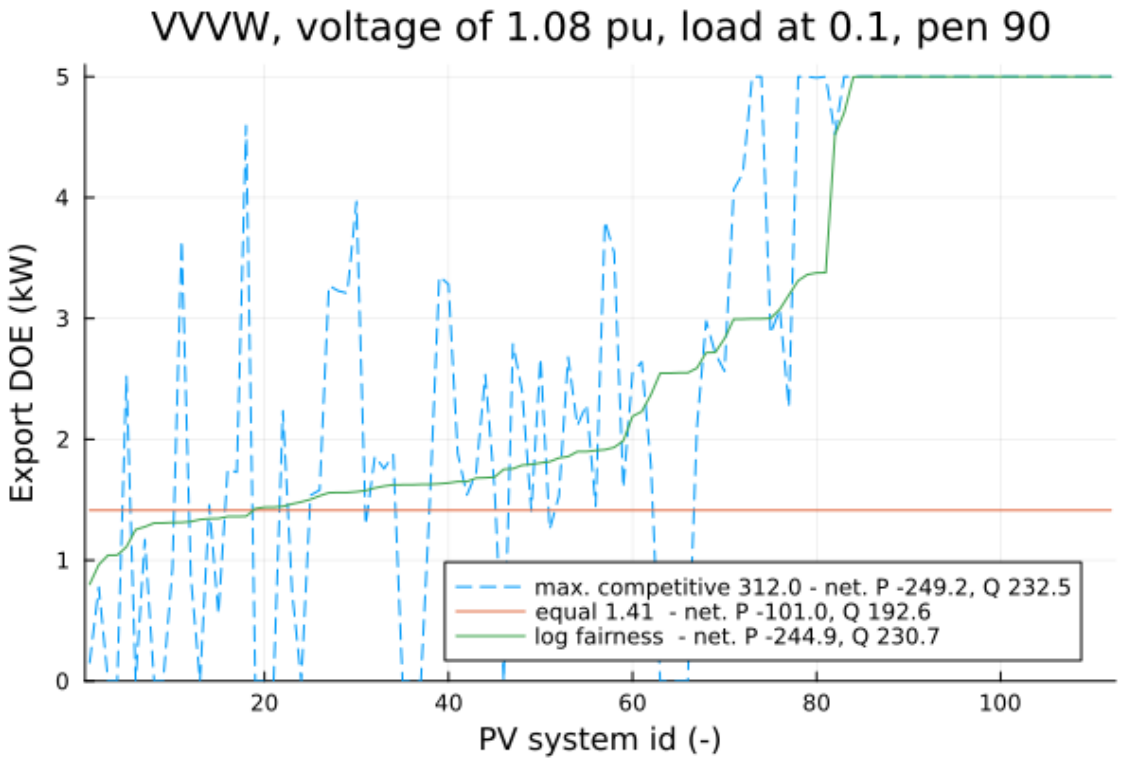

| OPF example | DOE example |
| --- | --- |
| min. cost of dispatch | max. aggregate exports |
| s.t. network physics | s.t. network physics |
| operating limits: S, V, I | operating limits: S, V, I |
| devices: EV, PV, battery | devices: EV, PV, battery |
| inverter control | inverter control |
| outcome: | outcome: |
| feasible dispatch & network within operating limits | feasible export ranges |

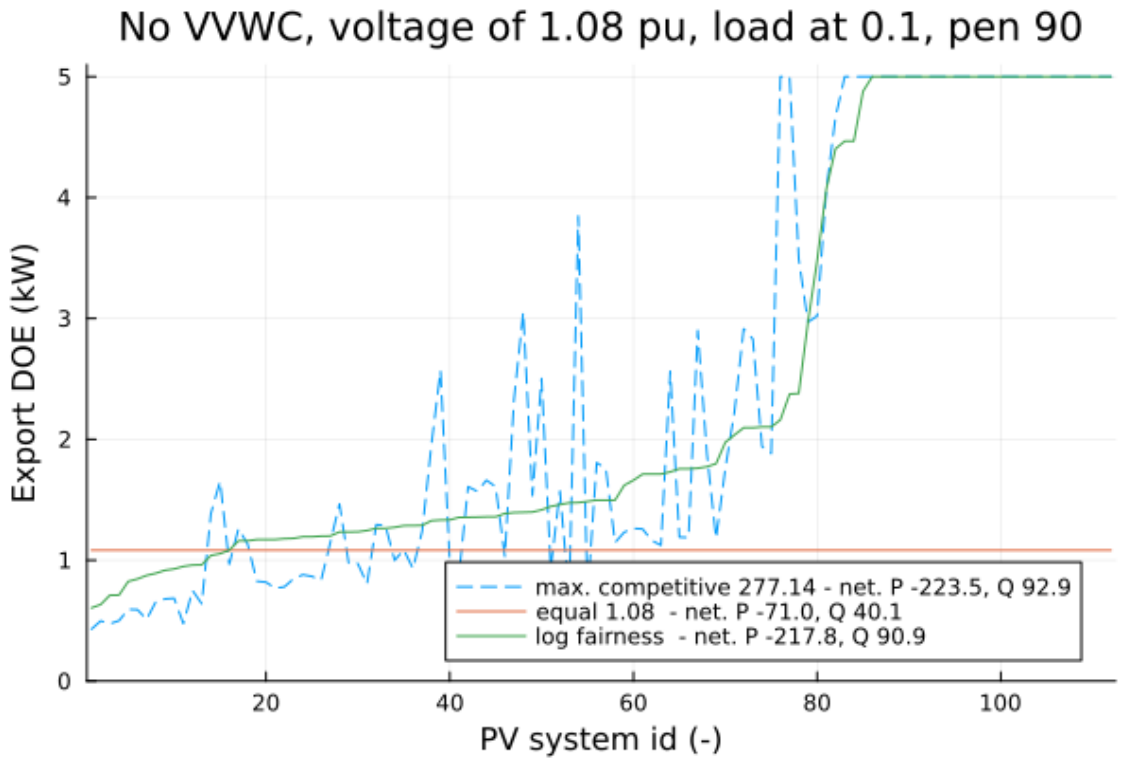

No VVWC, voltage of 1.07 pu, load at 0.1, pen 80
Export DOE (kW)
PV system id (-)
0
1
2
3
4
5
20
40
60
80
max. competitive 274.05 - net. P -220.3, Q 1.6)
equal 1.07 - net. P -46.6, Q 0.1
log fairness - net. P -215.6, Q 1.5

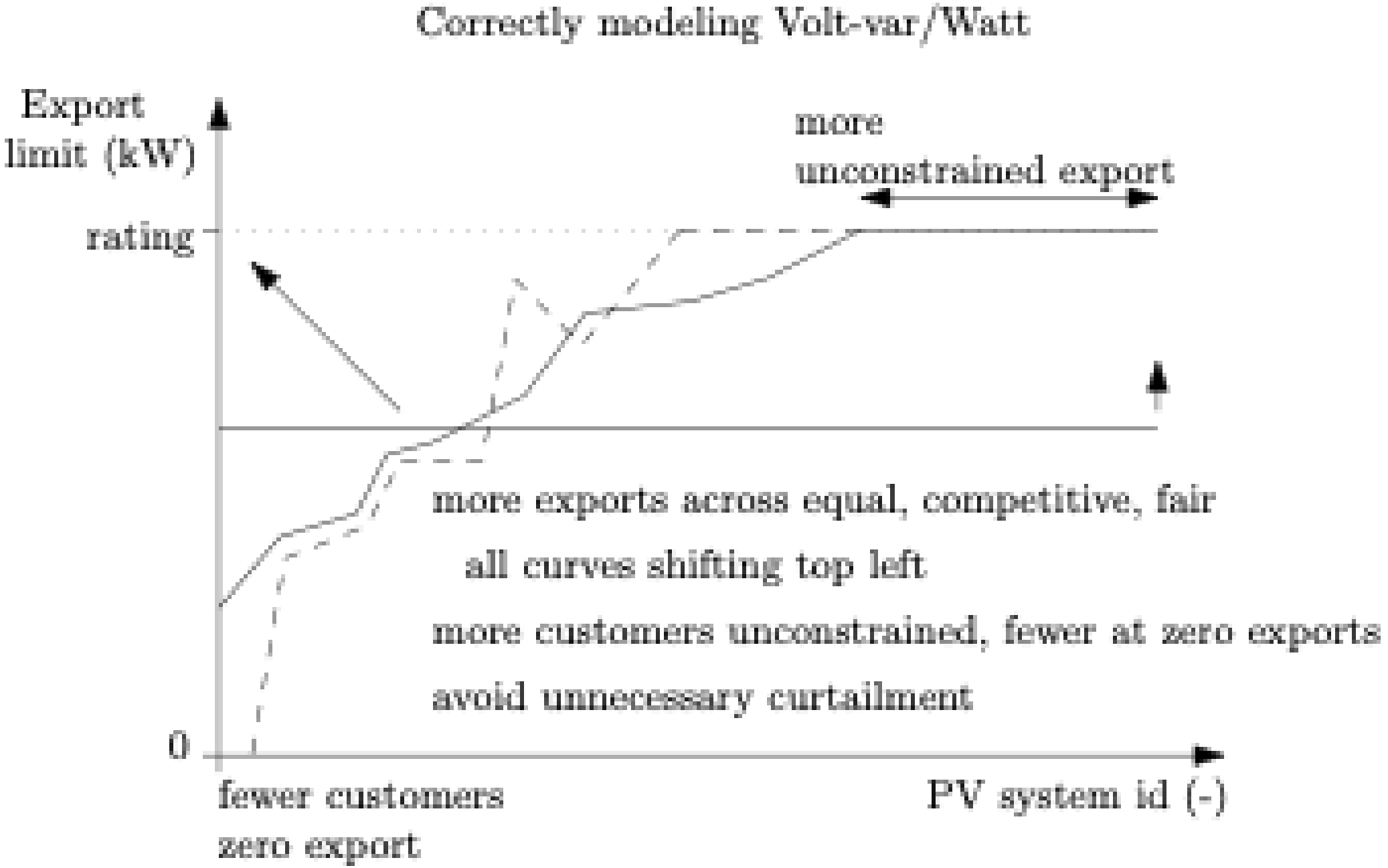
Studies across
- PV penetration
- Load levels
- Voltage at infeeder
- PV with volt-var/watt
  vs constant power factor



# With vs without volt-var/watt




Correctly modeling Volt-var/Watt
Export limit (kW)
rating
0
more unconstrained export
more exports across equal, competitive, fair
all curves shifting top left
more customers unconstrained, fewer at zero exports
avoid unnecessary curtailment
fewer customers zero export
PV system id (-)

# Challenges

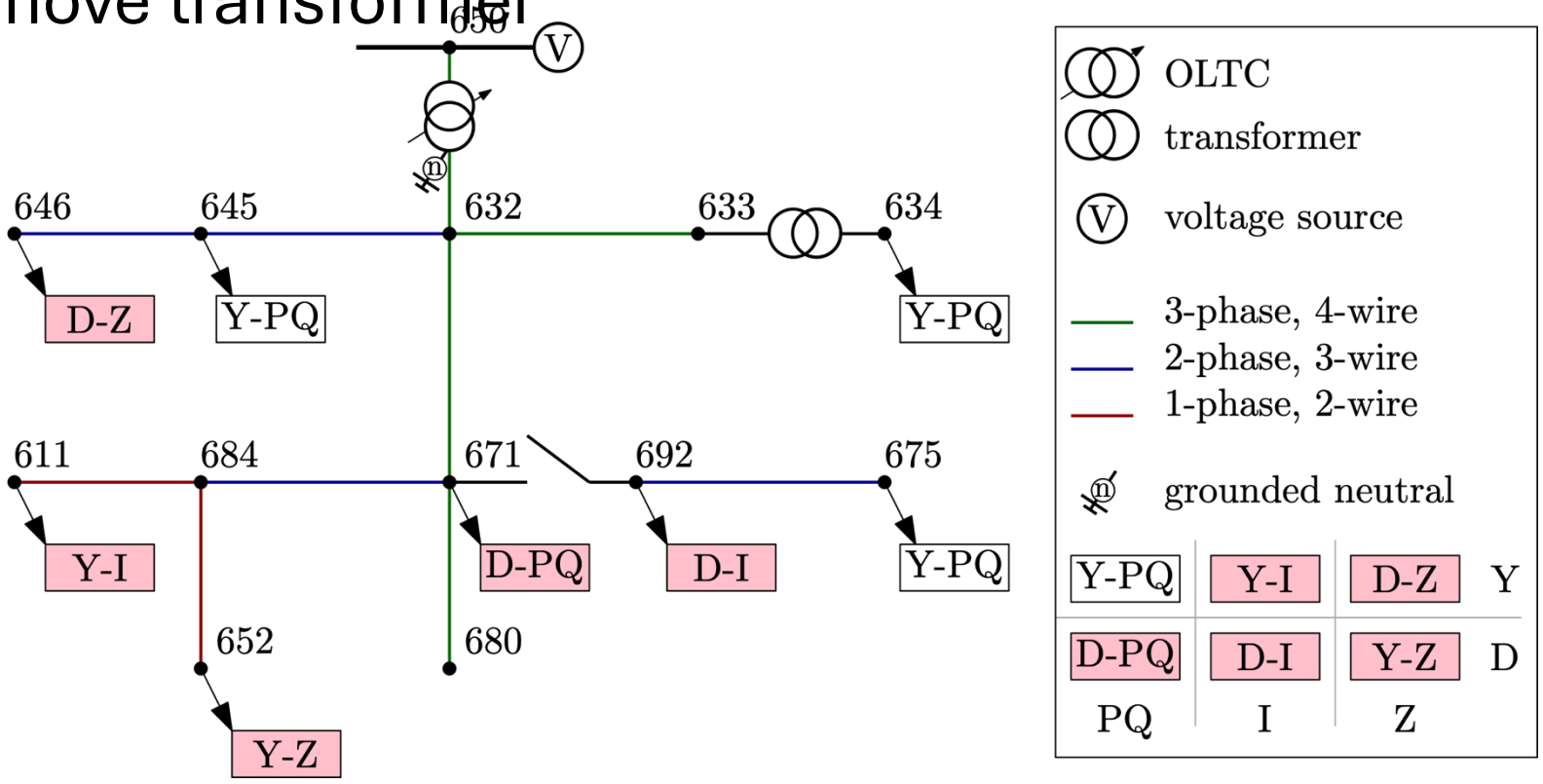
- How to make the choice of a specific DOE fairness trade-off as utility or policy maker
- Flexible import limits: flexible vs inflexible demand
  - EVs, battery, pool heaters
- Incorporating OLTCs and statcoms into DOE quantification
- Network planning methods taking the presence of DOEs into account
- Network data debugging, cleaning, calibration, validation methods
- How to measure and validate the outcomes of DOE *deployments*
- Flex markets and DOEs: compatibility and complementarity?

# IEEE Distribution test feeders

- https://cmte.ieee.org/pes-testfeeders/resources/
- Made for simulation, not optimization
- Designed to confirm consistent results between different simulation engines, for a large variety of network topologies, transformer designs

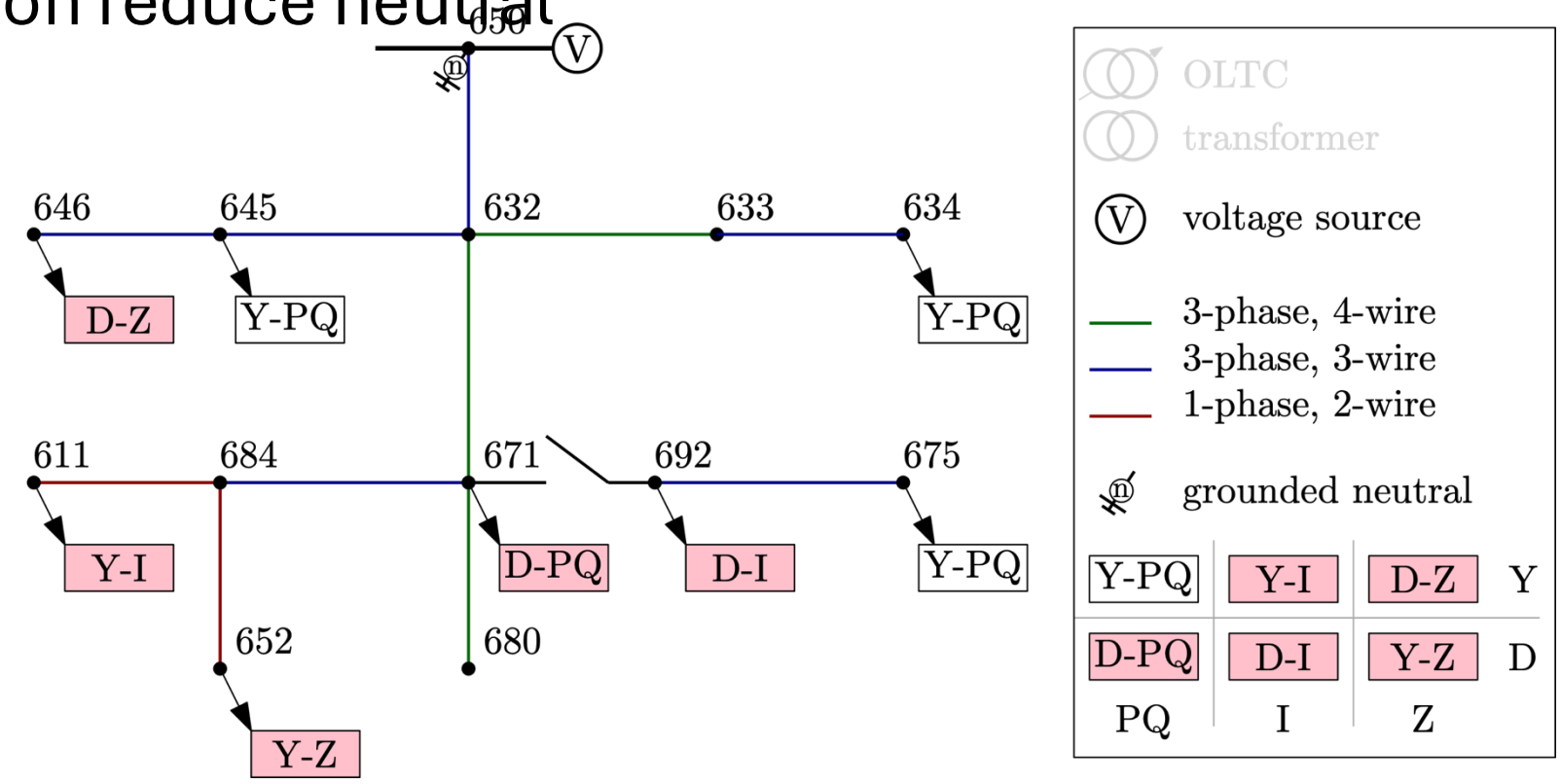
# Example: IEEE 13

- remove transformer



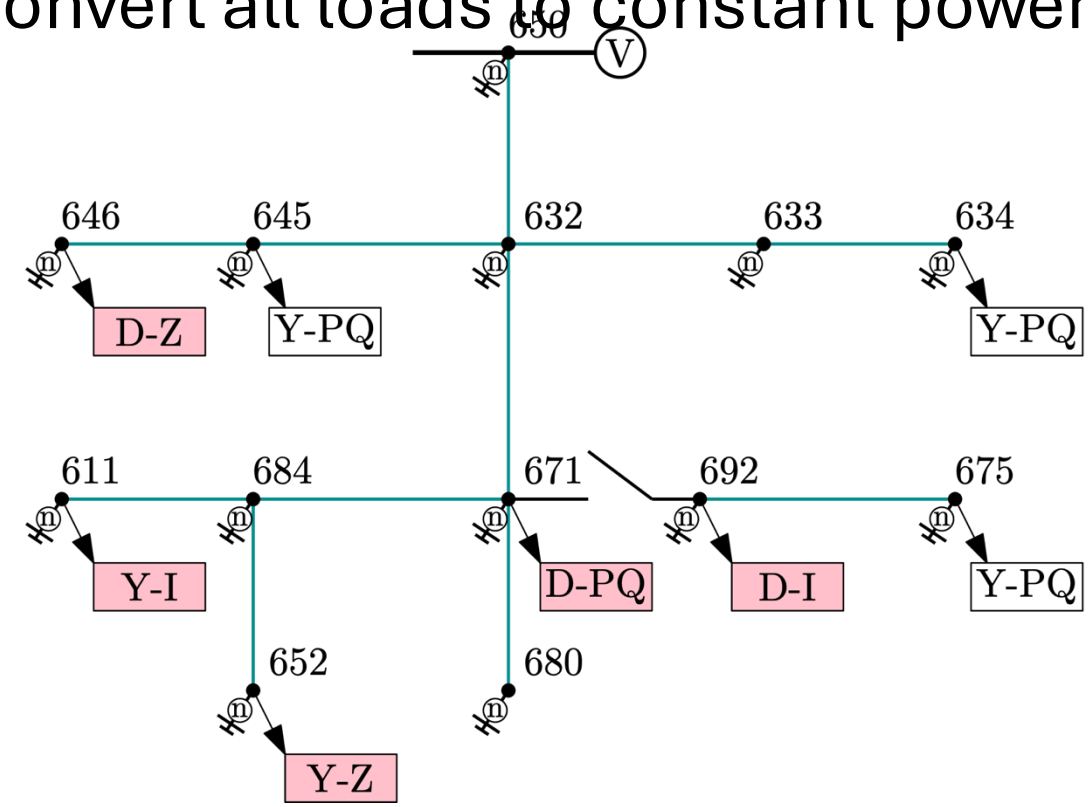
# Example: IEEE 13

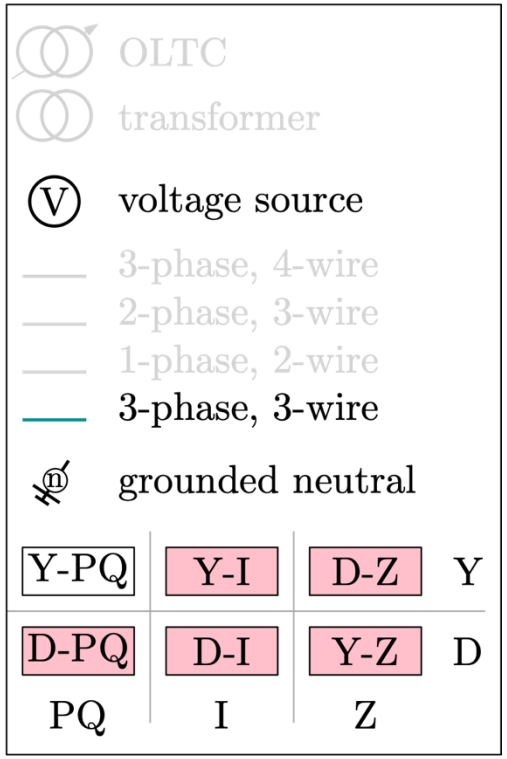
- Kron reduce neutral



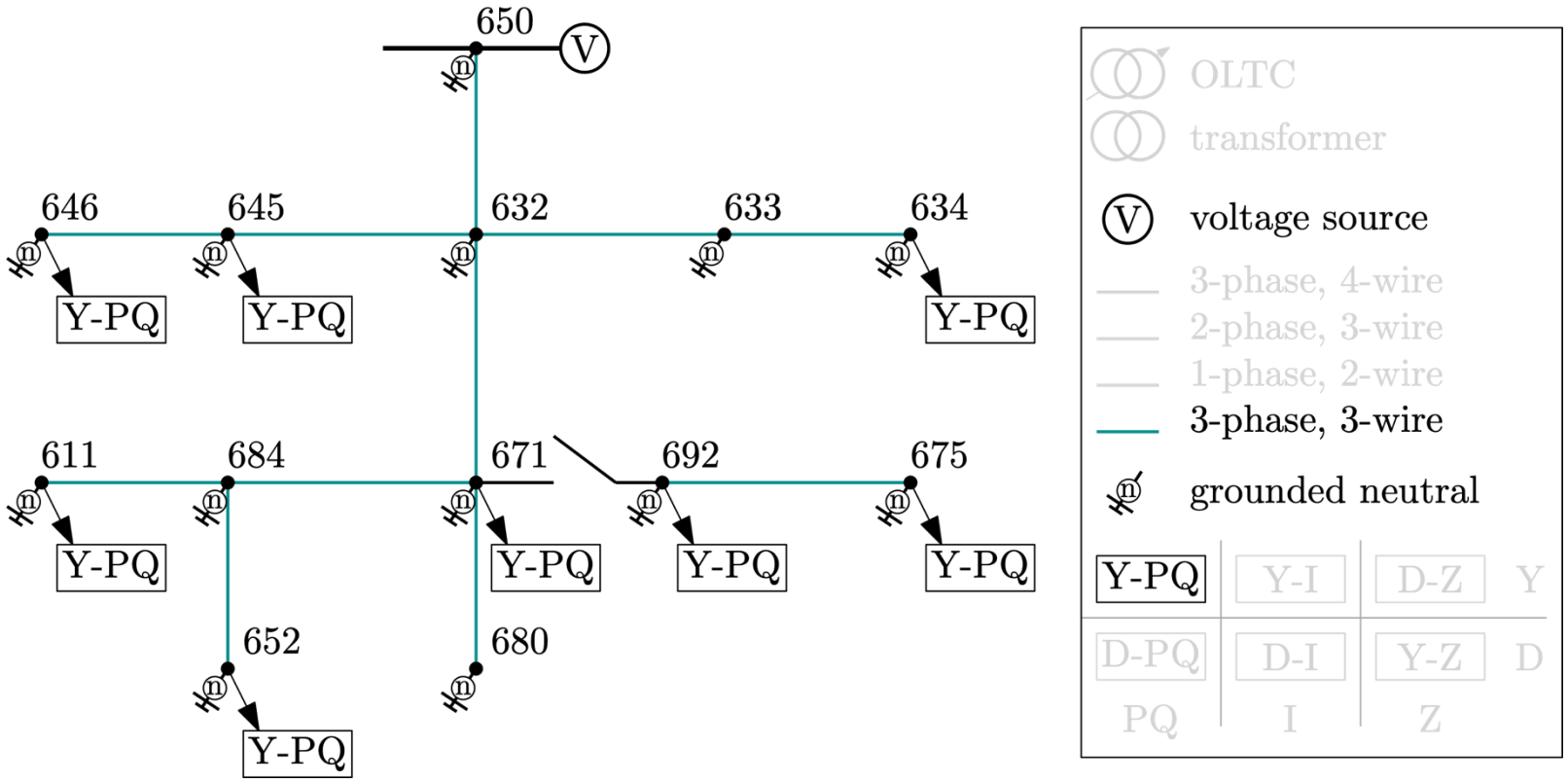
# Example: IEEE 13

- convert all loads to constant power



# Example: IEEE 13



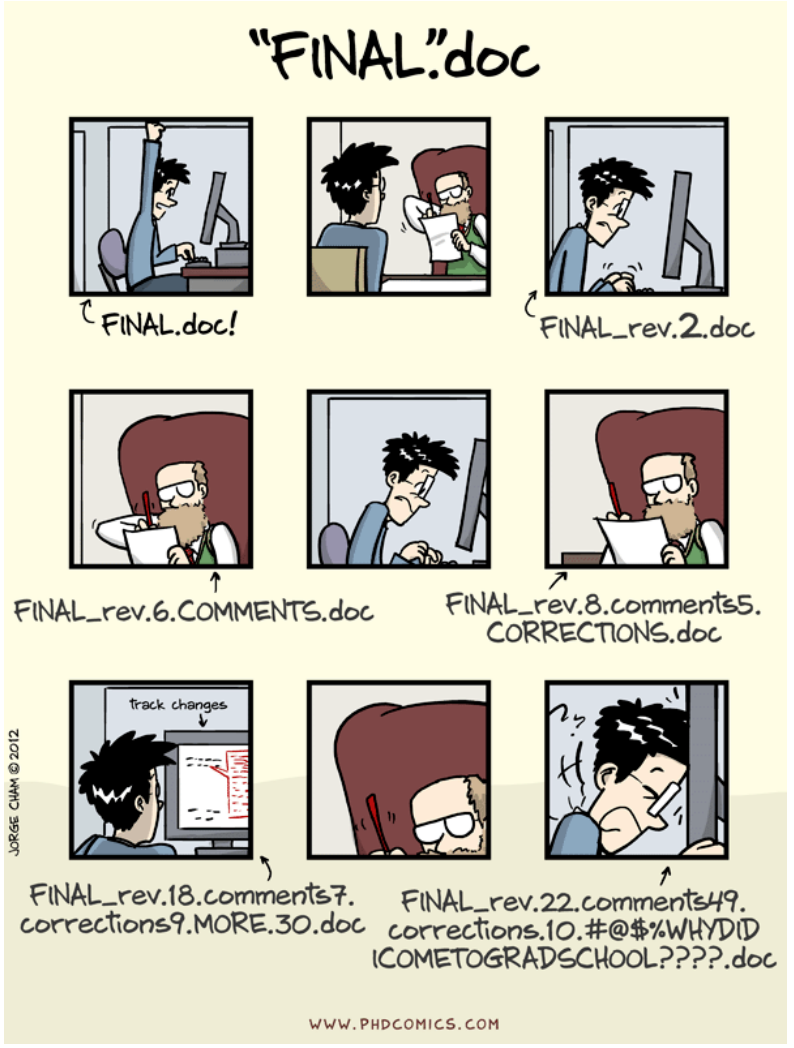
# Australian Distribution Network Data

- https://data.csiro.au/collection/csiro:65408?q=geth%20heidari&_st=keyword&_str=3&_si=1

Parses in both PMD and OpenDSSDirect

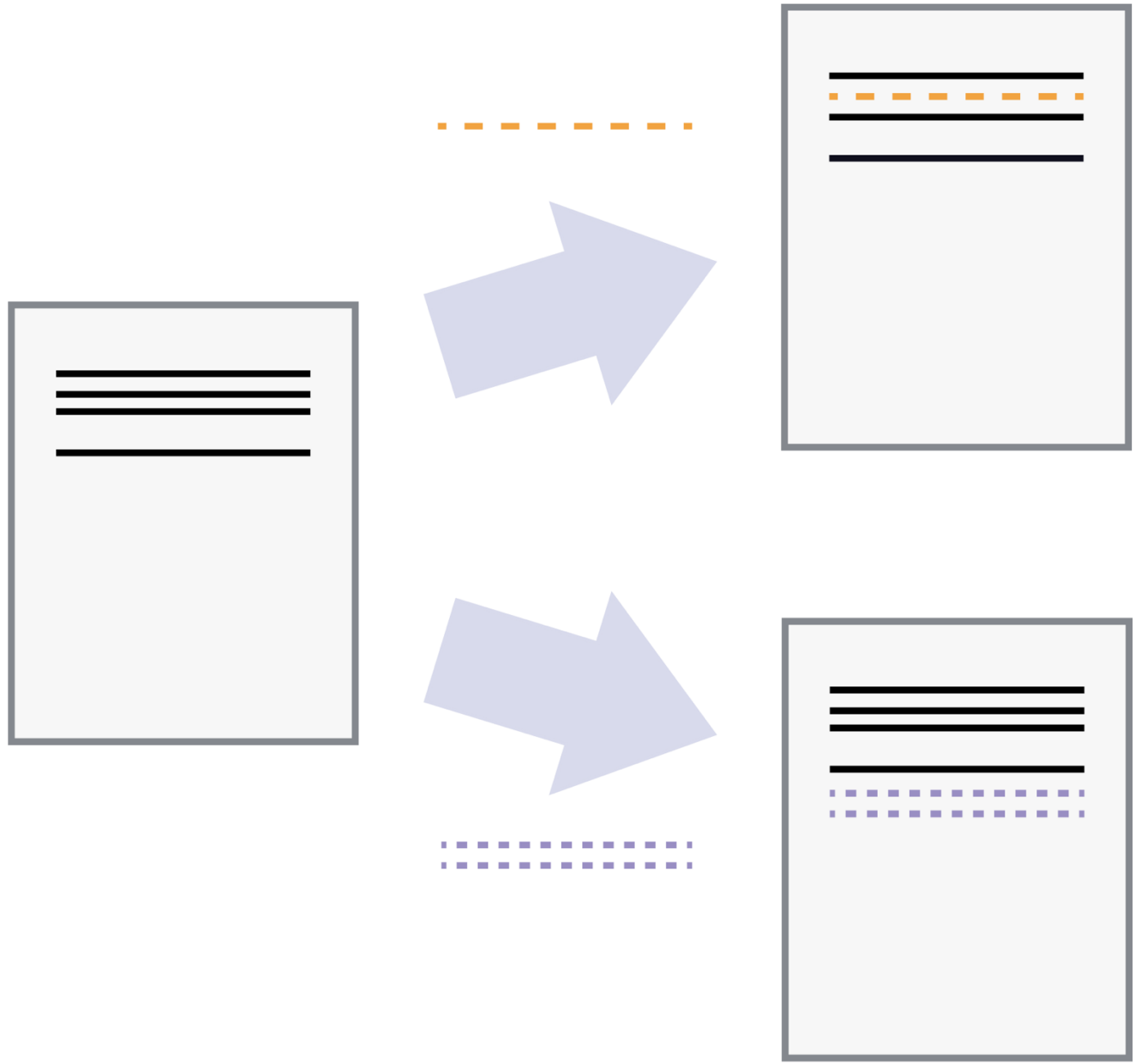
# Automated Version Control

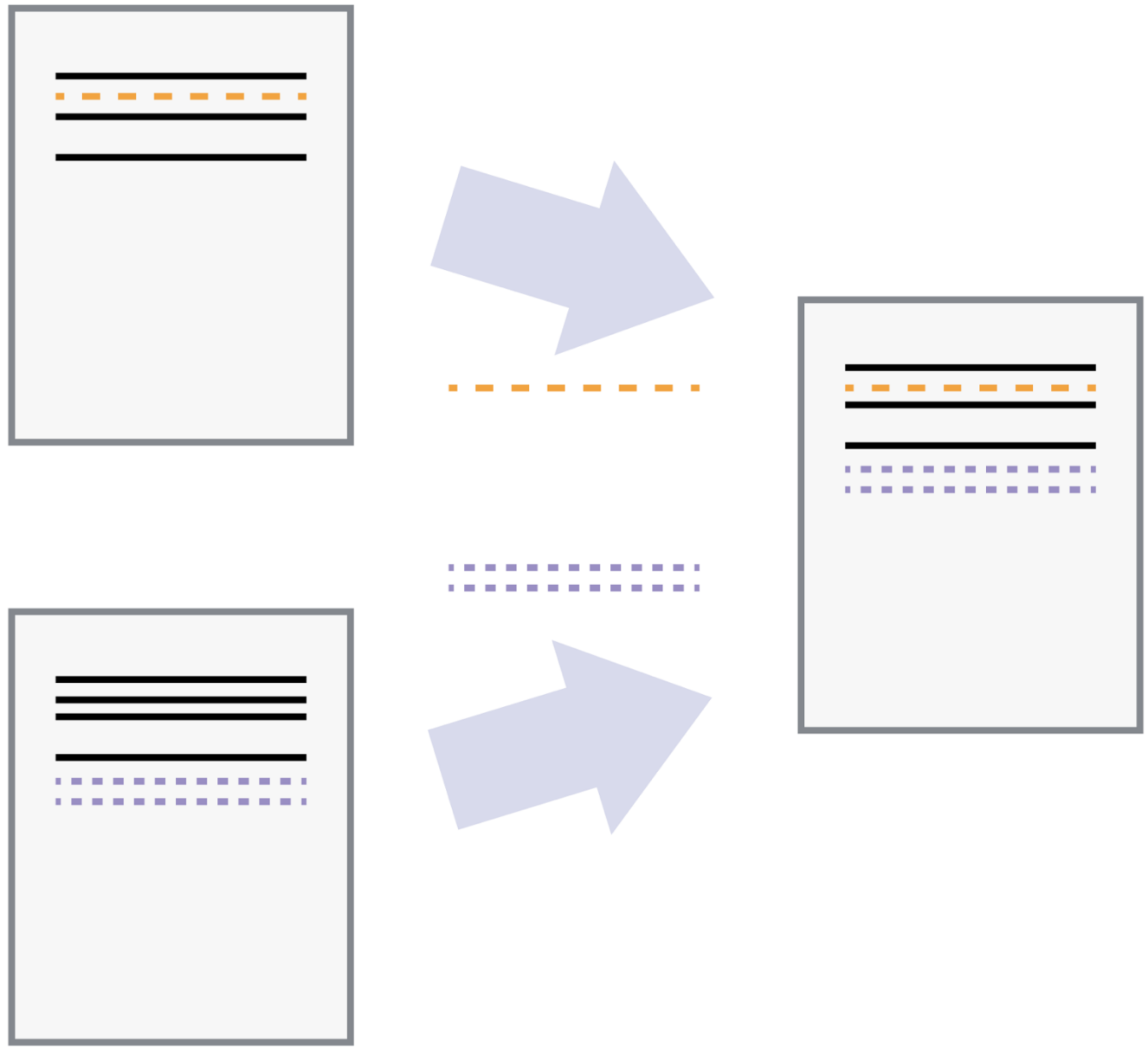
- Version control is like an unlimited ‘undo’.
- Version control also allows many people to work in parallel.
- Even if you work on your own, it can be very powerful
- Cheat code if you collab



# Version Control with Git: Key Concepts

- Git is a toolkit for Automated Version Control
- We need to do some initial work to configure the Git software for each computer.
- Git works at the level of a folder/directory of files.
- We start using Git when we 'initialise a repository' and we 'add' files we want to do version control with.
- From then on, Git tracks changes to those files.
- We control the snapshots / save points / check-points of every file that is being tracked.
- Git allows us to write useful explanations of what we did, so we have a history of our work.
- Git is designed for sharing code over the internet.
- We have tools to help us agree on changes made by more than one person.





# Automated Version Control: Key Concepts

- Automated version control is the **lab notebook** of the digital world
- A version control system is a tool that tracks changes for us, like an **unlimited ‘undo’**.
- Version control also allows many people to work in parallel
  - and is equally useful for an individual
- Version control is really a way of recording **metadata** about your work.

# Setting Up Git: `git config`

```
$ git config --global user.name "Vlad Dracula"
$ git config --global user.email
“vlad@tran.sylvan.ia"

$ git config --global core.editor
“c:/Windows/System32/notepad.exe"

$ git config --global init.defaultBranch main

$ git config --list
```

# git init

- Initialise / create a **repository**.
  - meaning: set up a folder so it is ready for tracking changes using Git.

- A *repository* is a folder where a version control system stores the full history of a project, alongside the project files.
- Git stores all of its repository data in the *.git* directory (or folder).
  - This directory is often hidden in the filesystem.

# Staging

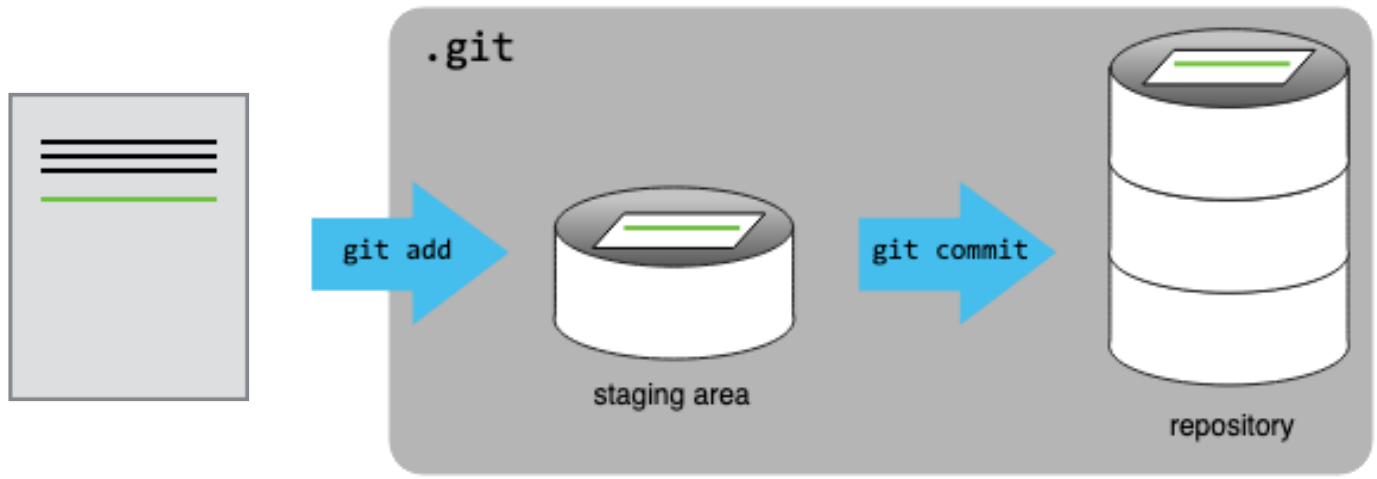
- Suppose we have a file called `mars.txt`
- We will now tell Git that we want to *start tracking changes* for our new file.
- Do this for our new file `mars.txt` by ***add***ing it to our set of *Staged* files.

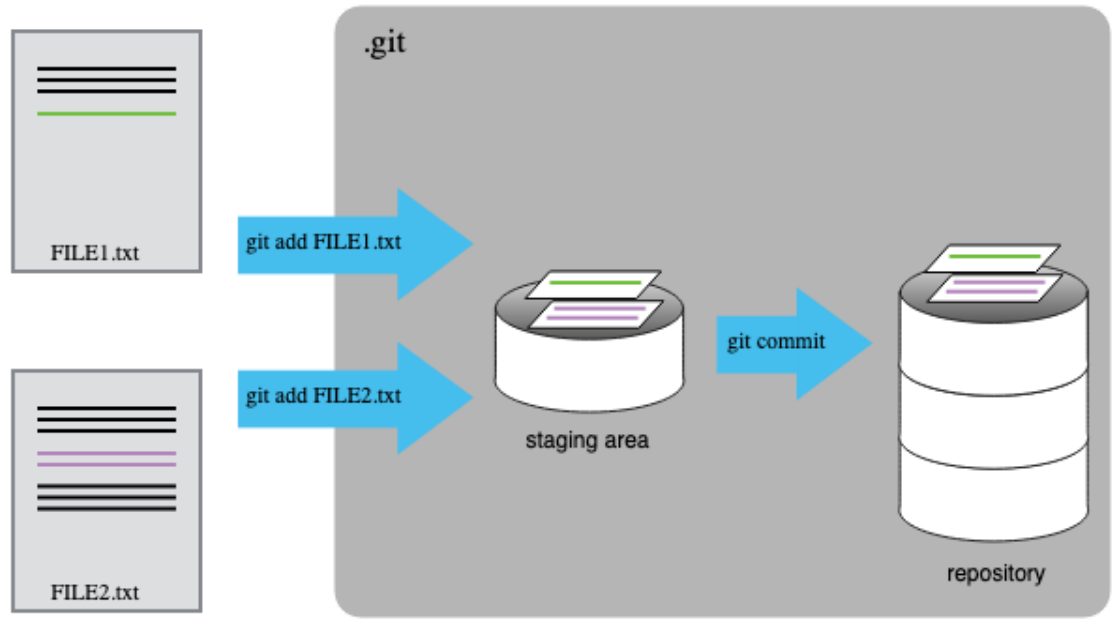
```
git add mars.txt
```

- Git now knows that it's supposed to keep track of `mars.txt`, but it hasn't recorded these changes as a *commit* yet.

# Commits

- Before we get Git to record the change, we need to add a description of the changes.
- This is called a *commit message* for adding the file `mars.txt`
  - **git commit –m "Start notes on Mars base."**
- Once done, we can now record the changes along with our message to our future selves by *committing*.
- When we run `git commit`, Git takes everything we have told it to save by using `git add` and stores a copy permanently inside the special *.git* directory.
- This permanent copy is called a *commit* (or *revision*)


.git
git add
staging area
git commit
repository

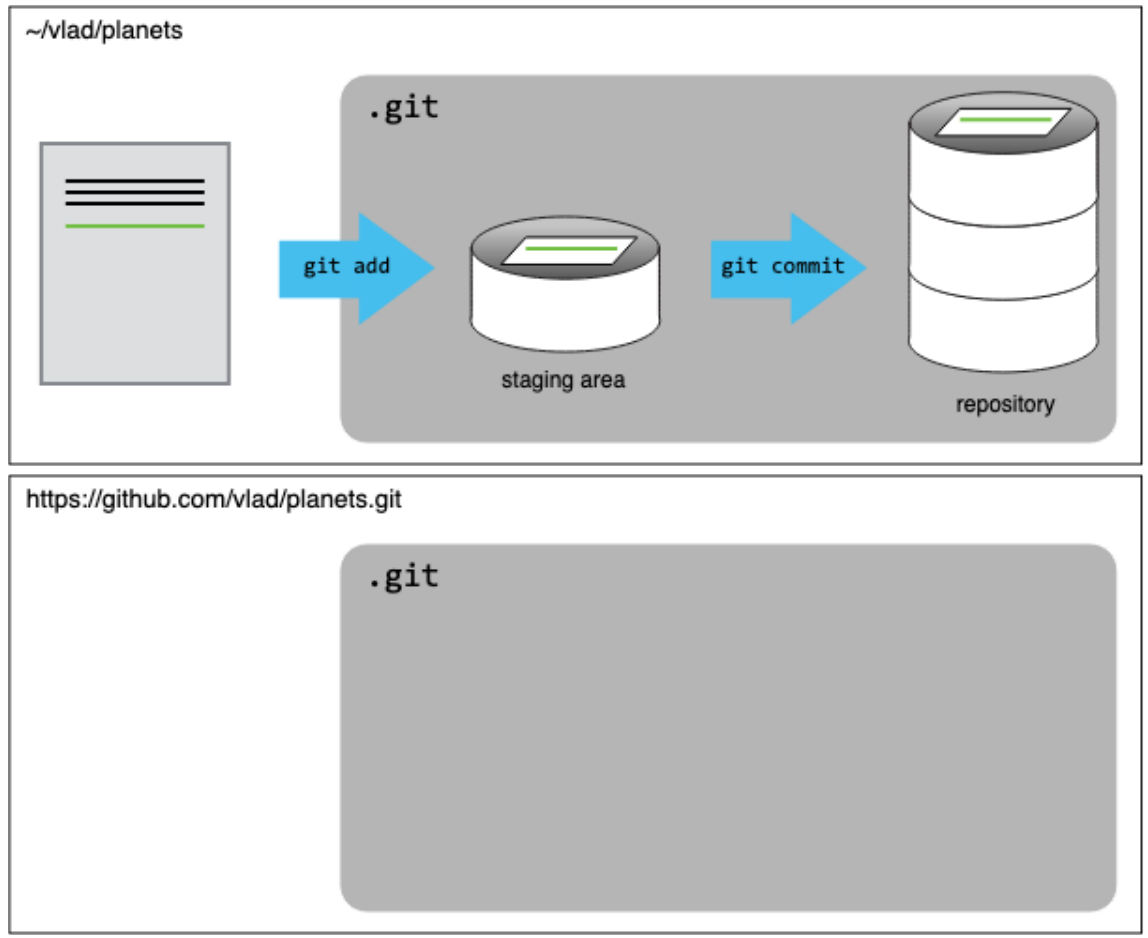

.git
FILE1.txt
FILE2.txt
git add FILE1.txt
git add FILE2.txt
staging area
git commit
repository

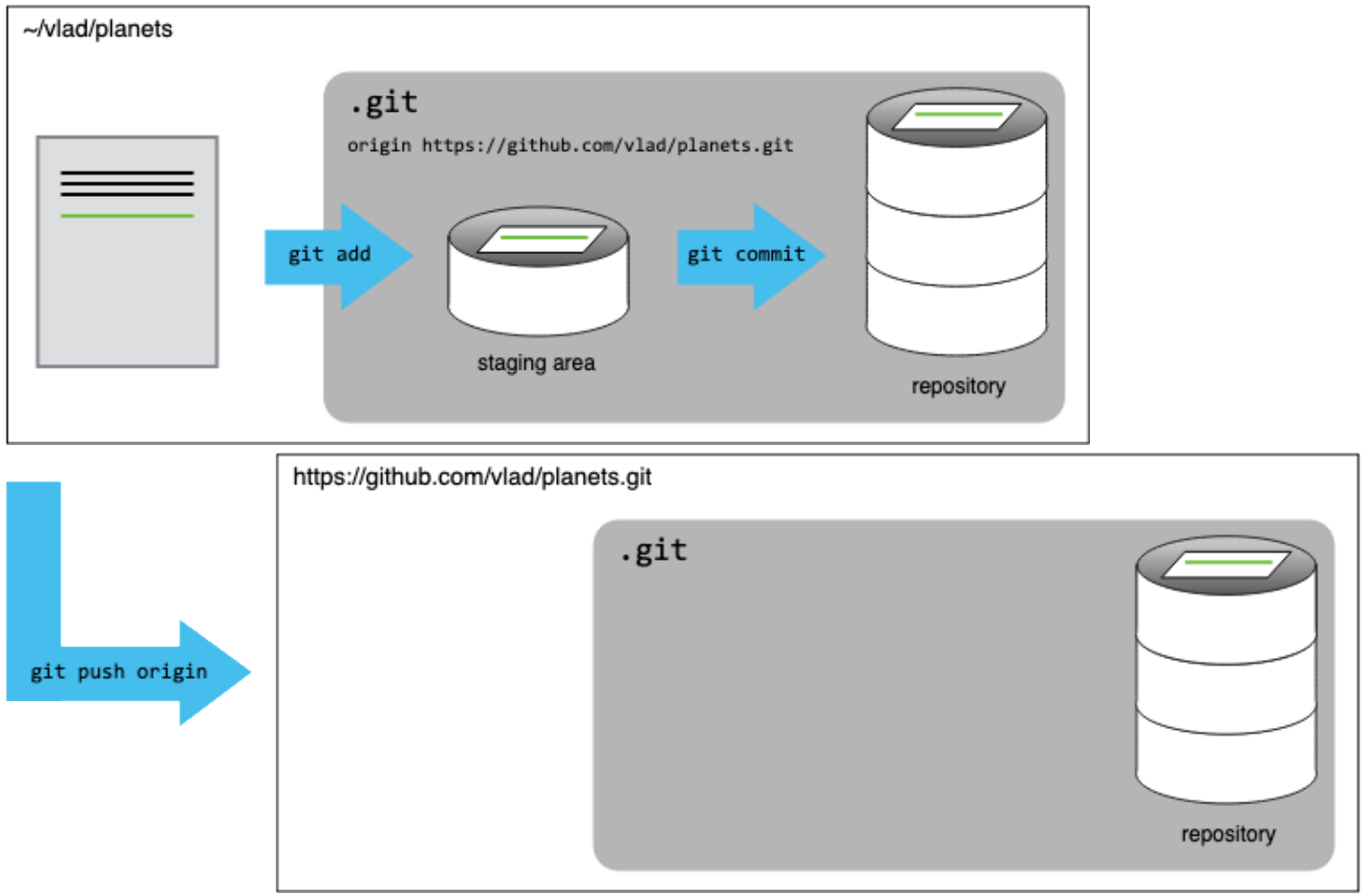
# Remotes and cloning

- A local Git repository can be connected to one or more remote repositories.
  - The SSH protocol is used to connect to remote repositories.

- `git push`
  - copies changes from a local repository to a remote repository.

- `git pull`
  - copies changes from a remote repository to a local repository.


~/vlad/planets
.git
git add
staging area
git commit
repository
https://github.com/vlad/planets.git
.git


~/vlad/planets
.git
origin https://github.com/vlad/planets.git
git add
staging area
git commit
repository
git push origin
https://github.com/vlad/planets.git
.git
repository

# `push` vs. `commit`

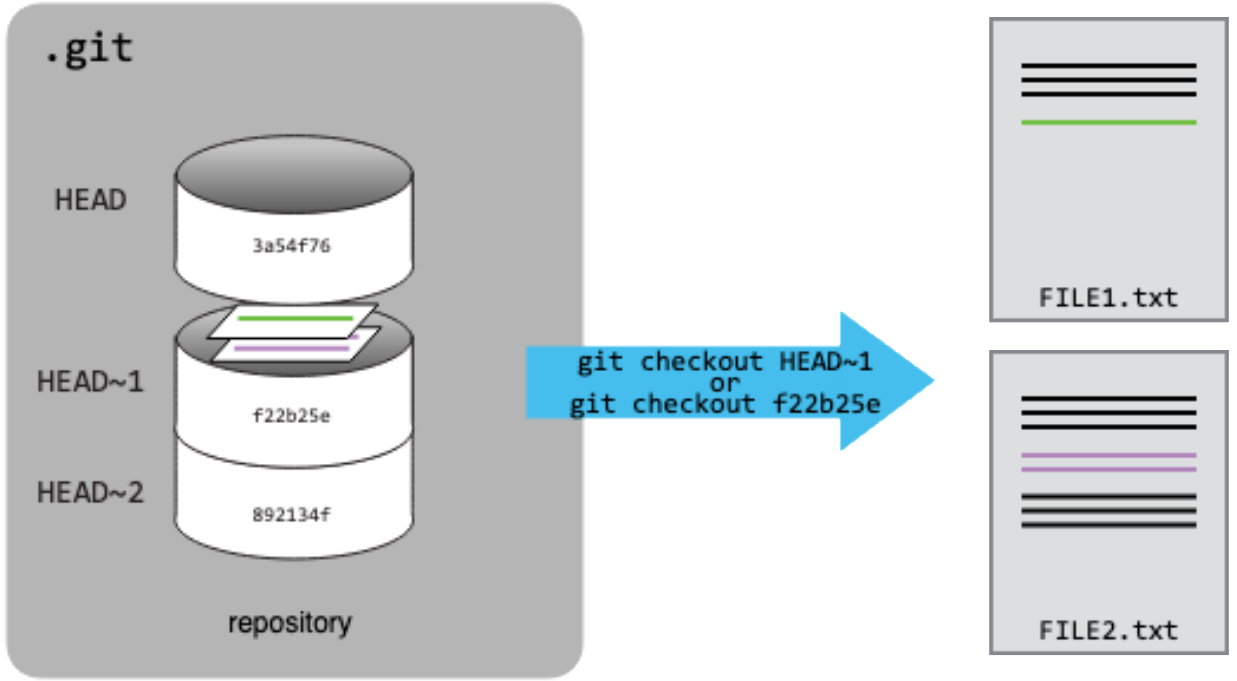
- When we `push` changes, we're interacting with a remote repository to update it with the changes we've made locally

- This corresponds to sharing the changes we've made with others.

- `commit` only updates your local repository with your own changes.

# Unlimited undo: `git checkout`

- We can go ‘back in time’ to a previous state of our project.
- This is very useful if
  - we’ve introduced a bug or error
  - want to reproduce our our analysis at a specific point in time.
- We can get the complete state of the project at a specific commit point by using a unique identifier or relative to the current state (or `HEAD`)



# Demonstrating an IDE with Git integration

# Repo with notebooks

- Install VSCode
  - https://code.visualstudio.com/
- Install Julialang
  - https://julialang.org/
- Download notebooks (check out the whole repo )
  - https://github.com/Lauraz48/EPICS_TUT_2026

# Tutorial

- Coding tutorials focus on 2 things
  - Running a power flow for a network directly using OpenDSSDirect.jl
  - PowerModelsDistribution -> set up an optimisation problem for same network
    - Showcase some of the benefits

- Notebooks
  - https://github.com/Lauraz48/E_tut_ongoing

# YouTube resources

- JuliaLang
  - https://www.youtube.com/c/TheJuliaLanguage
- PMD
  - https://www.youtube.com/watch?v=LAxytFlle9E
- PMDSE
  - https://www.youtube.com/watch?v=Bl3FsP_6WuM
- InfrastuctureModels
  - https://www.youtube.com/watch?v=POOt1FCA8LI
- JuMP-dev 2025
  - https://www.youtube.com/watch?v=mbIbjj2waxE&list=PLP8iPy9hna6RLvzHeEVCwkUkZV3aChElQ

# OpenDSS video training

- https://www.epri.com/events/d2a46142-33e3-4cf5-9d12-ef8d31c9f84f -> attachments -> media
- https://www.epri.com/events/c97292af-0790-4514-b9df-ccef80d685e1 -> attachments -> media
- Prof. Nando Ochoa has various recordings linked here 🙏 https://sites.google.com/view/luisfochoa/invited-talks-webinars-and-tutorials

# Distribution network modeling text books

- T.A. Short, Electric Power Distribution Handbook, Second Edition, CRC Press, ISBN9781466598652
- James J. Burke, Power Distribution Engineering: Fundamentals and Applications, CRC Press, ISBN 9780824792374
- Turan Gonen, Electric Power Distribution Engineering, Third Edition, CRC Press, ISBN 9781482207002
- J. Duncan Glover, Thomas Overbye, Mulukutla S. Sarma, Power System Analysis and Design, Sixth Edition, Cengage, ISBN 9781305636200
- Luces M. Faulkenberry, Walter Coffer, Electrical Power Distribution and Transmission, Prentice Hall, ISBN 9780132499477
- James Momoh, *Smart Grid: Fundamentals of Design and Analysis, 1st Edition*, Wiley, ISBN 978-0470889398
- Steven Low's lecture notes, https://netlab.caltech.edu/book/

# PMD related publications


- Geth, F., Heidari, R., Chapman, A. & Clark, J., Considerations and Design Goals for Unbalanced Optimal Power Flow Benchmarks, PSCC 2024
- Heidari, R., and Geth, F., Improved Algebraic Inverter Modelling for Four-Wire Power Flow Optimization, PSCC 2024
- Vanin M., Geth F., D'hulst R, & Van Hertem, D. "Combined Unbalanced Distribution System State and Line Impedance Matrix Estimation", Int. J. Electrical Power & Energy Systems
- Geth F., Claeys S., & Heidari R. "On the Implementation of the Fixed Point Iteration Current Injection Method to Solve Four-Wire Unbalanced Power Flow in PowerModelsDistribution.jl"
- Geth F. "Pitfalls of Zero Voltage Values in Optimal Power Flow Problems", IEEE PES General Meeting 2023
- Geth F., Heidari R. & Koirala A. "Computational analysis of impedance transformations for four-wire power networks with sparse neutral grounding", ACM E-energy 2022
- Geth F. & Liu B., "Notes on BIM and BFM optimal power flow with parallel lines and total current limits", IEEE PES General Meeting 2022
- Claeys, S., Geth, F., Sankur, M., & Deconinck, G. (2021). No-load linearization of the lifted multi-phase branch flow model: equivalence and case studies. IEEE ISGT Europe, October.
- Van Acker, T., Geth, F., Koirala, A., & Ergun, H., General **polynomial chaos** in the current-voltage formulation of the optimal power flow problem, EPSR 2022
- Geth, F. & Van Acker, T., **Harmonic** optimal power flow with **transformer excitation**, EPSR 2022
- Claeys, S., Geth, F., & Deconinck, G. Optimal power flow in **four-wire** distribution networks: formulation and benchmarking. EPSR 2022
- Claeys, S., Vanin, M., Geth, F., & Deconinck, G., "Applications of optimization models for electricity distribution networks", Wiley Interdisciplinary Reviews: Energy and Environment, vol. 10, no. 5, pp. e401, September/October 2021
- Claeys, S., Geth, F., & Deconinck, G. (2021). **Line parameter estimation** in multi-phase distribution grids without voltage angle measurements. CIRED Open Access Journal, September
- Geth, F., & Coffrin, C. (2019). "Direct method to **recover current and voltage** in multi-conductor optimal power flow models." In IEEE PES General Meeting Atlanta 2019, Georgia.
- Claeys, S., Geth, F., & Deconinck, G. "Optimization of **gang-operated on-load tap changers** in multi-conductor radial networks: formulation and **convex relaxation**. In CIGRE Chengdu 2019 Symposium (pp. 1–10).
- Fobes, D. M., Coffrin, C., Geth, F., & Claeys, S. (2020). **PowerModelsDistribution.jl**: an open-source framework for exploring distribution power flow formulations. Electric Power Systems Research, 189(December), 106664.
- Geth, F., Claeys, S., & Deconinck, G. (2020). **Nonconvex lifted unbalanced branch flow model**: derivation, implementation and experiments. Electric Power Systems Research, 189(December), 106558.
- Geth, F., Coffrin, C., & Fobes, D. M. (2020). A flexible **storage** model for power network optimization. ACM E-Energy, 1–6. http://arxiv.org/abs/2004.14768
- Claeys, S., Deconinck, G., & Geth, F. (2021). **Voltage-dependent load models i**n unbalanced optimal power flow using power cones. IEEE Trans. Smart Grid (Prepublished).
- Claeys, S., Vanin, M., Geth, F., & Deconinck, G. (2021). **Applications of optimization models** for electricity distribution networks. Wiley Interdisciplinary Reviews: Energy and Environment, e401.

# References on software


- Gao, D. W., Muljadi, E., Tian, T., & Miller, M. (2017). **Software comparison** for renewable energy deployment in a distribution network. Golden, CO. Retrieved from https://www.nrel.gov/docs/fy17osti/64228.pdf
- Meinecke, S., Braun, M., Meier, F., Scheidler, A., Schafer, F., Dollichon, J., … Menke, J.-H. (2018). **Pandapower** - An open-source python tool for convenient modeling, analysis, and optimization of electric power systems. IEEE Trans. Power Syst., 33(6), 6510–6521. https://doi.org/10.1109/tpwrs.2018.2829021
- Rigoni, V., & Keane, A. (2020). **Open-DSOPF** : an open-source optimal power flow formulation integrated with OpenDSS,
- Dugan, R. C., & McDermott, T. E. (2011). An open source platform for collaborating on smart grid research (**OpenDSS**). IEEE Power Energy Soc. General Meeting, 1–7. https://doi.org/10.1109/PES.2011.6039829
- Morstyn, T., Collett, K. A., Vijay, A., Deakin, M., Wheeler, S., Bhagavathy, S. M., … Mcculloch, M. D. (2020). **OPEN** : An open-source platform for developing smart local energy system applications. Applied Energy, 275(June), 115397. https://doi.org/10.1016/j.apenergy.2020.115397
- Fobes, D. M., Coffrin, C., Geth, F., & Claeys, S. (2020). **PowerModelsDistribution**. jl: an open-source framework for exploring distribution power flow formulations. Electric Power Systems Research, 189(December), 106664.
- Coffrin, C., Bent, R., Sundar, K., Ng, Y., & Lubin, M. (2018). **PowerModels.jl**: an open-source framework for exploring power flow formulations. In Power Syst. Comp. Conf. (Vol. 20, p. 8). Dublin, Ireland.
- Ergun, H., Dave, J., Van Hertem, D., & Geth, F. (2019). Optimal power flow for AC/DC Grids: formulation, convex relaxation, linear approximation and implementation (**PowerModelsACDC.jl**). IEEE Trans. Power Syst., 34(4), 2980–2990. https://doi.org/10.1109/tpwrs.2019.2897835
- Claeys, S., Vanin, M., Geth, F., & Deconinck, G. (2020). Applications of optimization models for electricity distribution networks. Submitted to WIREs Energy and Environment.
- Czekster, R. M. (2020). Tools for modelling and simulating the Smart Grid. ArXiv [Cs.PF].
- Henriquez-Auba, R., Lara, J. D., Roberts, C., Pallo, N., & Callaway, D. S. (2020). **LITS.jl** — An open-source Julia based simulation toolbox for low-inertia power systems. ArXiv, 1–8.
- Plietzsch, A., Kogler, R., Auer, S., Merino, J., Gil-de-Muro, A., Liße, J., … Hellmann, F. (2021). **PowerDynamics.jl** -- An experimentally validated open-source package for the dynamical analysis of power grids. Retrieved from http://arxiv.org/abs/2101.02103
- Vanin, M., Van Acker, T., D'hulst, R., & Van Hertem, D. (2020). A framework for constrained static state estimation in unbalanced distribution networks, (**PowerModelsDistributionStateEstimation.jl**) 1–8. Retrieved from http://arxiv.org/abs/2011.11614

THE UNIVERSITY OF QUEENSLAND
AUSTRALIA
CREATE CHANGE

# Smart inverters in Australia

# Control laws

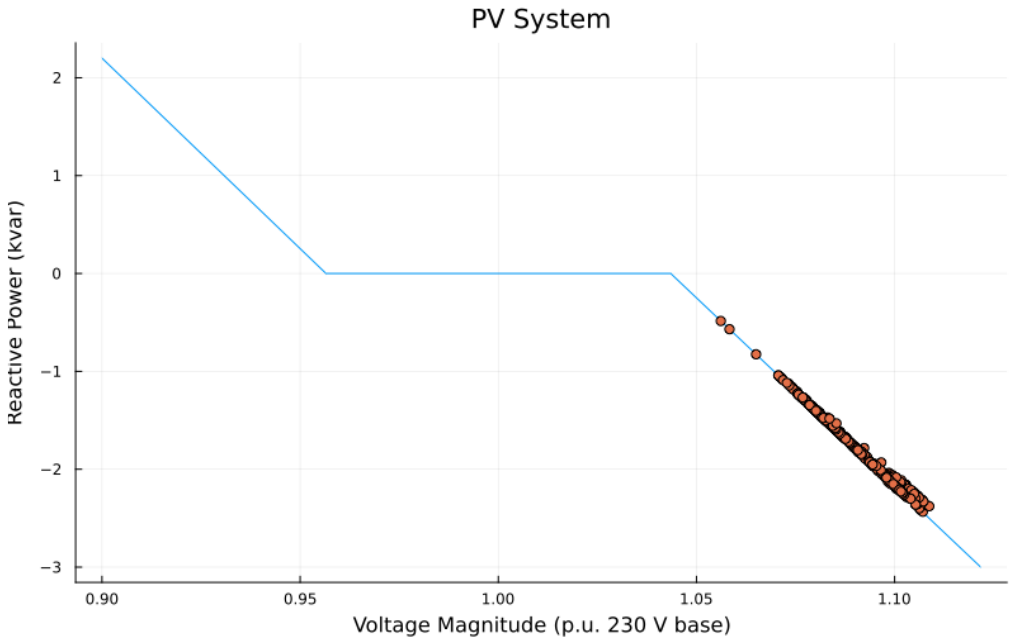
- Single-phase inverters control Q in the phase they are connected to
- Based on they see across the terminals
- Three-phase inverters take the average voltage magnitude of the 3 phases they see, which would be phase-to-phase typically
- Three-phase inverters without neutral are less expensive than with neutral
- Could be improved, doesn't allow room for unbalance compensation

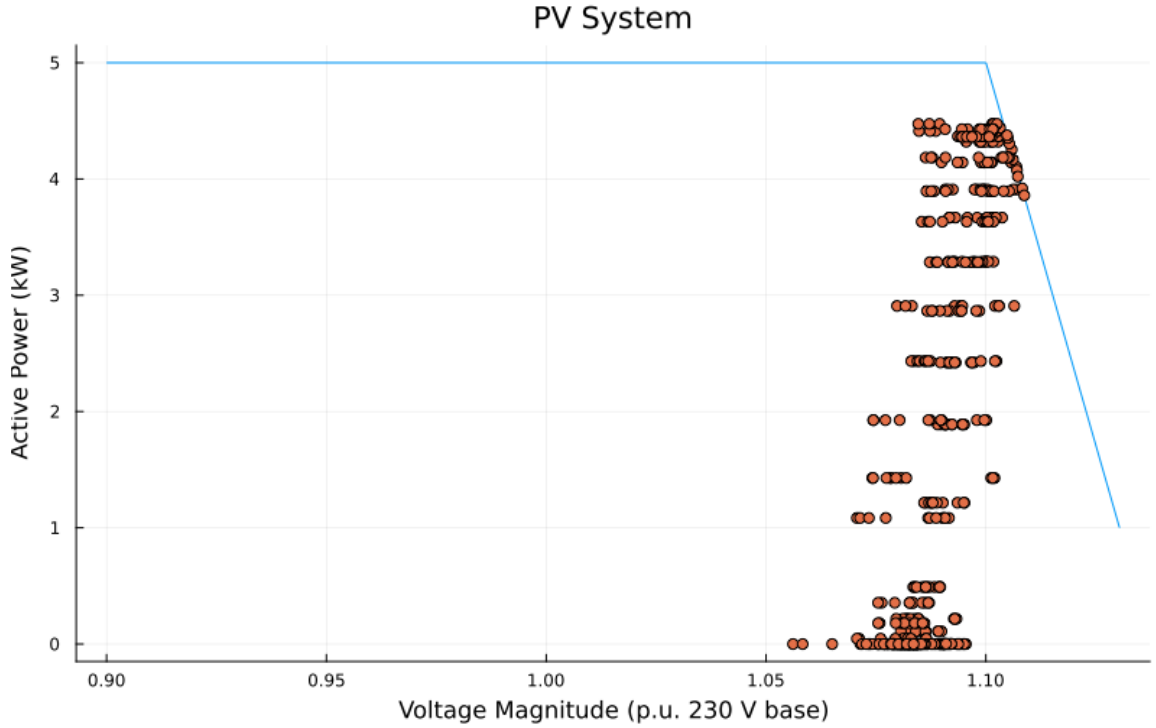
Volt–var and Volt–Watt Control Variations.

| Variation | Control |
|---|---|
| Phase-to-neutral | $\mathrm{P}_{g,an} = f^{VW}(U^{\mathrm{mag}}_{g,an})$<br>$\mathrm{Q}_{g,an} = f^{VV}(U^{\mathrm{mag}}_{g,an})$ |
| Phase-to-phase | $\mathrm{P}_{g,ab} = f^{VW}(U^{\mathrm{mag}}_{g,ab}/\sqrt{3})$<br>$\mathrm{Q}_{g,ab} = f^{VV}(U^{\mathrm{mag}}_{g,ab}/\sqrt{3})$ |
| Phase-to-phase averaged | $\mathrm{P}_{g,ab} = P_{g,bc} = P_{g,ca} = f^{VW}\left(\frac{U^{\mathrm{mag}}_{g,ab}+U^{\mathrm{mag}}_{g,bc}+U^{\mathrm{mag}}_{g,ca}}{3\sqrt{3}}\right)$<br>$\mathrm{Q}_{g,ab} = Q_{g,bc} = Q_{g,ca} = f^{VV}\left(\frac{U^{\mathrm{mag}}_{g,ab}+U^{\mathrm{mag}}_{g,bc}+U^{\mathrm{mag}}_{g,ca}}{3\sqrt{3}}\right)$ |
| Phase-to-ground averaged | $\mathrm{P}_{g,a} = P_{g,b} = P_{g,c} = f^{VW}\left(\frac{U^{\mathrm{mag}}_{g,a}+U^{\mathrm{mag}}_{g,b}+U^{\mathrm{mag}}_{g,c}}{3}\right)$<br>$\mathrm{Q}_{g,a} = Q_{g,b} = Q_{g,c} = f^{VV}\left(\frac{U^{\mathrm{mag}}_{g,a}+U^{\mathrm{mag}}_{g,b}+U^{\mathrm{mag}}_{g,c}}{3}\right)$ |
| Phase-to-neutral averaged | $\mathrm{P}_{g,an} = P_{g,bn} = P_{g,cn} = f^{VW}\left(\frac{U^{\mathrm{mag}}_{g,an}+U^{\mathrm{mag}}_{g,bn}+U^{\mathrm{mag}}_{g,cn}}{3}\right)$<br>$\mathrm{Q}_{g,an} = Q_{g,bn} = Q_{g,cn} = f^{VV}\left(\frac{U^{\mathrm{mag}}_{g,an}+U^{\mathrm{mag}}_{g,bn}+U^{\mathrm{mag}}_{g,cn}}{3}\right)$ |
| Phase-to-ground | $\mathrm{P}_{g,a} = f^{VW}(U^{\mathrm{mag}}_{g,a})$<br>$\mathrm{Q}_{g,a} = f^{VV}(U^{\mathrm{mag}}_{g,a})$ |

# Volt-var Watt

AS/NZS4777.2:2020 is a new standard for small-scale inverters that became mandatory for all new installations in Australia on 18 December 2021.




Volt-var *on* the curve

Volt-Watt below or on the curve

# Volt-var/Watt is mandatory in Australia AS4777.2

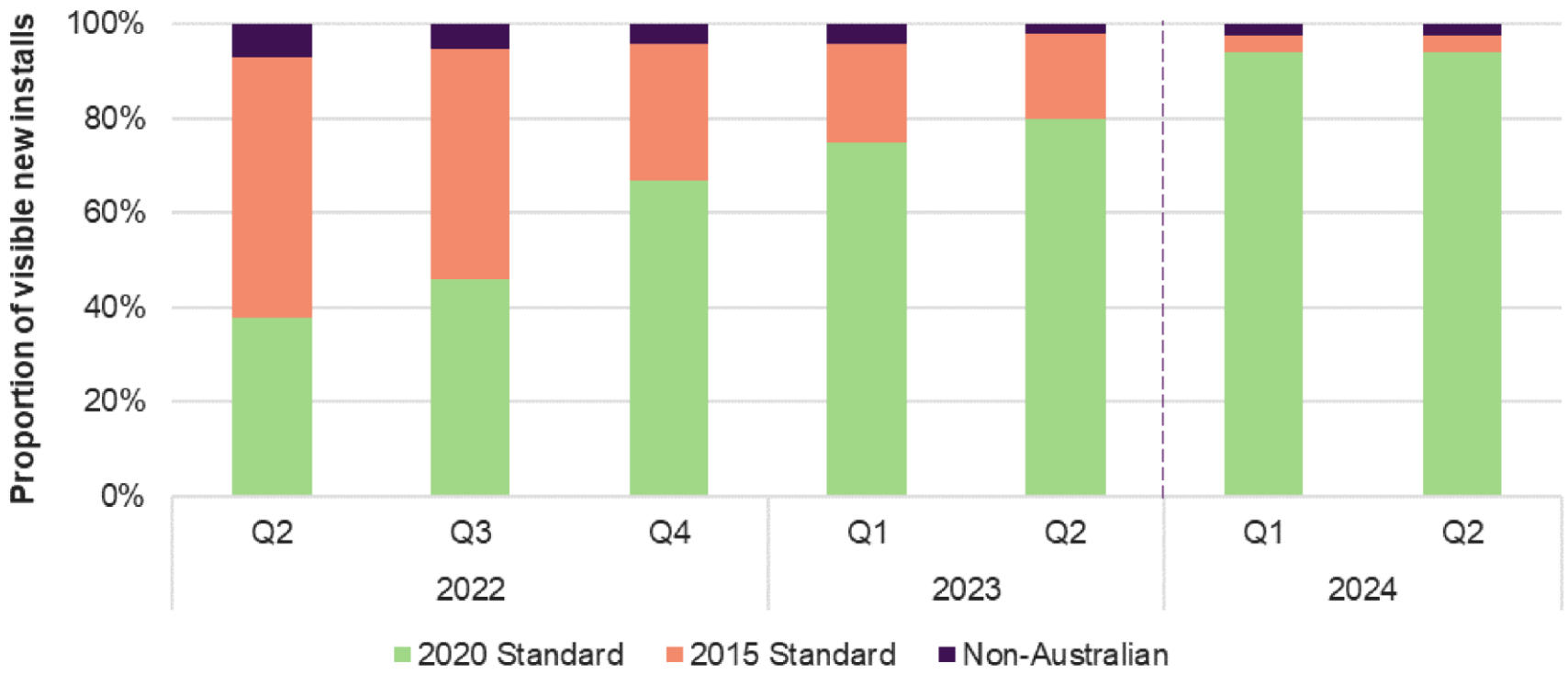
- Historically, PV systems operated at constant power factor (0.95, 0.98 lagging), often unity power factor
  - Lagging brings down the voltage

- Volt-var/Watt improves network voltage
  - End-users effectively contribute to voltage management
  - Implemented as control loops in standard PV inverters
  - Settings (break-points) should not be modifiable by user, but follow the standard

- Continuous, proportional behavior
  - In the past, there was mandatory disconnect on/off
- Postpones network investment if most (all) systems have VV/VW functionality with the same settings

# Compliance

**Figure 1 Estimates of quarterly compliance to the AS/NZS4777.2:2020 grid code**




https://www.aemo.com.au/initiatives/major-programs/nem-distributed-energy-resources-der-program/standards-and-connections/compliance-of-der-with-technical-settings

"2025 Update: Compliance of DER with Technical Settings"

Note: the OEMs included in the aggregation vary by quarter, based on which OEMs responded to the data request. Market coverage associated with these OEMs varies from 52% to 95%. Compliance estimates in this plot are scaled by OEM market share in that calendar year and assume that the visible portion of an OEM's fleet is representative of their entire fleet.

# Deployment challenges

# Current process to enable DOE deployments


- Building blocks to enable the minimum viable product

# Challenges and opportunities


- Smart meter data remains only usable off-line
- IEEE 2030.5 enabled inverters can also be a real-time source of measurement data
- Data models and standards for describing measurements
- Integration with DERMS

# Challenges and opportunities


- Requires up to date network models
- Network model calibration
- Network model validation
- Network model maintenance

# Challenges and opportunities


- Should be based on actual congestion, not congestion due to modeling or data artefacts
- Should be integrated w.r.t. network voltage management
  - Boosting voltage can throw solar of the network, ie reduce export limits
- Benchmarking
- Day-ahead estimates of DOEs for market (participants)
- Customer rights?

# Challenges and opportunities


- Communicate to retailers / VPPs / aggregators / markets
- Aggregate up to the TNSP/DNSP connection
- Fallback values in case communication drops
- Compliance? Cross checks between smart meter and inverter data