

**PSI** Center for Nuclear Engineering and Sciences  
Center for Energy and Environmental Sciences



# Could **GAMSPy** be an option for **TIMES** code maintenance and development?

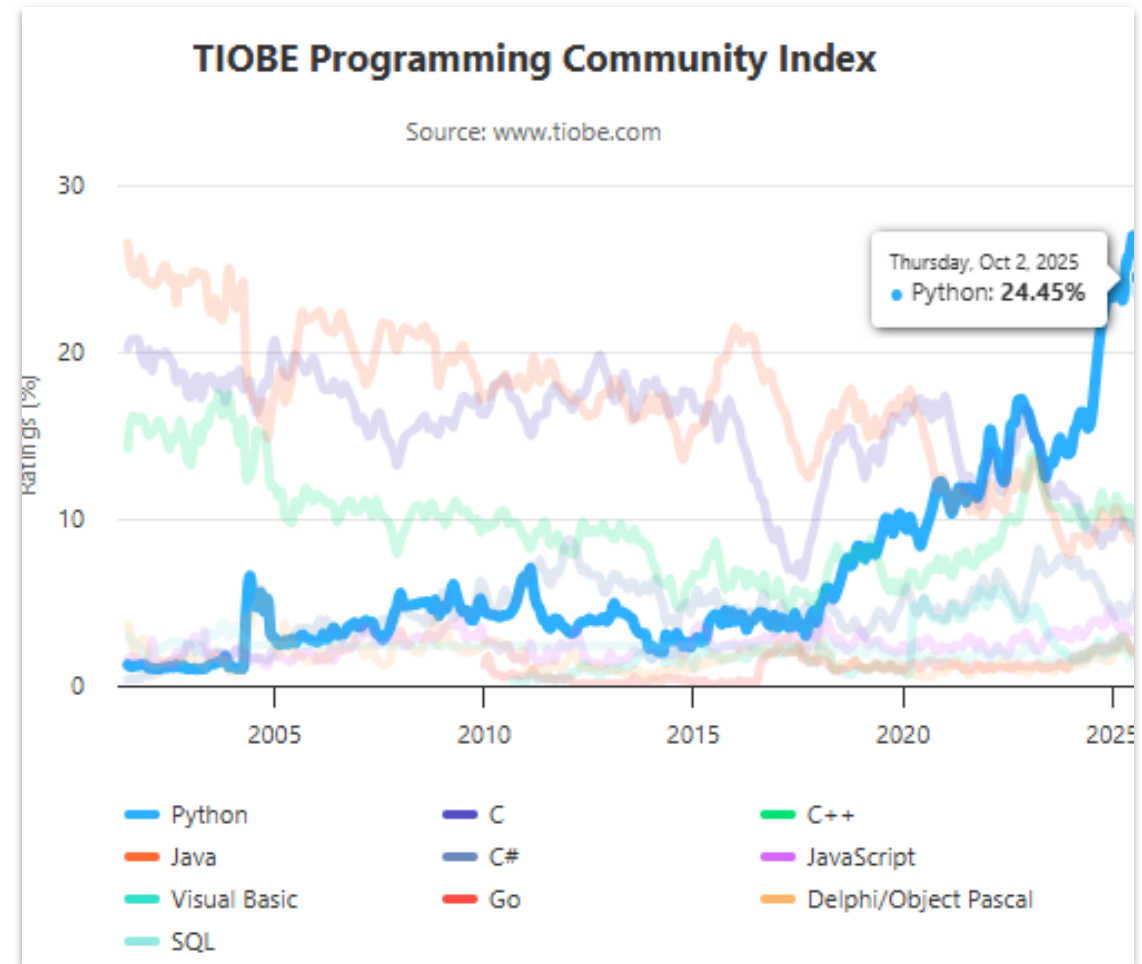
## Strengths & Limitations of the **GAMSPy**

Evangelos Panos (PSI) , Fred Fiand (GAMS GmbH)

ETSAP Winter Workshop, 24 November 2025

# Benefits of a Python port for TIMES

- Python is widely used and familiar to most graduates
- Provides Seamless access to the Python ecosystem for AI/ML, data tools and visualisations
- Lowers entry barriers for new contributors and future TIMES maintainers
- Expand the potential maintainer pool beyond GAMS specialists
- Increases likelihood of community contributions over time
- ..but Python's popularity does not automatically guarantee maintainers unless GAMSPy remains simple and well-designed

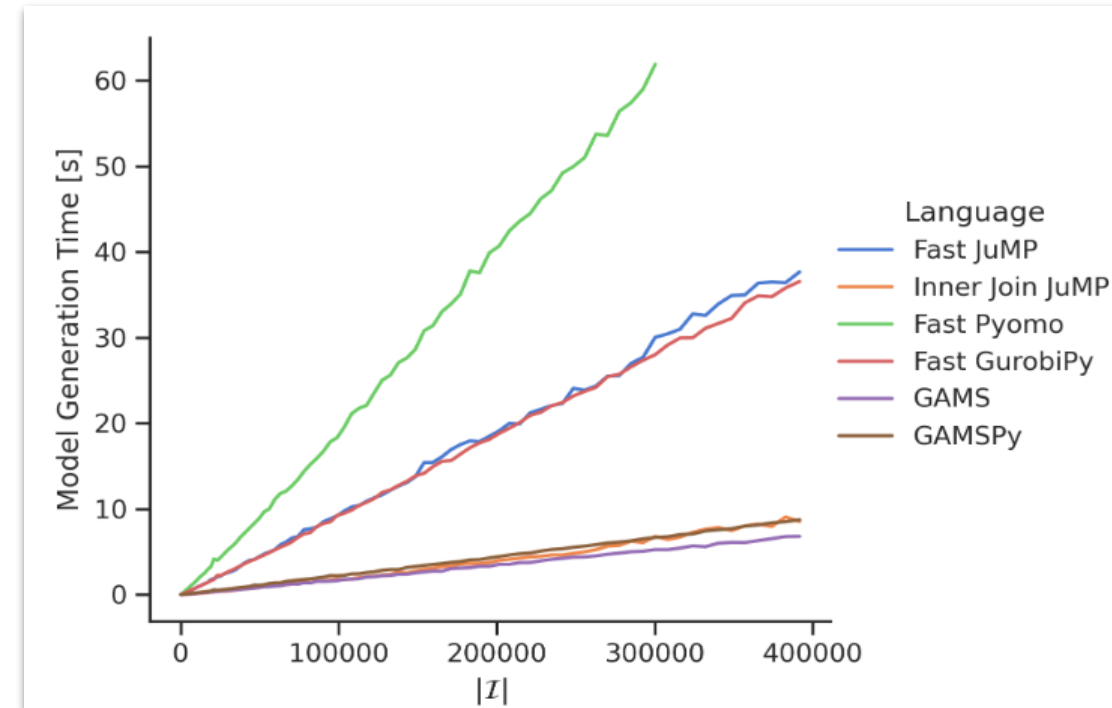


<https://www.tiobe.com/tiobe-index/>

# Why GAMSPy matters for large TIMES models

- GAMSPy is designed for **large-scale sparse models** mirroring GAMS performance
- GAMSPy preserves fast data execution and model generation times
- Fast generation is not for granted in other environments; it is the result of engineering in the GAMS toolchain
- TIMES2JuMP feasibility report confirms this challenge:

*“JuMP requires quite different code logical construction approach to match GAMS presolve performance”*



<https://www.gams.com/blog/2024/12/gamspy-high-performance-optimization-in-python/>

- Preserves GAMS-style sets, filters and model structure, **no need to rewrite the core modelling logic**
- Lets modellers focus on logic (sets, variables, equations) instead of low-level data structures
- Fully integrated with the GAMS ecosystem (solvers, MIRO, Engine)
- TIMES2JuMP report stresses the difficulty of rewriting set-based logic into vectorised structures:

*“Code logical construction needs to be very different to GAMS”*

```
transport.py

from gamspy import Container, Set, Parameter, Variable, Equation, Model, Sum, Sense

capacities, demands, distances, cost = load_data()

m = Container()

i = Set(m, "i", description="plants", records=capacities["city"])
j = Set(m, "j", description="markets", records=demands["city"])

a = Parameter(m, "a", domain=i, description="supply of commodity", records=capacities[["city", "capacity"]])
b = Parameter(m, "b", domain=j, description="demand for commodity", records=demands[["city", "demand"]])
c = Parameter(m, "c", domain=[i, j], description="cost per unit shipment", records=cost)

x = Variable(m, "x", domain=[i, j], type="Positive", description="amount of commodity to ship")

supply = Equation(m, "supply", domain=i, description="observe supply limit at plant i")
supply[i] = Sum(j, x[i, j]) <= a[i]

demand = Equation(m, "demand", domain=j, description="satisfy demand at market j")
demand[j] = Sum(i, x[i, j]) >= b[j]

obj = Sum((i, j), c[i, j] * x[i, j])

transport = Model(
    m, "transport", equations=m.getEquations(), problem="LP", sense=Sense.MIN, objective=obj
)

transport.solve(solver="CPLEX")
```

# GAMSPy: just another Python library




- Simple to install via `pip install gamspy`
- Python users stay in a familiar development environment
- Low learning curve, well-designed library, simple to understand & learn
- Higher chance of attracting and retaining maintainers
- Python programmers do not have to leave their familiar programming environment
- Solver-independent design: Just like GAMS, GAMSPy works with any compatible solver (as do Pyomo and JuMP)
- **Protects our investment:** when the solver landscape changes, no re-implementation is needed
- **Future-proof integration:** new solvers become available “automatically”, (see e.g. recent addition of NVIDIA’s GPU accelerated open source solver [cuOpt](#))

# What does it mean TIMES in GAMSPy?





- ?** *Does GAMSPy eliminate the need for GAMS?* No, internally GAMSPy uses the GAMS execution system
- ?** *Where do I change TIMES code?* It is done in Python; nothing to do anymore with GAMS
- ?** *Where TIMES data processing happens?* Everything TIMES does now in GAMS will be done in Python
- ?** *Do I still VEDA for model setup, checks, ...?* Yes, and VEDA needs to support the GAMSPy
- ?** *Is TIMES code compatibility ensured?* Yes, and code conversion to GAMSPy can be semi-automated
- ?** *If TIMES is in GAMSPy will also be in GAMS?* Yes, for a transition period two code bases will exist
- ?** *Who has the IP of the conversion?* The Intellectual Property (IP) remains at ETSAP
- ?** *How the conversion affects me as a TIMES user?* For most users will make no difference; for advanced users opens possibilities to combine TIMES with advanced data science, ML/AI and visualisation tools implemented in Python

## Academic Licenses are FREE, including solvers

-  GAMS Academic Portal offers FREE + full featured GAMSPy licenses including commercial solvers Copt, Cplex, Mosek, Xpress, and Gurobi-Link.  
→ This would make a GAMSPy TIMES implementation **available for free** to academics

## Commercial licenses

- **Fully complementary licenses:** Purchasing a GAMS license includes GAMSPy, and vice versa.
- Flexible licensing models for any environment:
  -  Network licenses: Secure connection to a central license server via the internet or a local network.
  -  Local licenses: Bound to a specific machine, fully operational offline.

## Commercial Licenses (Pricing Options): main constraint for ETSAP and should be part of the strategic decision

- **Paid-Up License:** Higher initial investment, low annual maintenance fees.
- **Annual Subscription License**
- Indicative Single-User Pricing (GAMS BASE + CPLEX, up to 12 cores)

License Type	Year 1	Year 2	Year 3	Year 4	Year 5	Total (5 Years)
<b>Paid-up License</b>	14 000 €	2 800 €	2 800 €	2 800 €	2 800 €	25 200 €
<b>Annual Subscription</b>	7 000 €	7 000 €	7 000 €	7 000 €	7 000 €	35 000 €

- Competitive package deals for **high-volume licenses** can be arranged with GAMS upon request (similar to Embedded Software Agreement offered by IBM for CPLEX within JuMP feasibility study).

5.1. Embedded Software agreement (ESA) €204k for 80 seats or €2,500 per user for CPLEX (2.5% the usual commercial cost)  
5.1.1. ETSAP can distribute ESA licences within within the TIMES code or within the community as the ExCo wishes.

# GAMSPy: Strengths & Limitations (Recap)



## Strengths:

- Maintainers can be recruited from the large pool of Python developers
- Minimal incompatibility risk: the logic stays in the GAMS paradigm
- Optimised for large sparse models → ensures fast execution
- GAMS Execution System ensures robust presolve and model generation
- Focus on model logic rather than low-level programming
- TIMES users can integrate Python-based data science, ML, AI, and visualisations tools
- Academic users get top-tier solvers for free → A full GAMSPy is provided for free for academics
- Can deliver a full port of TIMES and not a feasibility study

## Limitations:

- Requires a commercial GAMS license for non-academic users
- Relies on a proprietary backend → vendor lock-in risk

## Limitations also existing if another port from GAMS is chosen:

- Two TIMES codebases will need to co-exist during transition to ensure compatibility
- VEDA will need to be extended to fully support the workflow in the new port

# For Discussion: A Maintainable Path Forward for TIMES



ETSAP has three strategic options:

## 1. Stay in GAMS (Status Quo)

Technically robust but dependent on a shrinking pool of GAMS experts

- High continuity risk after Antti retires
- Limited integration with modern data/AI workflows
- Does not address the core challenge: future maintainability

## 2. GAMSPy (Python front-end, GAMS engine)

A low-risk, immediate conversion option that ensures uninterrupted TIMES maintenance

- Semi-automated conversion, minimal migration effort
- Preserves GAMS modelling logic and performance (via GAMS Execution System)
- Large Python developer pool → strong pipeline of future maintainers
- Enables integration with AI/ML, data science, and modern workflows
- Retains dependence on the proprietary GAMS backend, but operational continuity is ensured immediately
- GAMS is willing to offer high volume prices with a significant discount → ETSAP can negotiate

## 3. Full Rewrite in Python or JuMP/Julia

A fully open-source long-term vision, but slow and risky for near-term needs

- Requires a complete rewrite of the TIMES code base (multi-year effort)
- High engineering and performance uncertainty:
  - Pyomo: slow model generation and presolve for large sparse systems
  - JuMP: more modern but still behind GAMS in presolve efficiency
- Major architectural changes needed to replace GAMS sets & filters
- Full TIMES conversion is not feasible within 2026 and after Antti retires
- Suitable only as a long-term R&D pathway, not as an immediate solution

# Thank you for the attention!

Evangelos Panos

Head of Energy Economics Group

Laboratory for Energy Systems Analysis

[evangelos.panos@psi.ch](mailto:evangelos.panos@psi.ch)