

Tulipa

Applying Software Best Practices

ESI-far Workshop – Day 2

Netherlands eScience Center | Amsterdam



Short Introduction of Myself

- Born and raised in Medellín, Colombia
- Bachelor's degree in Electrical Engineering
- MSc in Operations Research (OR)
- PhD in Power Systems. Thesis on *co-optimization of long-term and short-term energy storage planning*.
- Experience:
 - Industry: XM - Colombian ISO, Endesa - Spanish Utility Company, AFRY - Swedish Consultant company
 - Research: Universidad Pontificia Comillas, TNO

<https://www.linkedin.com/in/diego-tejada-phd-bb288471/>



Source: <https://easyliving.com.co/en/place/plaza-botero/>

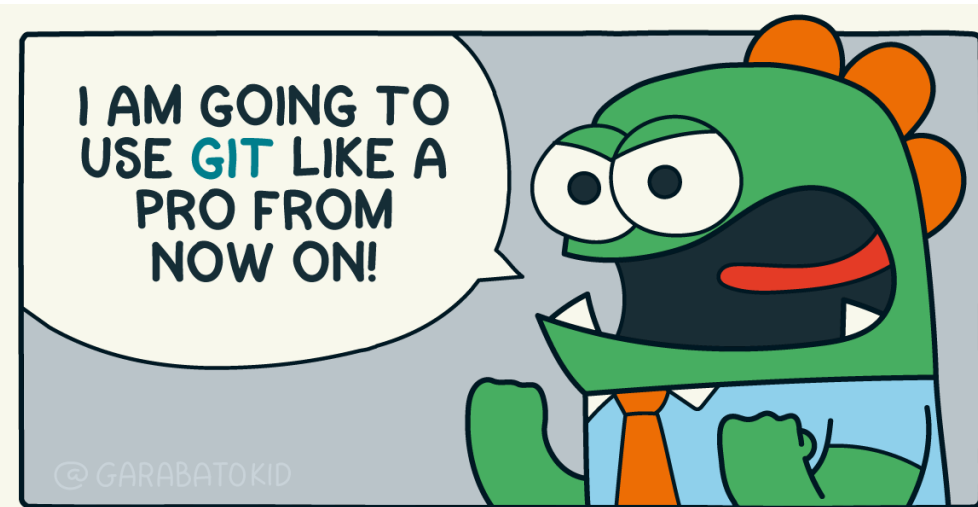
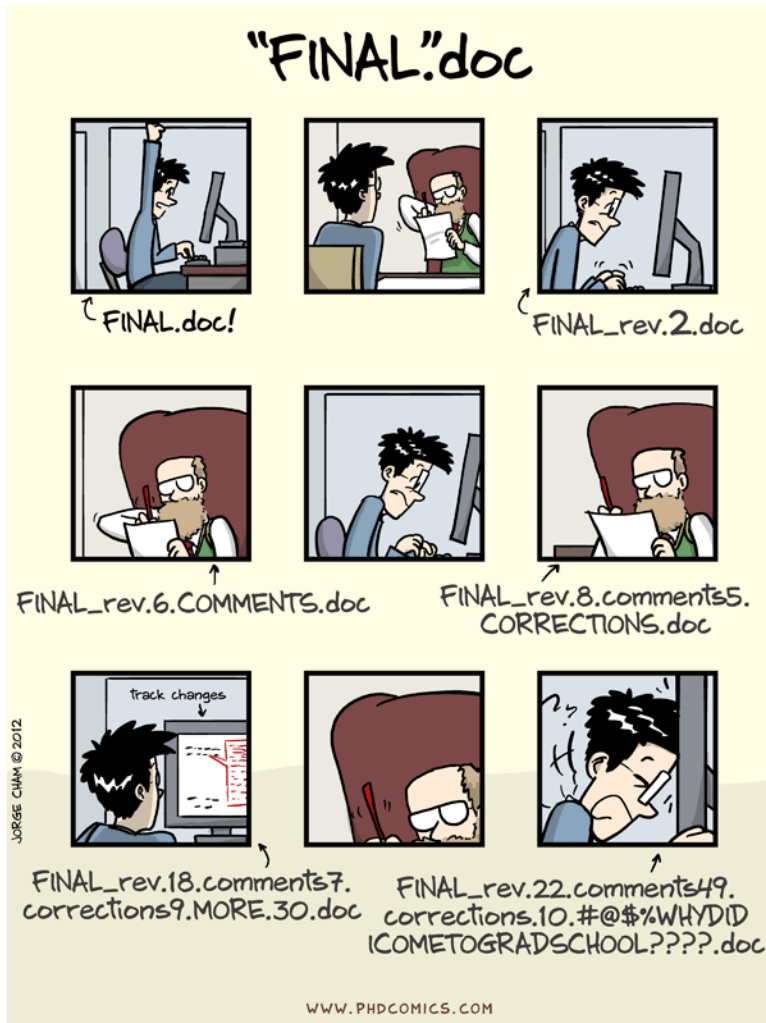
An exhibition currently in the EU: <https://www.euronews.com/culture/2023/03/24/artwork-by-renowned-colombian-artist-fernando-botero-on-display-in-valencia>

Agenda



- Motivation
- Tulipa Energy Model
- Best Practices

The Jokes...



ONE HOUR LATER



The Cruel Reality at TNO – COMPETES

- Version control is the name of the folder
- The developers access the code in a server, all using the same account
- Documentation is one page in a Word document (somewhere...)
- No code reviews
- Not thoroughly tested

> COMPETES versions >

Name

- ✗ _DemandResponse Competes (2020)_Ricardo
- ✗ _DemandResponse Competes (2022)_WACC
- ✗ _DemandResponse Competes (2023)_vRETROFIT
- ✗ _DemandResponse Competes (2023)_vRETROFIT - Base coupling
- ✗ _DemandResponse Competes (2023)_vRETROFIT - DR PV
- ✗ _DemandResponse Competes (2023)_vRETROFIT - OPETS_CO2_dispatchables
- ✗ _DemandResponse Competes (2023)_vRETROFIT_others
- ✓ _DemandResponseCompetes(2020)_KIPBCOR
- ✗ NL Installed Capacity (+heat).xlsx
- ✗ NL Installed Capacity Decentralized (+heat).xlsx



New Challenges for Energy System Modelling

- More sector coupling in the future
- Even more large-scale optimisation problems
- More operation constraints to take into account
- Highly renewable shares imply exposure to more uncertain events
- Transparency on formulations and code
- Ah! And getting the results in less time



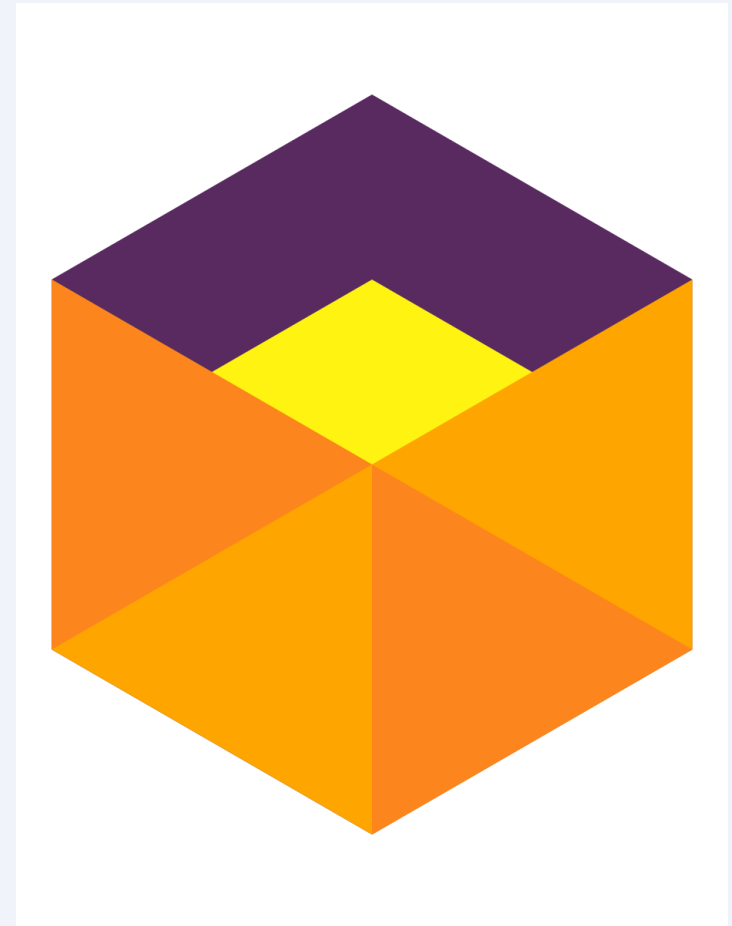
Agenda



- Motivation
- **Tulipa Energy Model**
- Best Practices

General Description

- New energy model from Scratch (model design and coding)
- Sector coupling: e.g., electricity, H2 and heat
- The main objective is to determine the optimal investment and operation decisions
- Representation of different types of energy assets (e.g., producers, consumers, conversions, storages, and transports)



Aiming for Computational Efficiency

Graph theory for a more adaptable model representation

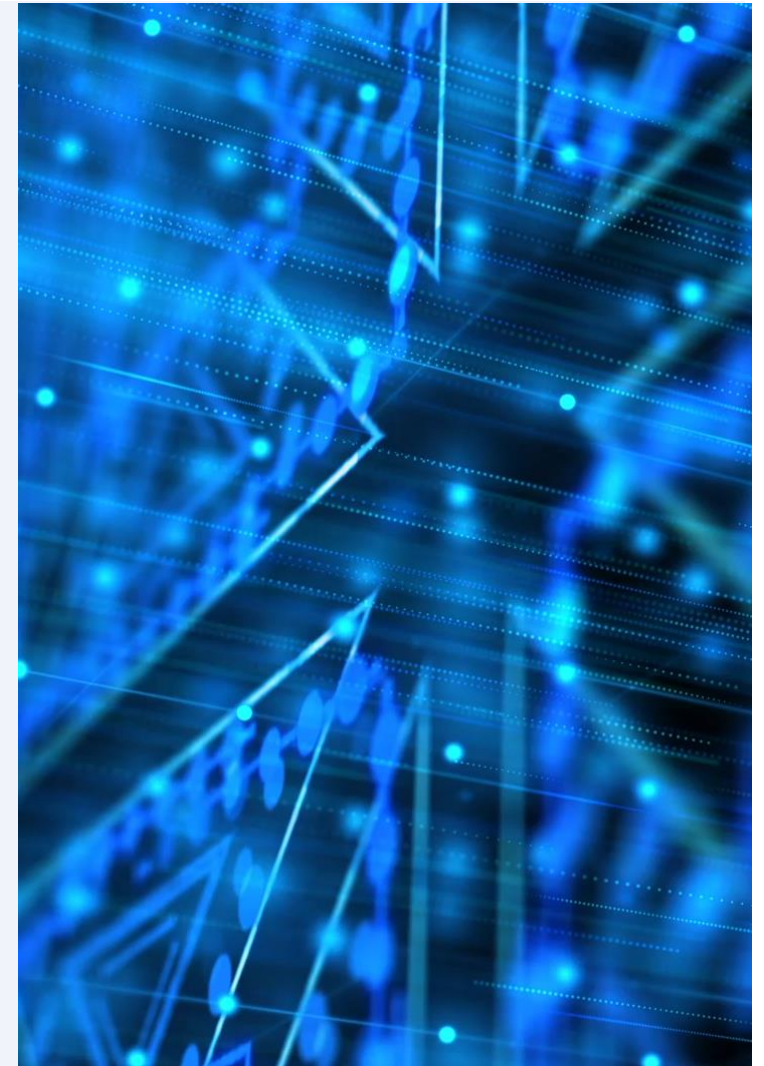
- Vertices → Energy assets (production, demand, conversion, storage, balancing nodes)
- Edges → Flows between assets (transmission lines, pipelines)
- Fewer variables and constraints
- **More flexibility - but more abstract in its modelling**

Flexible temporal resolution

- Allows multiple temporal resolutions that are not multiples of each other
- PhD student from Utrecht University is currently working on this research topic

Representative periods without losing accuracy

- Allows utilizing linked representative periods for short-term and seasonal storage
- Post-doc from TU-Delft has made improvements to the representative selection process (i.e., clustering).



The preprint paper with the mathematical formulation is available here:
<https://arxiv.org/abs/2309.07711>

How do We Avoid Previous Mistakes?

- Our team is strong in:
 - Knowledge and analysis of energy system modelling
 - Experience solving large-scale optimisation problems
 - Collaboration with Universities (research projects)
- Our team is *not so good* at software development:
 - Version control
 - Review and testing the code
 - Documentation
 - Coding formatting



WHO
YOU
GONNA
CALL?



TM & © 2019 CPIL. All Rights Reserved.

Artificial
Intelligence

Analytics

Data
Processing



National center of expertise for Research Software

✓ \approx 80 Research Software Engineers

✓ For all Dutch universities, institutes, disciplines

Computing

Software
Quality

But more than that...

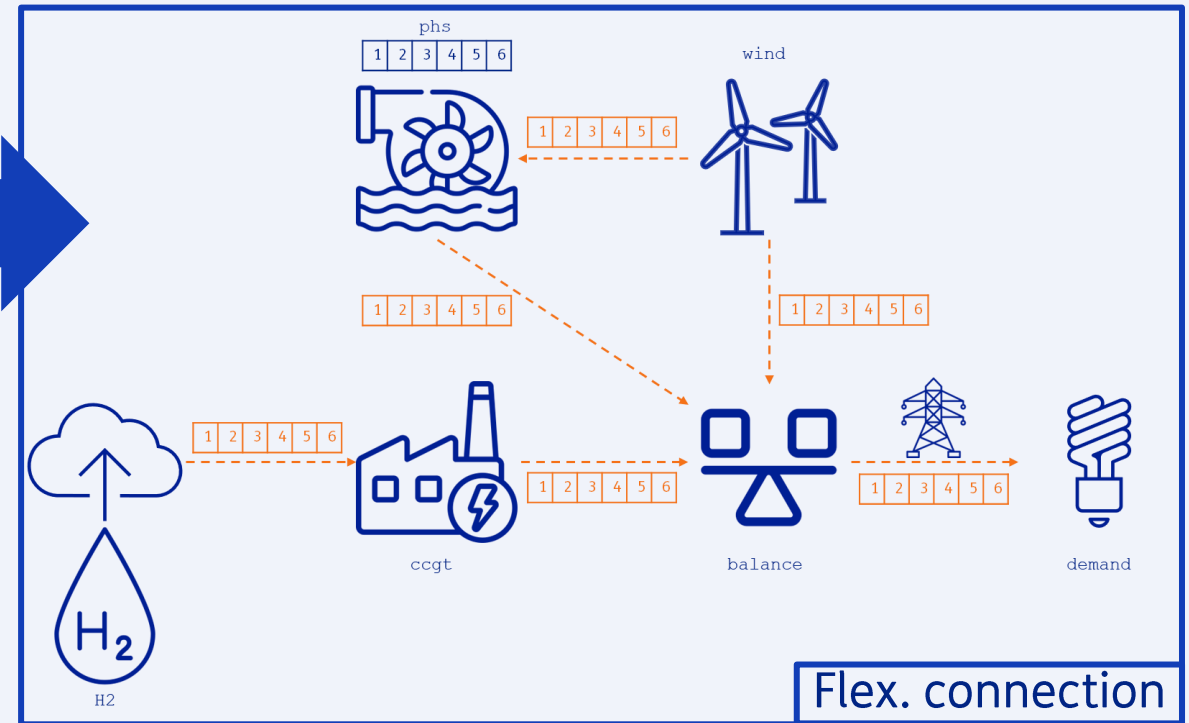
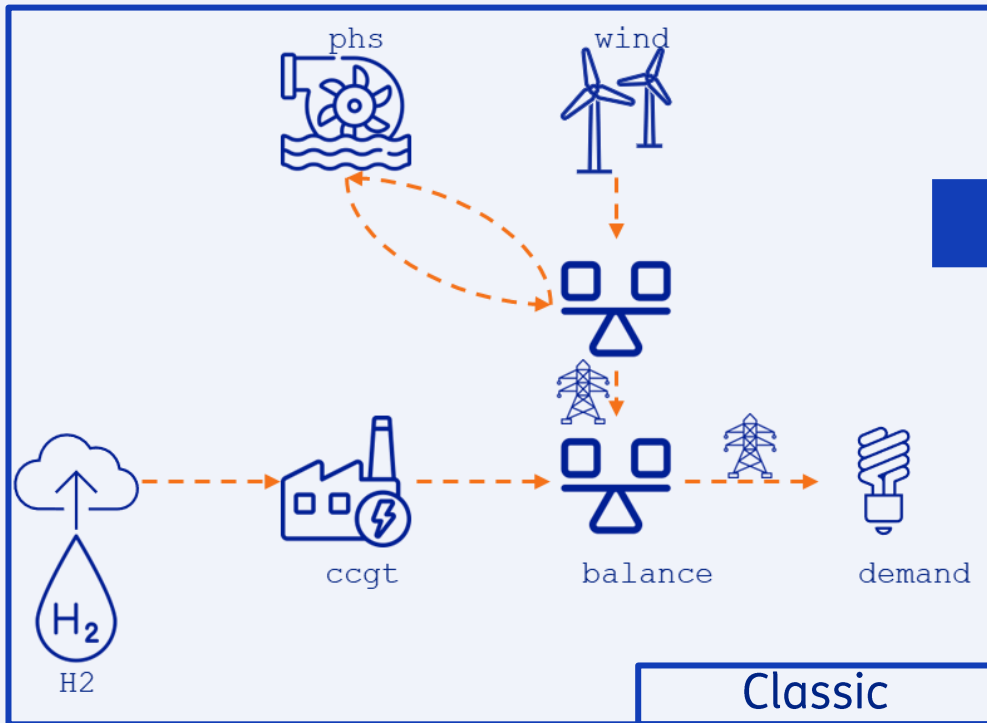


TulipaEnergyModel.jl

- Open-source Julia/JuMP package available on GitHub
- Timeline:
 - **2023** → Core features development
 - **2024** → Multi-year investment and power system operation constraints
 - **2025** → Operation constraints in other sectors (e.g., gas) and uncertainty
- **Applying best practices for software development** (e.g., atomic commits, semantic versioning, code review, tests, documentation)

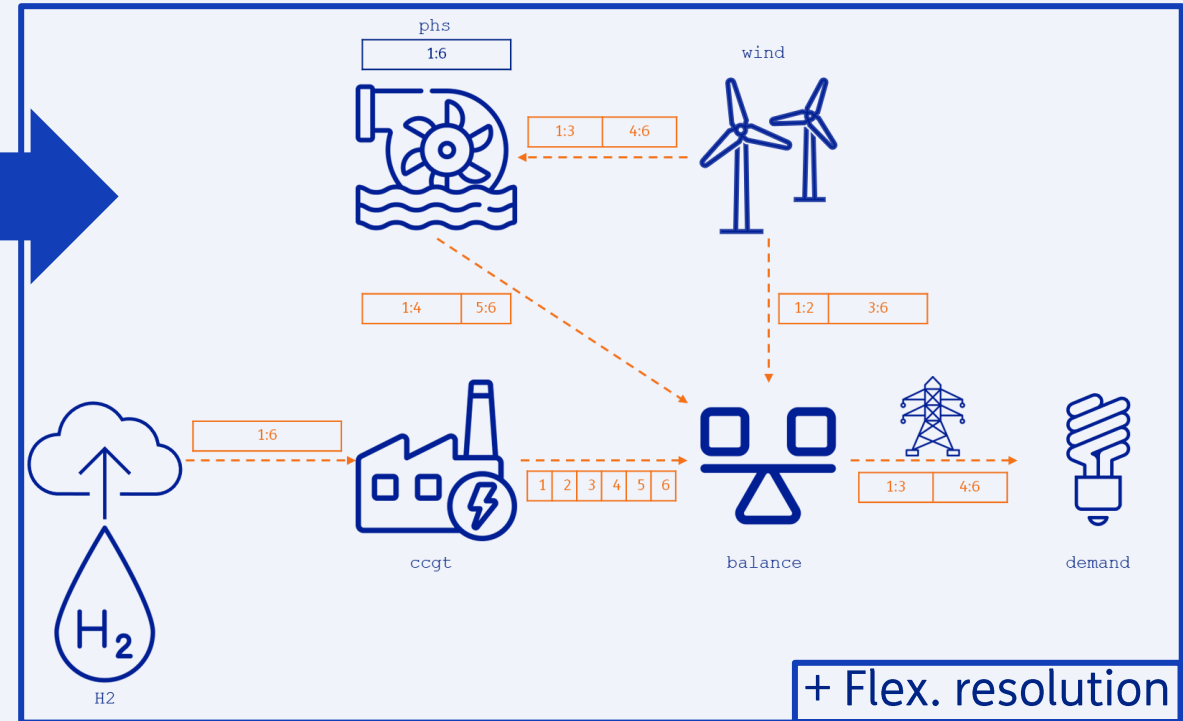
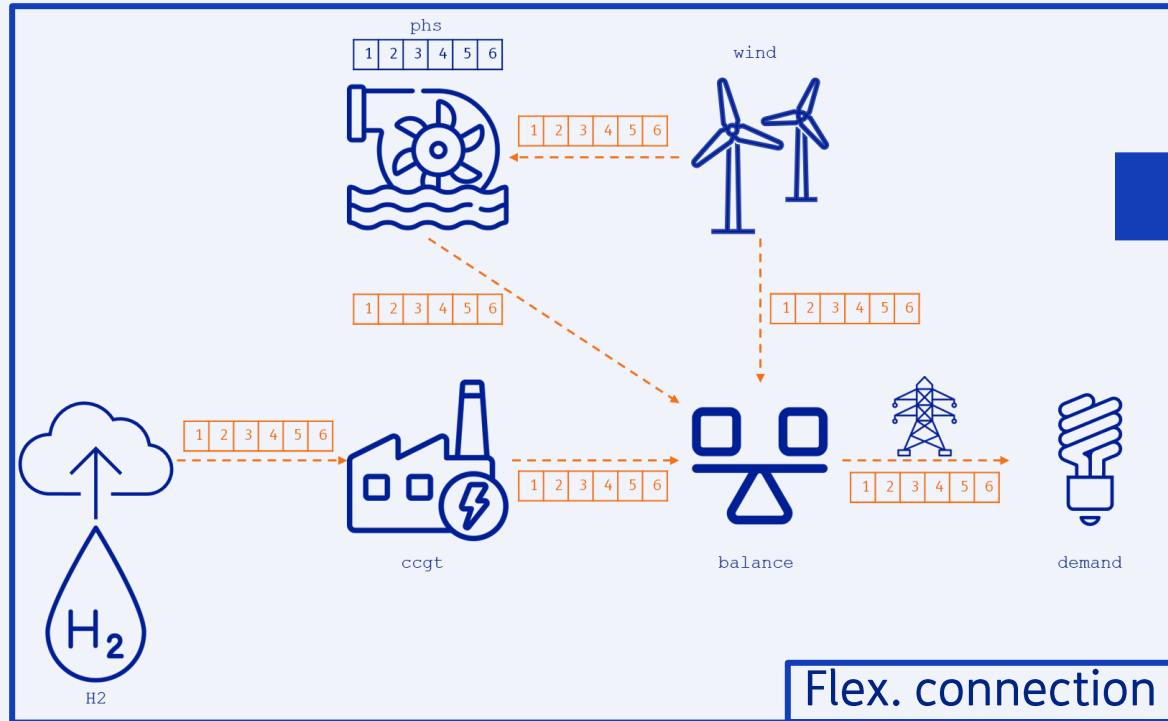


Flexible connection



	Variables	Constraints	Tot. Costs (k€)	Simplex iterations
Classic	48	84	28.44	8
Flex. connection	42 ↓12.5%	72 ↓14%	28.44	2 ↓4x

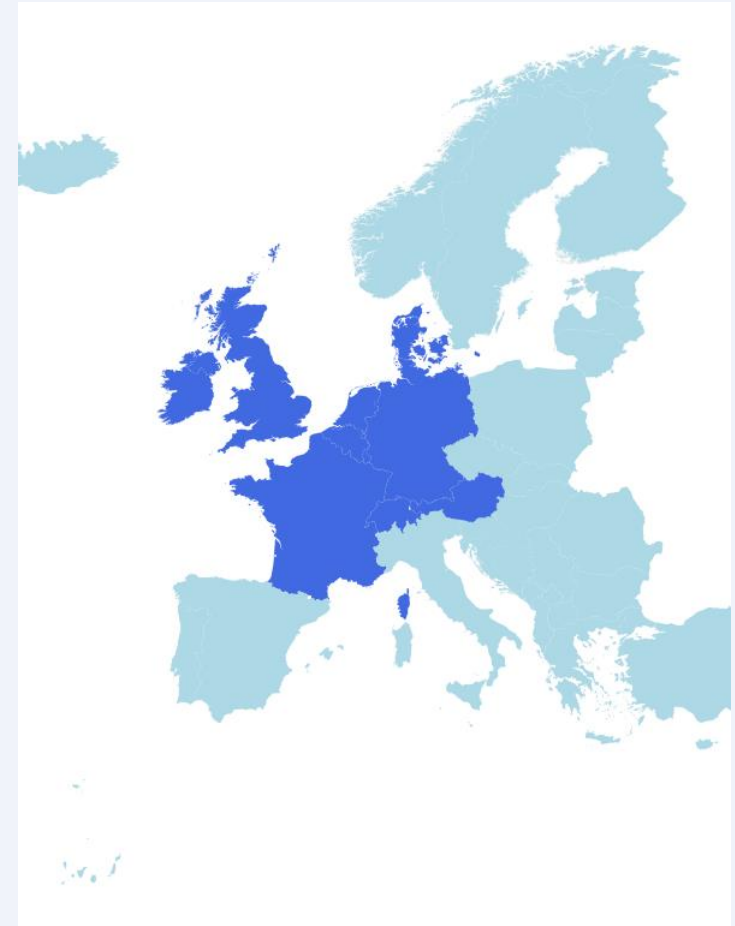
Flexible connection + flexible resolution



	Variables	Constraints	Tot. Costs (k€)	Simplex iterations
Classic	48	84	28.44	8
Flex. connection	42 ↓12.5%	72 ↓14%	28.44	2 ↓4x
+ Flex. resolution	16 ↓67%	25 ↓70%	28.46 ↑0.07%	1 ↓8x

Performance Results - Western European Countries

- 10 European Countries with an hourly resolution
- Minimize operating costs for one year
- Optimization problem size:
 - # variables: \approx 1.2 million
 - # constraints: \approx 2.5 million
- **TulipaEnergyModel.jl** building time[†] and memory usage:
 - Initial code (Basic JuMP): 314s and 32GB
 - Optimized code (Dataframes): 86s (**↓73%**) and 18GB (**↓44%**)
- The **eScience** support was vital to achieve this improvement



[†]First draft of the code: 8min for 2 EU countries

Agenda



- Motivation
- Tulipa Energy Model
- **Best Practices**

Requirements of a Software Management Plan

netherlands
eScience center



TNO Coding Principles

These principles ensure the quality of software developed at TNO and they govern all our coding work.



READABLE



Our code is easily readable by colleagues that see it for the first time

REUSABLE



Our code and/or parts of it are reusable

DOCUMENTED



Our code is appropriately documented

REVIEWED



Our code is appropriately reviewed

REPRODUCIBLE



Results that our code produces are reproducible

TESTED



Our code is properly tested

SHARED



Our code is developed and shared using version control software

i The 7 principles are relevant for any project where code is written. The principles leave room for context in implementation but should always be addressed to some extent.

codingguild.tno.nl

Readable



- We write code for people, not computers, prioritizing clarity over cleverness.
- We avoid complexity by structuring our code in functions and components.
- We use descriptive and unambiguous names for entities like variables, properties and functions.
- We use a consistent coding style, enforced by tooling like linters and formatters.

TulipaEnergyModel.jl / .JuliaFormatter.toml

abelsiqueira and clizbe Create function to read paramet

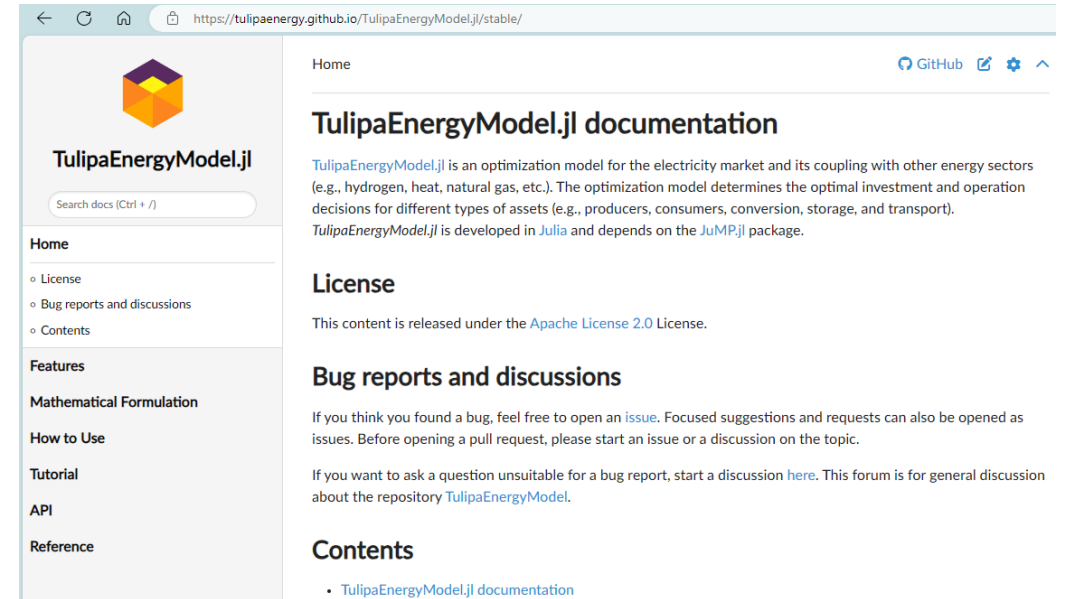
Code Blame 13 lines (13 loc) · 307 Bytes

```
1 align_assignment = true
2 align_matrix = true
3 align_pair_arrow = true
4 align_struct_field = true
5 conditional_to_if = true
6 format_docstrings = false
7 format_markdown = false
8 indent = 4
9 margin = 100
10 normalize_line_endings = "unix"
11 remove_extra_newlines = true
12 separate_kwargs_with_semicolon = true
13 verbose = true
```

Documented



- We document primarily for other people (but also for our future self).
- We comment our code to explain what we do (where necessary) and why we do it.
- We include at least a high-level description, installation instructions, usage instructions with examples and architecture details in our documentation.
- We keep our documentation updated with every (major) code update.



Reproducible



- We document all input and environmental details required to reproduce the same output.
- We work in isolated environments.
- We document which commits or versions were used for specific scenarios.

```
TulipaEnergyModel.jl / Project.toml

github-actions[bot] and CompatHelper Julia CompatHelper: add new compat ... 4630

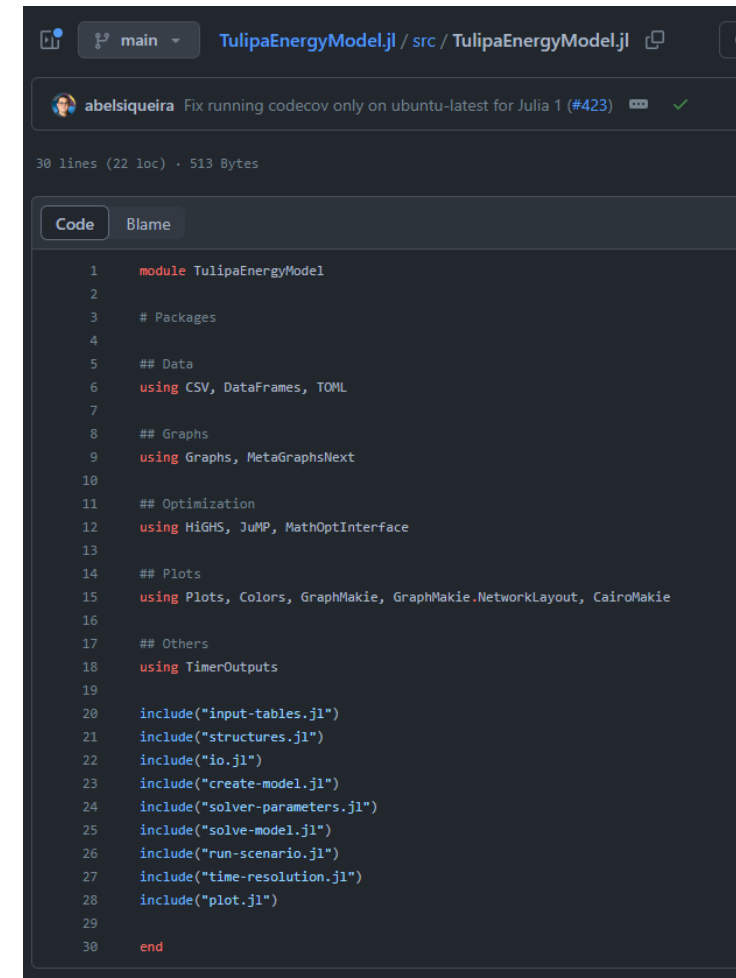
Code Blame 43 lines (39 loc) · 1.48 KB

1 name = "TulipaEnergyModel"
2 uuid = "5d7bd171-d18e-45a5-9111-f1f11ac5d04d"
3 authors = ["Diego A. Tejada-Arango <diego.tejadaarango@tno.nl>", "Germán Morales-España"]
4 version = "0.5.2"
5
6 [deps]
7 CSV = "336ed68f-0bac-5ca0-87d4-7b16caf5d00b"
8 CairoMakie = "13f3f980-e62b-5c42-98c6-ff1f3baf88f0"
9 Colors = "5ae59095-9a9b-59fe-a467-6f913c188581"
10 DataFrames = "a93c6f00-e57d-5684-b7b6-d8193f3e46c0"
11 GraphMakie = "1ecd5474-83a3-4783-bb4f-06765db800d2"
12 Graphs = "86223c79-3864-5bf0-83f7-82e725a168b6"
13 HiGHS = "87dc4568-4c63-4d18-b0c0-bb2238e4078b"
14 JuMP = "4076af6c-e467-56ae-b986-b466b2749572"
15 MathOptInterface = "b8f27783-ece8-5eb3-8dc8-9495eed66fee"
16 MetaGraphsNext = "fa8bd995-216d-47f1-8a91-f3b68fbeb377"
17 Plots = "91a5bcdd-55d7-5caf-9e0b-520d859cae80"
18 SparseArrays = "2f01184e-e22b-5df5-ae63-d93ebab69eaf"
19 TOML = "fa267f1f-6049-4f14-aa54-33bafae1ed76"
20 TimerOutputs = "a759f4b9-e2f1-59dc-863e-4aeb61b1ea8f"
```

Reusable



- We follow the principle of separation of concerns, at least in terms of code, data and configuration.
- We make environmental details configurable and avoid hard-coding.
- We will consider generalizing new functionalities.



```
1 module TulipaEnergyModel
2
3 # Packages
4
5 ## Data
6 using CSV, DataFrames, TOML
7
8 ## Graphs
9 using Graphs, MetaGraphsNext
10
11 ## Optimization
12 using HiGHS, JuMP, MathOptInterface
13
14 ## Plots
15 using Plots, Colors, GraphMakie, GraphMakie.NetworkLayout, CairoMakie
16
17 ## Others
18 using TimerOutputs
19
20 include("input-tables.jl")
21 include("structures.jl")
22 include("io.jl")
23 include("create-model.jl")
24 include("solver-parameters.jl")
25 include("solve-model.jl")
26 include("run-scenario.jl")
27 include("time-resolution.jl")
28 include("plot.jl")
29
30 end
```

Tested



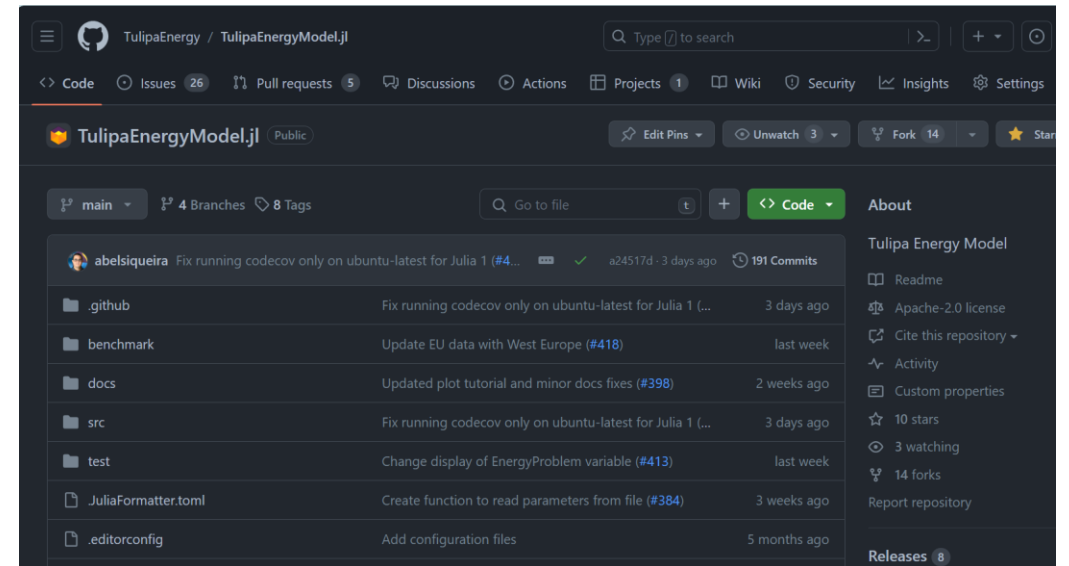
- We test key functionality of our code with every (major) change.
- We use automated testing as much as possible.



Shared



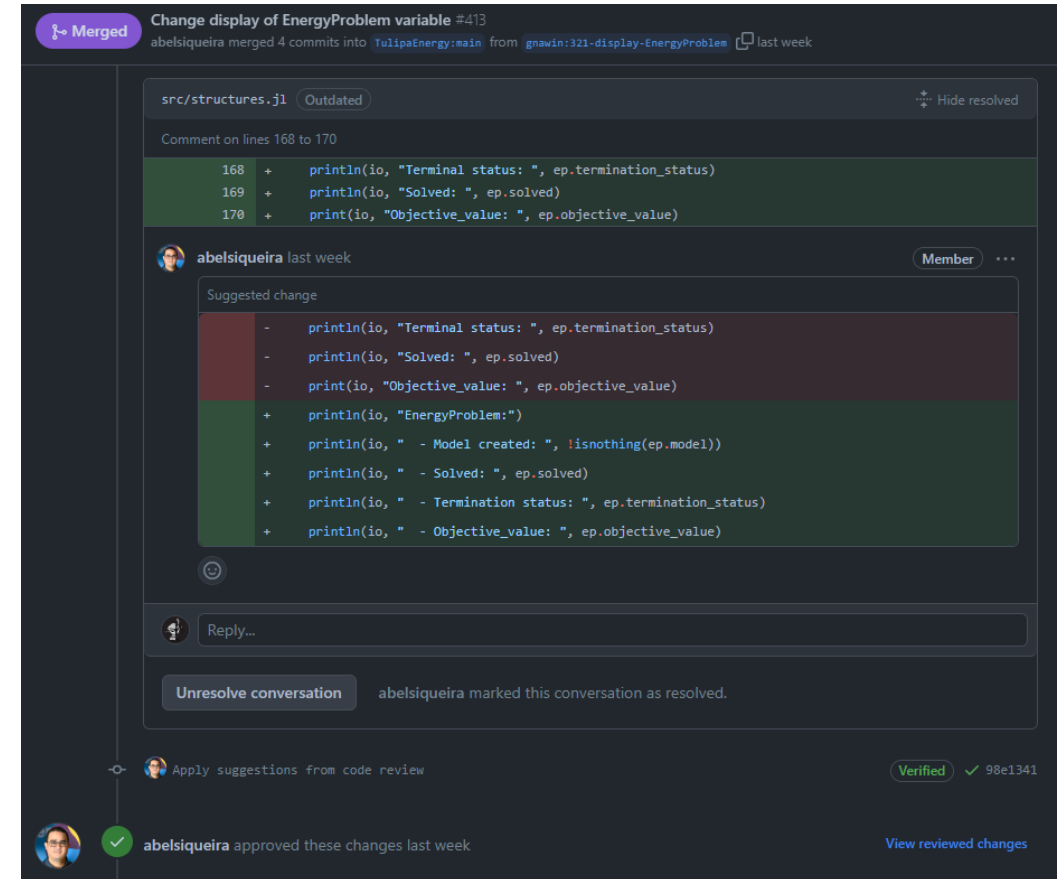
- We make our code accessible by sharing it via a version control platform like GitHub.
- We use source control for our documentation and consider this the single source of truth.
- We follow a git collaboration workflow.



Reviewed



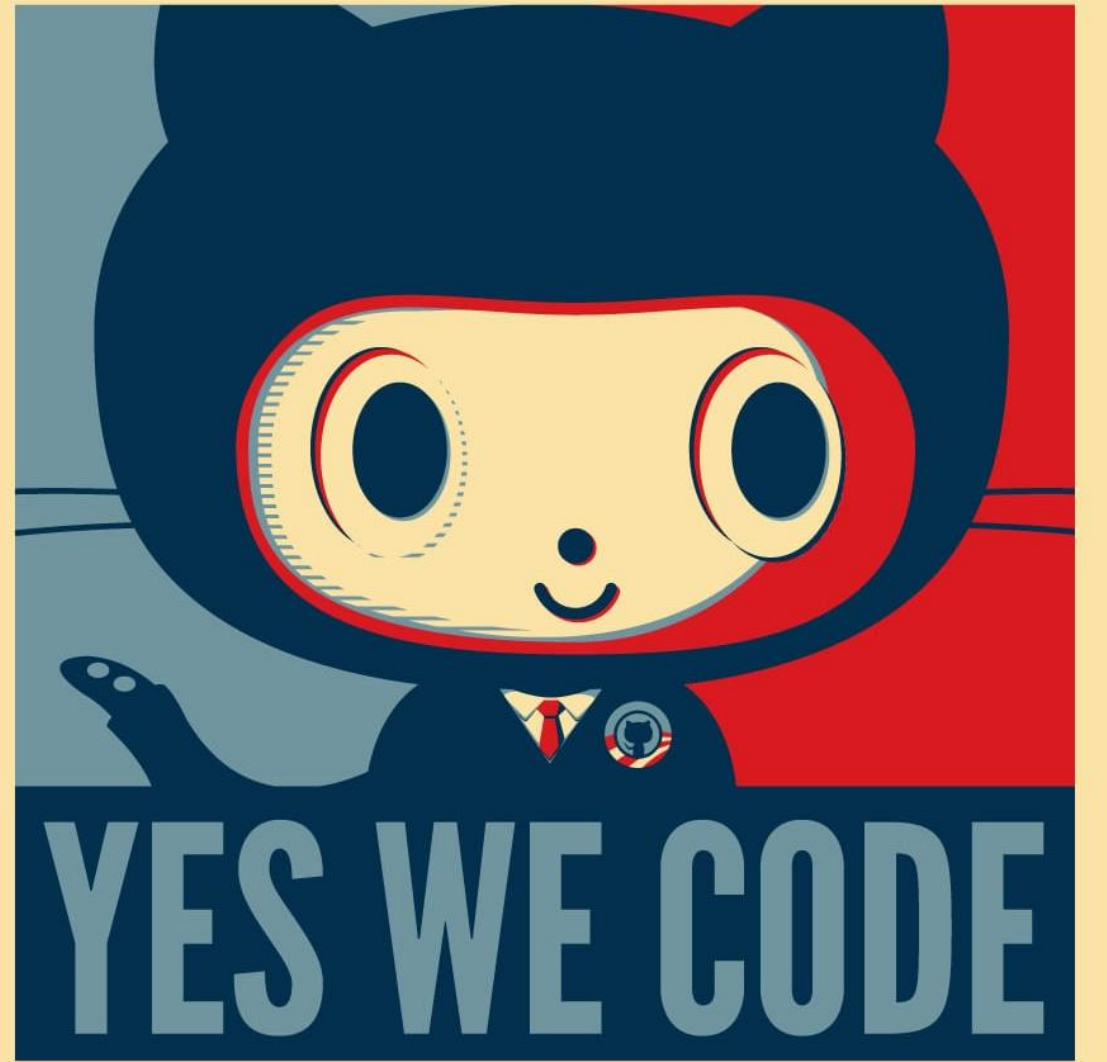
- We use review processes like pair programming, colleague reviews or team reviewing.
- We review our code before being published or before results produced by our code are used.
- We review our code to verify the coding principles are satisfied.



Let's have a Tour of Tulipa's Repo!



<https://github.com/TulipaEnergy/TulipaEnergyModel.jl>





TNO innovation
for life