

TIMES2JuMP – Feasibility Study of migrating the TIMES code from GAMS to Julia–JuMP

Prepared by Prof James GLYNN (ESMA, IRELAND), and Dr Siddharth Krishna (Open Energy Transition, Malawi).

AI disclosure statement: Anthropic's Claude Opus 4.8 was used in synthesising meeting notes, transcripts, code repositories and drafting initial text, within the manually defined outline from the project terms of reference. All text was reviewed by Prof James GLYNN, revised and edited as appropriate.

Email comments to Dr James GLYNN <james@esma.ie>.

Prepared for: The IEA-ETSAP Steering Committee for discussion

Date: June 2026.

1. Executive Summary

1.1. Performance

1.2. Migration Risks

1.3. Costs

1.4. Acceptance by the TIMES Community

1.5. Workplan and Budget Estimation

2. Computation Performance and Other Aspects

2.1. Efficiency of Model Generation

The prototype

Prototype performance

Main findings

2.1.1. Large Model Feasibility - Can JuMP transform large scale optimisation models in solver ready formats at a comparable speed to GAMS

2.2. Solver Performance – Can open source free solvers be used as an alternative to commercial solvers like CPLEX, Gurobi and XPRESS?

2.3. Solver Pricing – Purchasing power. Possibilities to acquire commercial solvers at lower costs should be investigated

3. Migration Risks

3.1. Implementation of TIMES in Julia JuMP vs GAMS

- [3.1.1. Open Source Community Support Vs Commercial Proprietary Support](#)
- [3.2. Backward Compatibility](#)
- [3.3. Ability to address logical constructions implemented in TIMES](#)
- [3.4. Address the full range of possible TIMES model functionality.](#)
- [3.5. Post processing of solver outputs](#)
- [3.6. TIMES Code maintenance risks under Julia–JuMP](#)
- [3.7. Strategic Risk: Julia vs. Python Developer Ecosystem](#)
- [4. Cost](#)
 - [4.1. Academic use Case](#)
 - [4.1.1. Assess commercial solvers costs for existing GAMS CPLEX users](#)
 - [4.1.2. Assess Initial commercial solvers costs](#)
 - [4.1.3. Assess annual maintenance costs for commercial solvers](#)
 - [4.1.4. Assess costs related to academic use of use of VEDA2](#)
 - [4.2. Non–Academic. Others* Commercial/Public Body/Government Use Case costs](#)
 - [4.2.1. Assess commercial solvers costs for existing GAMS CPLEX users](#)
 - [4.2.2. Assess Initial commercial solvers costs](#)
 - [4.2.3. Assess annual maintenance costs for commercial solvers](#)
 - [4.2.4. Assess costs related to academic use of use of VEDA2Annual costs related to commercial solvers](#)
 - [4.3. Initial Migration Development cost to Julia JuMP](#)
 - [4.4. Initial Migration cost to Julia JuMP for TIMES Users.](#)
 - [4.4.1. Licence Costs?](#)
 - [4.4.2. Point to Summary of work plan from below.](#)
 - [4.5. Cost Savings to some TIMES users using Free Open Solvers](#)
 - [4.6. Assess limitations of free solvers](#)
 - [4.6.1. OET Benchmark Study](#)
 - [4.6.2. DLR PIPS–IPM++ Study](#)
 - [4.6.3. NVIDIA's CuOpt as an emerging large model solver](#)
 - [4.7. Assess limitations of open source solvers](#)
 - [4.7.1. OET Benchmark Study](#)
 - [4.7.2. DLR PIPS–IPM++ Study](#)
- [5. IEA–ETSAP Virtual Community Round Table Dialogues](#)
 - [5.1. Backward compatibility with existing models](#)
 - [5.2. Acceptance of the TIMES users to migrate to a new framework](#)
- [6. Additional emergent points not in the TOR](#)
 - [6.1. CPLEX is no longer being developed.](#)
 - [6.2. Costs of CPLEX](#)

[6.3. Gurobi already faster than CPLEX for many benchmarked ESOM model instances](#)

[6.4. Role of AI in migration ? Is Python going to be even more dominant in the future?](#)

[7. Migration workplan](#)

[7.1. Project Workpackages](#)

[7.2. Gantt Chart timeline of workpackages, deliverables, tasks and milestones.](#)

[7.3. Migration Project Budget](#)

[Appendix: Consultation Participants](#)

1. Executive Summary

This report presents the findings of an IEA-ETSAP funded feasibility study assessing whether the TIMES energy systems model generator can be migrated from the General Algebraic Modelling System (GAMS) language to the Julia programming language and its JuMP mathematical optimisation package. The study was conducted over twelve months by Prof James Glynn (ESMA Ltd), Dr Olexandr Balyk ([TIMES.jl](#) prototype development), and Dr Siddharth Krishna, with expert input from JuMP developers Oscar Dowson and Benoît Legat, and from community dialogue input from IEA-ETSAP country delegates, the TIMES maintainer, TIMES Developers, TIMES Users and OpenMod community members who have migrated GAMS energy system model codes to JuMP.

The central finding is that migration is technically feasible. A working prototype (TIMES.jl) is under development and demonstrates that the core TIMES LP formulation can be expressed in Julia/JuMP. Performance parity with GAMS is judged achievable with idiomatic, performance-engineered Julia. The economic, policy, and community pressures for migration are real and growing. If the ETSAP Executive Committee recommends proceeding to full implementation, there is a phased workplan outlined in Section 7, including risk mitigation.

1.1. Performance

Julia and JuMP are technically capable of expressing the LP-class problems that TIMES encodes. The TIMES.jl prototype is under development: it implements the core LP equation families and has been exercised on a small-scale utopia model, with Oscar Dowson (JuMP lead developer) providing feedback on the code performance. The model-generation gap was originally diagnosed (early 2025) as conditional summation (“sum-if”) patterns over sparse sets implemented in a GAMS-style manner. The prototype has since been refactored toward idiomatic Julia — global variables removed, typed Dictionaries and DataFrame-based index joins adopted, and a profiler added — so the initial “in the region of 2× slower” estimate predates these changes and has not yet been re-benchmarked on the current code. Improving the idiomatic code towards concrete tuple types is a known, addressable step expected to bring model generation toward parity with GAMS, consistent with analogous JuMP benchmarks and the experience of SINTEF and other energy-modelling groups. A clean, feasible solve on a full reference TIMES Demo model and end-to-end numerical validation against GAMS remain outstanding work items at the time of writing. Julia/JuMP additionally offers

in-memory model-manipulation advantages unavailable in GAMS.

1.2. Migration Risks

The principal risk is that migration is a re-engineering effort, not a line-for-line translation. The TIMES GAMS code — particularly its compact pre-processing pipeline — must be re-implemented using performant Julia data structures. The expert consensus (Antti Lehtila, Benoît Legat and to a lesser degree Oscar Dowson) is that an automated, deterministic approach is preferred, and a tool such as the GAMS-to-JuMP converter Benoît Legat, or similar, is the preferred mechanism for this translation. The single greatest project risk is the availability of personnel combining TIMES domain knowledge, GAMS expertise, and idiomatic Julia/JuMP skill. Secondary risks include VEDA2/xl2times interface continuity and the management of a parallel GAMS, GAMS Py & Julia support window. Julia (since v1.0, 2018) and JuMP (since v1.0, 2022) are stable with no planned breaking changes, eliminating the version-instability concern raised in the previous 2017 assessment.

1.3. Costs

GAMS/Base costs \$1,750 for a single user and \$5,600 for 5 users per year. GAMS/Base is required to link TIMES.GAMS files to CPLEX and Gurobi solvers at their individual solver prices. GAMS/CPLEX pricing (provided by GAMS, 2025) places a five-user paid-up base licence at USD 44,800 in year one and USD 80,640 over five years with an annual average cost per user between \$3,226 and \$4,480. A single named user costs USD 14,000 in year one (paid-up) or USD 7,000 per year (subscription) with an annual average per user cost of \$5,040 – \$7,000 over a 5 year period. GAMS/Gurobi licences are only available on a per year basis with no long term maintenance discount. Gurobi one user per year is \$31,500 with concurrent cases on a machine with no more than 8 physical cores. 5 users costs \$157,500 per year for concurrent cases with no more than 8 physical cores. Gurobi_single with non concurrent cases costs \$17,200 for 1 seat per year and \$86,000 for 5 seats per year. Julia-JuMP, and the HiGHS solver are free of price to the user and open-source. For the largest TIMES models, commercial solver licences (CPLEX, Gurobi, etc.) would still be necessary until open-source solvers mature, but the GAMS base-system and GAMS solver-link costs could be eliminated entirely, and solvers can then be procured directly. CPLEX, Gurobi and other commercial solvers provide academic licences for free for academic research purposes. The full migration development programme is estimated at approximately €1.1–1.3 million over 18 months. The long-term outcome is a materially lower cost of entry for non-academic, policy, governmental, Global South users and with less but still

considerable cost reductions for academic users of TIMES. The presumption is that this lowers the barrier to entry and would help grow the TIMES user base to new regions and groups.

A summary of GAMS and Solver Licence costs are presented below:

**Comparing annual lease prices on a machine with up to 8 physical cores, prices are as follows:*

Licence/Solver	1_user (USD2026)	5_users (USD2026)
GAMS/BASE	1,750	5,600
CPLEX	5,250	16,800
Gurobi	31,500	157,500
Gurobi_Single_Use	17,200	86,000
Copt	3,500	11,200
Mosek	1,750	5,600
Xpress-MIP	1,750	5,600
HiGHS	0	0
PIPS-IPM++	0	0
CuOPT	0	0

1.4. Acceptance by the TIMES Community

Community sentiment, gathered through four structured virtual Round Table Dialogues and numerous bilateral consultations, is “agnostic to positive”. No participant expressed unconditional opposition. The primary driver for change is economic rather than ideological. Key conditions for acceptance are: (i) demonstrated numerical validation against GAMS across a diverse set of partner models; (ii) a clearly communicated and time-bounded parallel-support plan (tolerance is approximately one to two years); and (iii) assurance that the full range of TIMES functionality, not merely a “TIMES-light” subset, will ultimately be supported. Open-source mandates from European governments are an increasingly important external driver.

1.5. Workplan and Budget Estimation

The study outlines a phased, milestone-gated 18-month development programme delivered by a team of four senior specialists (one TIMES/GAMS domain expert, one

GAMS expert, two Julia/JuMP software engineers (or one JuMP engineer if very senior)), with advisory support from the JuMP core team. Each phase is gated on explicit performance and validation exit criteria. The total estimated budget, including a 15% contingency, is approximately €1.1–1.3 million. Full details are in Section 7.

1.6. Matters arising additional to Terms of Reference

A broader strategic question raised in consultations is whether the rapid maturation of AI coding tools — which are currently more effective on Python than on Julia, owing to Python’s larger training corpus — will entrench Python’s dominance and erode Julia’s relative performance advantage. The balanced view from the study is: Julia retains a genuine mathematical-programming and performance edge for advanced modelling, but Python has a larger, more mature community and stronger AI-tooling synergy. The recent emergence of JuMPy may narrow this trade-off by offering Python accessibility atop JuMP performance, and is a route to reconcile the two. Most TIMES users interface with the model via Excel input files, and so those who work mostly via excel, would be less sensitive to this functionality “under the hood”. This question should be revisited at the start of full development, as the AI-tooling landscape is evolving very quickly.

2. Computation Performance and Other Aspects

This section addresses the ToR’s computational performance dimension: the efficiency of model generation in JuMP relative to GAMS, the viability of open-source solvers, and the scope to acquire commercial solvers at lower cost.

2.1. Efficiency of Model Generation

Model generation — the translation of the TIMES database into a solver-ready LP matrix — is the phase where the GAMS-to-Julia paradigm shift matters most. GAMS is a declarative, set-oriented language that handles matrix generation and sparsity exploitation internally. Julia is a general-purpose language in which JuMP provides a declarative layer for constraint formulation, but all surrounding data manipulation is the programmer’s responsibility. Achieving performant model generation therefore depends on data structure choice, pre-processing algorithm design, and the manner in which constraint expressions are assembled. Even if JuMP is slower than GAMS, it is possible that JuMP writes the problem to memory more efficiently by removing some of the layers of writing files to and from disk, ultimately in a shorter

run-time.

The prototype

The TIMES.jl prototype (github.com/olejandro/TIMES.jl), developed by Dr Olexandr Balyk, implements a representative thin-slice of the TIMES model generator. It is LP only, targets single- and two-region configurations, and implements the core TIMES equation families, and the cost-minimisation objective. Data read uses SQLite and the intended solver backend is HiGHS, with the aim of a fully open-source pipeline from data ingestion through model generation to solver. The prototype remains in development and is available for translate to IEA-ETSAP Institutional open git repository.

Prototype performance

Benchmarking is intended to separate the model generation into three stages – data load, model build (LP matrix generation), and solve. The figures below are preliminary and indicative, gathered on a small-scale “utopia” TIMES model during development, rather than from a finalised, validated benchmark run. They also predate the most recent refactoring of the prototype (global variables removed, typed Dictionaries and DataFrame-based index joins adopted, profiler added, through October 2025), so the model-build comparison in particular should be re-measured on the current code. As of late November 2025 the prototype runs in HiGHS, but the chosen demo returns an infeasible solution – an outstanding model-formulation or input-data issue to resolve before the solve can be validated against GAMS, rather than a failure of the solver itself. Separately, benchmarking the new HiGHS interior-point (barrier) solver for representative timing results was still in progress. The solve-stage comparison is therefore not yet established. However, solver times are well documented in benchmark studies. The model build time is less well known, and so a similar order of magnitude build time, with considerably improveable code, is deemed a positive result for closing the gap between JuMP and GAMS on model build time.

Pipeline Stage	GAMS / Reference	TIMES.jl (preliminary)	Status & Driver
Data load	Baseline	Competitive	SQLite ingestion efficient; no material time gap observed
Model build (matrix generation)	Baseline	~2X – 10X slower than GAMS	Prototype since refactored: globals

Pipeline Stage	GAMS / Reference	TIMES.jl (preliminary)	Status & Driver
		(pre-refactor estimate; re-benchmark pending). Additional methods for improvement are known.	removed, typed Dictionaries and DataFrame joins adopted. Residual lever: abstractly-typed tuple containers.
Solve (HiGHS vs CPLEX)	CPLEX baseline	Comparable but model scale dependent. Note: Once a model is passed to the solver, runtime speed is largely agnostic of original model language.	Chosen demo returns infeasible (model/data issue, not solver failure); interior-point benchmarking in progress

Main findings

- Expressibility is established: the core TIMES LP formulation (commodity balance, activity-flow, capacity-activity, capacity transfer, objective) has been successfully expressed in Julia/JuMP within the prototype.
- Full validation is outstanding: a clean, feasible solve on a reference model and end-to-end numerical agreement with GAMS have not yet been demonstrated. The prototype runs in HiGHS, but the chosen demo currently returns an infeasible solution — a model-formulation or input-data issue to resolve, not a solver failure. Performance benchmarking of the new HiGHS interior-point solver was also still in progress.
- The performance signal is in model generation, not solving: preliminary benchmarking points to a model-generation gap attributable to data-structure and algorithm choices rather than any intrinsic limitation of Julia or JuMP.
- The gap is known to be closeable: idiomatic, type-stable Julia using DataFrames.jl, sparse containers, and single-pass constraint assembly should bring model-build time toward parity with GAMS, based on analogous benchmarks and other's experiences (SINTEF, TNO, NRL).
- A fully open-source pipeline (Julia-JuMP + HiGHS) is the design target and is plausible for small and medium TIMES models, but this remains to be confirmed by a validated end-to-end run.

2.1.1. Large Model Feasibility – Can JuMP transform large scale optimisation models in solver ready formats at a comparable speed to GAMS

The evidence indicates yes, conditional on idiomatic implementation. JuMP is used to build very large optimization models in production energy tools including GenX (MIT/Princeton), Sienna (NRL/NREL), SpineOpt, and PowerModels.jl. SINTEF reports that model construction time has not been a limiting factor for their large Julia/JuMP models, except where sparsity was handled naively – precisely the issue addressed by SparseVariables.jl. For TIMES specifically, large-model feasibility rests on: (i) re-engineering the pre-processing pipeline for type stability and sparse iteration; (ii) adopting an efficient data layer (SQLite is already in use in the prototype; DuckDB or Apache Arrow are potential candidates for very large datasets); and (iii) replicating the TIMES reduction logic. Julia’s in-memory model modification and re-solve capability is an additional advantage for parametric scenarios and rolling-horizon workflows.

2.2. Solver Performance – Can open source free solvers be used as an alternative to commercial solvers like CPLEX, Gurobi and XPRESS?

Partially, and increasingly so. The assessment is TIMES model-instance-dependent:

- HiGHS, the leading open-source LP/MIP solver, performs competitively on small and medium TIMES instances. It solved TIMES-NZ and some TIMES-Ireland scenarios successfully where temporal and regional complexity was moderate to low.
- Production-scale TIMES models with full temporal resolution, multi-regional trade and millions of non-zeros currently exceed HiGHS’s capability, and commercial solvers (CPLEX, Gurobi, Mosek, Xpress or similar) remain necessary for these. HiGHS interior point method (IPM) solver HiPO is in focused development and improving every few months.
- Solver complexity scales with regional trade and the number of commodities, not just regional and temporal resolution; HiGHS solution viability therefore depends on the specific model configuration, how well the model is scaled and is currently competitive typically below $1e^6$ variables.
- Crucially, the choice of solver is independent of the modelling language. JuMP connects to CPLEX, Gurobi, Xpress, Mosek and HiGHS (+50 solvers in total) through the same MathOptInterface layer, so migration does not lock users out

of commercial solvers — it removes only the GAMS licence layer.

Open-source solver capability is improving rapidly. The Open Energy Transition (OET) benchmark and the DLR PIPS-IPM++ work (Section 4.6–4.7) are the most relevant ongoing references for tracking this trajectory.

2.3. Solver Pricing – Purchasing power. Possibilities to acquire commercial solvers at lower costs should be investigated

Several avenues to reduce commercial solver costs were identified during the study:

- Distribution rights: ETSAP could invest in distribution rights to offer CPLEX licences to the community at reduced cost. A figure of approximately €204,000 was indicated by IBM as the investment required to secure such rights for 80 [ETSAP] seats — a significant but potentially community-wide saving.
- Academic programmes: CPLEX is available free to academics via the IBM Academic Initiative, and Gurobi offers free academic licences. These cover a substantial share of the ETSAP academic user base at zero solver cost.
- Vendor dynamics: Gurobi’s shift to subscription pricing has increased commercial costs, while CPLEX development has stagnated under IBM (Section 6). These dynamics strengthen the case for an open-source-capable full stack toolset that preserves commercial-solver optionality while protecting ETSAP modellers from pricing volatility.
- Another option is to make available a centralised HPC system for ETSAP members and ETSAP future members to solve in the cloud. ETSAP could maintain a single advanced commercially licensed server and queue TIMES jobs based on a few simple criteria. This could also considerably reduce the collective commercial solver licence costs across the ETSAP–TIMES Community if users were willing to share hardware and licences.

3. Migration Risks

This section assesses the risks of implementing TIMES in Julia/JuMP relative to GAMS, structured around the dimensions specified in the ToR.

3.1. Implementation of TIMES in Julia JuMP vs GAMS

The foundational risk is that migration is a re-engineering exercise rather than a

simple translation exercise. The TIMES GAMS code, refined and compacted over two decades, embeds implicit behaviours and domain assumptions. Re-expressing this in Julia requires deconstructing why the GAMS code is structured as it is, not merely what it superficially does. Misinterpreting this logic could yield a model that runs but produces subtly incorrect results — the principal “lost in translation” risk. The mitigation is twofold: (i) an automated, deterministic, well-tested GAMS-to-JuMP converter (Section 6.4) to reduce manual error; and (ii) an automated GAMS-to-JuMP data driven “Validator” to provide comprehensive numerical validation against GAMS outputs at every code functional improvement milestone.

3.1.1. Open Source Community Support Vs Commercial Proprietary Support

A change of support model accompanies the change of language. Under GAMS, support is commercial and centralised: a vendor with contractual obligations, mature documentation, and decades of solver-link refinement. Under Julia/JuMP, support is community-based and distributed: active Discourse forums, GitHub, and a responsive core developer team, but without a contract to the end user. The evidence from the study is reassuring on this point. The JuMP team has maintained a stable, backward-compatible API since v1.0 (2022) with no planned breaking changes, has rewritten all solver interfaces, and offers code review and performance advice to energy-modelling projects (e.g., GenX (MIT/Princeton)). SINTEF (Leading Norwegian scientific research lab) described the JuMP community as inclusive and supportive, and reported that open-source development improved their code quality through better automated testing. The residual risk — that community priorities may not always align with TIMES-specific needs — is mitigated by ETSAP’s ability to fund targeted development and by committed advisory relationships with the JuMP core team.

Open source communities can be paid to maintain stability under open source licenses. This is typically done to maintain a free price to end users, while the core cost is borne by institutional research funding as opposed to a commercial price per licence model.

3.2. Backward Compatibility

Two senses of backward compatibility apply. First, compatibility of the Julia/JuMP platform with itself over time: this is strong. Julia has had no breaking changes since v1.0 (2018) and is now at v1.12+; JuMP has been stable since v1.0 (2022). Code

written years ago still runs. This directly answers the version–instability concern that featured prominently in the 2017 ETSAP feasibility assessment. Second, compatibility of a Julia–based TIMES code with existing TIMES models and datasets: this is the harder problem, addressed in Section 5.2, and depends on the data input layer faithfully consuming existing VEDA/xl2times model definitions and on numerical equivalence of results.

3.3. Ability to address logical constructions implemented in TIMES

TIMES makes extensive use of GAMS’s conditional (\$) operator, set–membership logic, and implicit looping to express which variables and constraints exist for which index combinations. These logical constructions are fully expressible in Julia — indeed Julia’s general–purpose nature affords more explicit control — but they must be re–expressed rather than transliterated. Conditional summations become filtered comprehensions or pre–built sparse supports; dollar–conditions become boolean masks applied during index construction. The risk is not capability but correctness and performance: naive translation of conditional logic is both slow and error–prone. This reinforces the case for an automated converter and validator with a comprehensive testing process.

3.4. Address the full range of possible TIMES model functionality.

A recurring community concern is that an initial Julia implementation might support only a reduced “TIMES–light” feature set, leaving advanced constructs (e.g., lumpy investments, advanced time–slicing, trade, vintaging, user constraints, climate module, stochastics, monte carlo analysis etc.) unsupported. Expert advice (notably from Gary Goldstein, drawing on the MARKAL–to–TIMES migration experience) is explicit: if the ultimate goal is a full replacement, the architecture must be designed from the outset to accommodate the advanced constructs, even if they are implemented later. Building a TIMES–light prototype that cannot scale to full functionality would risk an expensive re–architecture. The proposed workplan (Section 7) therefore front–loads architectural decisions that preserve the path to full feature parity and future innovation.

3.5. Post processing of solver outputs

Beyond model generation and solving, TIMES includes substantial post–processing that transforms raw solver outputs into the structured results consumed by VEDA2

and analysts (flows, capacities, marginal/dual values, emissions, costs). Migrating this output-processing layer is a parallel work stream to the model-generator migration. The Julia ecosystem is well suited to it: DataFrames.jl, Arrow.jl and the broader data-science stack make extraction, aggregation and export straightforward. The risk is primarily one of scope and effort rather than feasibility, and is reflected in the Phase 2 integration work.

3.6. TIMES Code maintenance risks under Julia–JuMP

Maintenance risk has three components. First, succession: the current GAMS code maintainer is approaching retirement, which is itself a risk to the TIMES in GAMS status quo and a motivation for modernisation; the maintainer has stated that any migration must proceed via an automated rewrite to be sustainable and mitigate the significant risk for manual human error. Second, expertise: a Julia-based TIMES requires maintainers fluent in both TIMES domain logic and idiomatic Julia, a presently scarce combination. The study's assessment is that programming-efficiency review (which does not require TIMES expertise) and mathematical-correctness review (which does) can be separated, broadening the pool of qualified contributors. Third, governance: an open-source codebase requires clear contribution guidelines, quality gatekeeping, and automated testing to prevent divergent, unvalidated forks. Establishing a Git-based continual integration (CI) test suite — explicitly recommended by the project team — is the single most important maintenance-risk mitigation, as it both protects correctness and lowers the barrier to community contribution.

3.7. Strategic Risk: Julia vs. Python Developer Ecosystem

A strategic risk raised in community dialogues is the choice of Julia versus a Python-based alternative, such as Linopy (or Pyomo).

Python is the world's most popular programming language and has a vastly larger developer community, mature libraries which would make it easier to hire programmers. Julia, by contrast, is a niche language with a developer base of a similar size to GAMS.

The Pros and Cons of Python & Linopy Vs Julia & JuMP as we understand them are summarised below.

Dimension	Python (& linopy)	Julia (& JuMP)
Developer Talent Pool	Pros: • World's largest programming-language community. • Much easier to hire general developers, interns, and students. • Popular for data science, ML, AI → attractive for newcomers. Cons: • General Python talent ≠ specialized optimization expertise. • Fewer developers with deep experience in high-performance modeling compared to Julia/JuMP community.	Pros: • Very strong specialized talent pool for mathematical optimization. • JuMP community is concentrated, experienced, and purpose-focused. Cons: • Julia is niche; the overall developer base is small (similar to GAMS). • Risk: small pool may be fully hired elsewhere; staffing challenges. • Less attractive to students/interns; fewer people learning it at universities even compared to GAMS.
Ecosystem & Libraries	Pros: • Rich ecosystem: pandas/polars, NumPy, SciPy, Plotly, Dash, Flask, Jupyter, etc. • Easy integration with other tools, workflows, and data pipelines. • Rapid development of dashboards, visualizations, and apps (e.g., PyPSA-Explorer). • Python widely used across fields → more reusable knowledge. Cons:	Pros: • JuMP offers a clean, mathematically expressive modeling layer many optimization practitioners prefer. • Specialized packages like SDDP.jl provide advanced capabilities (multi-stage stochastic optimization) that are difficult to replicate in Python today. Cons: • Smaller ecosystem for data processing, visualization, and

app building.

- Fewer high-level libraries for tasks surrounding modeling (I/O, dashboards, integration).

Performance & Modeling Capabilities

Pros:

- Linopy improving rapidly; supports warm starts and is exploring incremental/direct solver interfaces (via PyOptInterface or PyOptFrame).

Cons:

- No incremental model solving yet (useful for rolling-horizon or iterative formulations).

Pros:

- JuMP widely regarded as **state-of-the-art** open-source optimization modeling layer.

- Strong support for incremental solving and direct solver interactions.

- Performance benefits from Julia's JIT compilation for numerical workloads.

- More advanced extensions (e.g., for decomposition [SDDP.jl](#)).

Cons:

- Evidence of JuMP's performance superiority vs linopy often anecdotal; more benchmarking may be needed.

Interoperability with GAMS

Pros:

- GAMS-Python integration has improved; it works and enables mixed workflows during migration.

Pros:

- GAMS are developing [GAMS.jl](#) to integrate GAMS workflows with JuMP.

Community Risk & Long-Term Support	Pros: • Very large global community → much lower single-point-of-failure risk. • Easier to replace maintainers or contributors. • Students and interns more readily available for supporting tasks (I/O modules, visualization, tool integration). Cons: • Specialized optimization expertise may be harder to find despite the large general pool.	Pros: • Strong, dedicated optimization-focused community around JuMP. Cons: • Small developer base → harder to guarantee long-term maintainability. • Risk of key JuMP developers being unavailable in the future. • Harder to ramp up in-community capacity compared to Python.
Learning Curve & Accessibility	Pros: • Easiest mainstream language; very forgiving for beginners. • Students already familiar with it → lowers onboarding friction. Cons: • For complex optimization, Python tools may feel less “mathematical” or expressive than JuMP.	Pros: • Many optimization experts report Julia/JuMP’s mathematical structure “suits their way of thinking” for modeling. Cons: • Steeper learning curve for newcomers. • Much smaller educational footprint; rarely taught outside specialist groups. • Less appealing to the majority of students/interns (especially those seeking broader career applicability).
Ecosystem Maturity & Adoption	Pros: • Extremely mature, widely adopted in industry and academia. • Robust tooling for	Cons: • Julia ecosystem more limited and not seen as general-purpose by most developers.

development, deployment, and integration.

- Adoption outside scientific computing and optimization remains niche.

4. Cost

This section assesses costs across two TIMES user archetypes — academic and non-academic — covering commercial solver costs, VEDA costs, the initial migration development cost, and the cost implications for end users, including potential savings from free and open-source solvers. The GAMS licence pricing below, provided directly by GAMS (2025 & 2026), anchors the analysis. Where other authoritative figures were not available within the study.

GAMS licence pricing (provided by GAMS, 2025 & 2026)

GAMS offers two purchasing models: a paid-up (perpetual) licence with a high first-year cost and reduced annual maintenance thereafter, and an annual subscription with a level recurring fee. The tables below show the five-year cost profile for a single named user and for five named users for GAMS/BASE + CPLEX (all figures USD).

Single named user (USD):

Licence type	Year 1	Year 2	Year 3	Year 4	Year 5	5-yr total	Avg / user / yr
Paid-up licence	14,000	2,800	2,800	2,800	2,800	25,200	5,040
Annual subscription	7,000	7,000	7,000	7,000	7,000	35,000	7,000

Five named users (USD):

Licence type	Year 1	Year 2	Year 3	Year 4	Year 5	5-yr total	Avg / user / yr
Paid-up licence	44,800	8,960	8,960	8,960	8,960	80,640	3,226
Annual subscription	22,400	22,400	22,400	22,400	22,400	112,000	4,480

Module and solver-link licence prices (USD), as provided by GAMS, for one and five users:

GAMS module / solver link	1 user	5 users
GAMS/BASE	1,750	5,600
CPLEX	5,250	16,800
Gurobi	31,500	157,500
Gurobi (single-use)	17,200	86,000

Two observations follow directly from these figures. First, on the base system, the paid-up model is markedly cheaper over five years than the annual subscription once first-year costs are amortised: for five users, USD 80,640 versus USD 112,000 (a per-user-per-year average of USD 3,226 versus USD 4,480). Second, on solver links, Gurobi is far more expensive than CPLEX within the GAMS ecosystem – roughly six times the cost for a single user (USD 31,500 versus USD 5,250) and almost ten times for five users (USD 157,500 versus USD 16,800). This price differential is directly relevant to the solver discussion in Sections 2.2, 2.3 and 6, where Gurobi is noted to outperform CPLEX on many ESOM instances: the performance advantage carries a substantial licence premium.

Generally, TIMES users with existing GAMS licenses should not incur any additional solver related costs. This is due to the availability of the [GAMS.jl](#) package developed by GAMS which allows solving JuMP models using GAMS (and the included solvers). For any new users, it is important to distinguish between academic and non-academic use cases.

4.1. Academic use Case

Academic users benefit from substantial discounts on both GAMS and commercial solvers, and from free academic solver programmes. The cost burden of the current GAMS-based stack is therefore lower for this group than for non-academic/commercial users, which moderates (but does not eliminate) the financial case for migration. The GAMS pricing tables above provide the reference point; where these represent commercial rates, academic discounts apply on top (to be confirmed with GAMS). The principal academic motivations for migration are open-source¹ mandates, transparency and reproducibility rather than cost alone.

Academic users can benefit from free Gurobi [academic licenses](#), essentially making

¹ <https://digital-strategy.ec.europa.eu/en/policies/open-source-strategy>

the solver-related costs equal zero.

Additionally, GAMS^{Py} and a range of commercial solvers (CONOPT, COPT, CPLEX, MOSEK, XPRESS etc.) are available to academic users for free.

4.1.1. Assess commercial solvers costs for existing GAMS CPLEX users

Academic users of GAMS+CPLEX typically access CPLEX free of charge through the IBM Academic Initiative, or through GAMS's academic licensing. Migration to Julia/JuMP does not change this: CPLEX remains accessible to academics via JuMP's MathOptInterface on the same free-academic terms. The net change for an academic user is the removal of the GAMS academic licence requirement. One could connect TIMES.jl to a pre-existing GAMS perpetual license setup using GAMS.jl² which connects JuMP to GAMS using JuMP's MathOptInterface (MOI).

4.1.2. Assess Initial commercial solvers costs

For academics relying on free academic solver programmes, the initial commercial solver cost is effectively zero. For those requiring commercial-grade licences outside academic programmes (e.g., for funded contract research), initial costs apply per vendor terms.

4.1.3. Assess annual maintenance costs for commercial solvers

Annual maintenance/renewal under academic programmes is generally free or nominal, subject to annual re-registration. This is unchanged by migration.

4.1.4. Assess costs related to academic use of use of VEDA2

VEDA2 is a commercial user interface (UI) product maintained by Kanors-EMR, strongly linked with the IEA-ETSAP community, and is out of scope for this language migration. Academic VEDA licensing terms apply independently of the modelling-language choice. A Julia-based TIMES could continue to interface with VEDA, which would write the data requirements for the Julia code analogously to its current GAMS interface (confirmed by Amit Kanudia).

One can continue to use your existing GAMS perpetual licences by bridging Julia

² <https://github.com/GAMS-dev/gams.jl>

JuMP to GAMS, via [GAMS.jl](https://github.com/GAMS-dev/gams.jl).³

4.2. Non-Academic. Others* Commercial/Public Body/Government Use Case costs

Non-academic users bear the full cost of both GAMS and commercial solver licences, and do not qualify for free academic solver programmes. This group faces the steepest cost escalation and therefore has the strongest financial incentive to migrate — although it is also the group most reliant on large, complex models that currently require commercial solvers, and this group often provide support to new TIMES community members to get started or develop policy relevant TIMES models.

4.2.1. Assess commercial solvers costs for existing GAMS CPLEX users

Commercial and public-body users pay full rates for both the GAMS base system and the solver links. According to the figures above, a five-user paid-up GAMS-BASE/CPLEX licence costs USD 44,800 in year one and USD 80,640 over five years. Migration to Julia/JuMP removes the GAMS base-system and GAMS solver-link costs entirely, while preserving access to the same commercial solvers (CPLEX, Gurobi, Xpress) purchased directly, via JuMP's MathOptInterface .

4.2.2. Assess Initial commercial solvers costs

Within the GAMS ecosystem, the first-year solver-link costs are given above: CPLEX at USD 5,250 (one user) / USD 16,800 (five users), and Gurobi at USD 31,500 / USD 157,500. Standalone direct-purchase solver licences (for use with JuMP) are vendor-specific and differ from the GAMS-bundled prices. These solver costs are independent of the modelling language and persist under migration for users who require commercial-grade performance — although migration allows them to be procured directly rather than through GAMS.

4.2.3. Assess annual maintenance costs for commercial solvers

Under the GAMS paid-up model, annual maintenance from year two is approximately 20% of the year-one cost — USD 2,800 per year for a single user and USD 8,960 per year for five users on the base system. Under the annual-subscription model the full fee recurs each year (USD 7,000 single / USD

³ <https://github.com/GAMS-dev/gams.jl>

22,400 for five users). Commercial solver maintenance is additional and, in Gurobi's case, has moved to a subscription model that can be relatively expensive over time. These recurring costs are a significant component of total cost of ownership; the GAMS base-system maintenance is eliminated by migration, while solver maintenance is unaffected by the modelling-language choice.

4.2.4. Assess costs related to academic use of use of VEDA2

Non-academic VEDA2 licensing applies at commercial rates and is independent of the modelling language. Combined with annual commercial solver costs, VEDA represents a recurring cost that migration does not alter. The net migration saving for this group is the elimination of the GAMS licence. MIRO and XL2TIMES are alternative open source TIMES user interface options, but currently have less functionality than VEDA2.

4.3. Initial Migration Development cost to Julia JuMP

The one-off cost of developing a production-grade Julia/JuMP TIMES is estimated at approximately €1.1–1.3 million over 18 months, summarised below and detailed in Section 7.3.

Phase	Duration	Key activities	Estimated cost
Phase 1: Foundation	Months 1–4	Performance-ready prototype; core LP equations; demonstration-model validation (incl. feasible solve); CI test suite. Develop a Validator set of TIMES in GAMS model inputs and outputs for comparison with TIMES in JuMP results as the equation set increases.	€200k–250k
Phase 2: Integration & validation	Months 5–9	VEDA/xl2times interface; production-model validation; report writer; docs	€250k–300k
Phase 3: Core equation	Months 10–14	Full equation set; complete pre-processing pipeline;	€300k–350k

Phase	Duration	Key activities	Estimated cost
library		multi-region	
Phase 4: Community release	Months 15–18	Partner beta; training; handover	€150k–200k
Contingency (15%)	—	—	€135k–165k
Total	18 months	—	€1.035M–1.265 M

4.4. Initial Migration cost to Julia JuMP for TIMES Users.

Distinct from the development cost borne by ETSAP, individual users face their own transition costs — principally retraining and workflow adaptation, rather than software licensing. Integration with VEDA and XL2TIMES mitigations this retraining cost.

4.4.1. Licence Costs?

For end users, the licence cost of the Julia/JuMP platform itself is zero: Julia, JuMP and HiGHS are MIT-licensed and free in perpetuity. Users who require commercial solvers for large models continue to licence those separately, on the same terms as today, but the GAMS/Base licence is no longer required. For users whose models fit within HiGHS's capability, and use xl2times to run their model, the entire TIMES tool stack becomes free of licence cost. The non-licence transition cost is the learning curve: time invested by existing users to become proficient in the new workflow.

4.4.2. Point to Summary of work plan from below.

The user-facing transition (training materials, documentation, a programmer's manual for adding equations and variables, and partner beta testing) is delivered in Phases 3 and 4 of the development programme. See Section 7 for the full workplan, timeline and budget.

4.5. Cost Savings to some TIMES users using Free Open

Solvers

For the subset of users whose models can be solved with open source solvers such as HiGHS, migration delivers a complete elimination of GAMS platform and solver licence costs — removing both the GAMS base licence and the commercial solver-link cost. This is transformative for academic users in budget-constrained settings, for commercial-licensed institutions in the Global South, and for teaching and onboarding. It also lowers the barrier to reproducibility, since results can be regenerated by anyone without purchasing proprietary expensive software. The magnitude of saving is model-dependent: it is greatest for small and medium models and diminishes for the largest models that still require a commercial solver (which, post-migration, can be licensed directly rather than through GAMS).

4.6. Assess limitations of free solvers

“Free” solvers here include both free-of-charge academic licences of commercial solvers and free-tier offerings. Their principal limitations are conditional access (academic-only, or size-limited) and the absence of commercial support. Free academic CPLEX/Gurobi licences match commercial performance but are restricted to qualifying users and uses; free tiers (e.g., NEOS, VEDA-Online) cover only modest model sizes. The two studies below provide empirical grounding for the practical limits.

4.6.1. OET Benchmark Study

The Open Energy Transition (OET) open energy benchmark provides a public, ongoing comparison of solver performance across a suite of 200 LP and MIP problems from a range of energy-system models, including 17 instances from seven TIMES models.⁴ It is the most directly relevant reference for understanding where free and open-source solvers succeed and where they fall short on realistic TIMES-class problems, and should be cited to substantiate claims about solver viability.

4.6.2. DLR PIPS-IPM++ Study

PIPS-IPM++ is an open-source, parallel (distributed-memory) interior-point (IPM) solver developed in the context of the DLR-led work on large-scale energy system optimisation (BEAM-ME and successor project activities). It demonstrates that open and free solvers, when combined with problem-structure exploitation and

⁴ <https://openenergybenchmark.org/>

high-performance computing, can address very large energy-system LPs that exceed the reach of conventional single-node open-source solvers. It is an important counterpoint to the view that free solvers cannot scale, and points to a credible long-term pathway for fully open large-scale TIMES solving.⁵

4.6.3. NVIDIA's CuOpt as an emerging large model solver

A secondary counterpoint to the lack of open source solvers that solve large scale LP models is the emergence of NVIDIA's CuOpt, which is a GPU solver for very large scale LPs, starting at 1 million variables and upwards. CuOpt has a julia wrapper. CuOpt also has strong links to JuMP, as JuMP founder Miles Lubin (MIT PhD) is now a senior developer at NVIDIA working on CuOpt.

4.7. Assess limitations of open source solvers

"Open-source" solvers (HiGHS, SCIP, GLPK, PIPS-IPM++) are free of charge and fully inspectable, but face two main limitations for production TIMES use: (i) performance and robustness on the largest, most numerically challenging instances still lag the best commercial solvers (CPLEX, Gurobi); and (ii) commercial support and warranty are absent. HiGHS is the strongest general-purpose option and is closing the gap, but for the largest multi-regional, high-temporal-resolution TIMES models, commercial solvers currently remain necessary. The same two studies quantify these limits.

4.7.1. OET Benchmark Study

The OET benchmark again provides the empirical basis: it indicates that open-source solvers are presently not a complete substitute for commercial solvers on the largest TIMES models, while being entirely adequate for many small and medium instances (those with up to 1 million variables). The trajectory of improvement is favourable and should be tracked over the migration period.

4.7.2. DLR PIPS-IPM++ Study

PIPS-IPM++ illustrates both the limitation and its remedy: out-of-the-box open-source solvers struggle at extreme scale, but structure-exploiting,

⁵Zittel, Janina, et al. "Computational acceleration strategies for large-scale energy system optimization: a comparative study of GPU-accelerated and distributed-memory solvers." *arXiv preprint arXiv:2605.04605* (2026).

Hadidi, Lars, et al. "Large-Scale Linear Energy System Optimization: A Systematic Review on Parallelization Strategies via Decomposition." *arXiv preprint arXiv:2507.21932* (2025).

HPC-parallel open-source approaches can extend the reachable problem size substantially. For TIMES, this suggests that the long-term open-source solving strategy may involve decomposition and parallelisation rather than reliance on a single sequential solver.⁶

5. IEA-ETSAP Virtual Community Round Table Dialogues

Community acceptance for a migration from GAMS to JuMP was assessed through four structured IEA-ETSAP TIMES2JuMP Round Table Dialogues (July 2025) and a series of bilateral consultations spanning advanced, intermediate and early-career TIMES users, as well as JuMP core developers and external Julia-migration practitioners. Participating institutions included PSI (Switzerland), IFE (Norway), Luleå University of Technology (Sweden), University of Stuttgart (Germany), SINTEF (Norway), Kanors-EMR (India), the University of Auckland / JuMP, UCLouvain / JuMP, and ESMA Ltd (Ireland). The participant list is in the Appendix.

The overall finding is that the community is “agnostic to positive” about migration, with sentiment warming over the course of the consultations. Recurring themes were:

- Cost is the primary driver, not ideology. The consensus, articulated by Gary Goldstein, is that the discussion is fundamentally economic. Rising GAMS prices are the dominant motivation for senior users.
- Open-source mandates are real and growing. Government policy-advisory work in Sweden, Germany, Finland, UK, EU and Ireland increasingly requires open-source tools; some university faculties may soon require open-source models for PhD theses (University of Stuttgart).
- Performance sensitivity. Users running many scenarios care about model-build time; a 20- vs 30-minute difference is material to their workflow. This reinforces the need to achieve, not merely approach, performance parity.
- A parallel-support window must be managed. Community tolerance for maintaining GAMS and Julia in parallel is roughly one to two years; a clear transition plan is essential to avoid fragmentation.
- New-generation enthusiasm. Early-career researchers and budget-constrained institutions were the most enthusiastic and represent the primary growth vector for the community.

⁶ Hadidi, Lars, et al. "Large-Scale Linear Energy System Optimization: A Systematic Review on Parallelization Strategies via Decomposition." *arXiv preprint arXiv:2507.21932* (2025).

- Complementary signal: GAMS-Py. The ExCo vote (Nov 2025) to fund GAMS-Py — a Python wrapper over the existing GAMS code — passed with the explicit intention of using it as a stepping stone toward a future migration, indicating institutional openness to change.

5.1. Backward compatibility with existing models

Community acceptance is contingent on a Julia-based TIMES consuming existing model excel definitions and reproducing existing results. Two enablers reduce this risk. First, the data layer: VEDA can write data requirements for Julia code in a manner closely analogous to its current GAMS interface (confirmed by Amit Kanudia); this is also true for xl2times. Second, validation: the trust gateway for the community is multiple demonstrated numerical agreement between Julia and GAMS outputs across a comprehensive suite of partner models, not merely the utopia or Demo12 reference. It was stressed that validating existing ETSAP partner models against the new framework is both essential and substantial, and that a reduced-feature initial version cannot validate the full-complexity models in current use. The workplan treats large-scale validation as a first-class, non-negotiable deliverable.

5.2. Acceptance of the TIMES users to migrate to a new framework

Willingness to migrate is conditional but real. The conditions repeatedly surfaced were: (i) assurance that the full range of TIMES functionality will ultimately be supported, not just a “TIMES-light” subset (Section 3.4); (ii) a managed, time-bounded transition that does not strand existing GAMS users; (iii) retention of skilled coders within the ETSAP community and availability of maintainers; and (iv) demonstrated reliability and reproducibility through validation. Some users also welcomed the possibility of parallel Python and Julia versions running side by side, reflecting genuine interest rather than resistance. Provided these conditions are met — and the workplan is designed to meet them — the consultations indicate the community is prepared to accept, and in significant part actively welcomes, migration to an open framework. As one senior participant put it, the migration is widely viewed as “inevitable” given GAMS pricing trends.

6. Additional emergent points not in the TOR

The study surfaced several material points beyond the explicit ToR dimensions that bear on the migration decision.

6.1. CPLEX is no longer being developed.

CPLEX, long the default commercial solver for TIMES, has seen its development stagnate under IBM ownership. IBM is not investing meaningfully in advancing CPLEX. All the original CPLEX staff are now with other companies, largely Gurobi. This has strategic implications: continued reliance on a static commercial solver is itself a risk, while the open-source solver ecosystem (HiGHS) and the actively developed commercial alternative (Gurobi) are advancing. A migration that maintains diversity of solver options for TIMES — as JuMP (MathOptInterface.jl) does — reduces exposure to this stagnation.

6.2. Costs of CPLEX

Within the GAMS ecosystem the CPLEX solver link is comparatively modest — USD 5,250 for one user and USD 16,800 for five (per the pricing tables in Section 4) — whereas the Gurobi link is far costlier at USD 31,500 and USD 157,500 respectively. The route to lower-cost community access to CPLEX (e.g. ETSAP securing distribution rights for ~€204,000, or a runtime/restart licence model) requires significant up-front investment. Given the stagnation noted in 6.1, committing further community resources to CPLEX-specific licensing arrangements warrants careful consideration against the alternative of investing in open-source solver capability — particularly as a migrated, solver-agnostic TIMES would let users procure whichever solver best fits their price/performance needs directly, outside the GAMS bundle.

6.3. Gurobi already faster than CPLEX for many benchmarked ESOM model instances

On many energy-system optimization model (ESOM) benchmark instances, Gurobi now outperforms CPLEX. Combined with CPLEX's stagnation, this suggests that the assumption of CPLEX as the default high-performance solver is outdated. A language-agnostic solver interface (JuMP) allows users to select the best-performing solver per problem without code changes at a price point they are comfortable with — an advantage of the migrated architecture. GAMS also allows

different solvers to be chosen internally at runtime.

6.4. Role of AI in migration ? Is Python going to be even more dominant in the future?

Two AI-related developments emerged late in the study that materially affect strategy. First, an automated GAMS-to-JuMP converter is now in active development by Professor Benoît Legat (UC Leuven / JuMP core developer), using AI to read the GAMS documentation and generate a deterministic, well-tested translator, working in tandem with James Foster's GAMS data reader. Previously estimated as a multi-year manual effort, supervised-automated conversion now appears feasible and is the maintainer-preferred path.

JuMPy — a Python interface to JuMP — is being developed by Benoit Legat. This could allow developers to call and run [TIMES.jl](#) from Python, if they so wished.

The broader strategic question raised in consultations is whether the rapid maturation of AI coding tools — which are currently more effective on Python than on Julia, owing to Python's larger training corpus — will entrench Python's dominance and erode Julia's relative advantage. The balanced view from the study is: Julia retains a genuine mathematical-programming and performance edge for advanced modelling, but Python has a larger, more mature community and stronger AI-tooling synergy. The emergence of JuMPy narrows this trade-off by offering Python accessibility atop JuMP performance, and is a route to reconcile the two. Most TIMES users interface with the model via Excel input files, and so those who work via excel, would be less aware or sensitive to this functionality. This question should be revisited at the start of full development, as the AI-tooling landscape is evolving quickly.

7. Migration workplan

This section sets out the proposed programme for full TIMES-to-Julia/JuMP development: the workpackages, an indicative timeline, and the budget. The programme is phased and milestone-gated, prioritising idiomatic performance-engineering from the outset, automated conversion tooling, continuous validation against GAMS, and early engagement with the VEDA and xl2times maintainers. Dr Olexandr Balyk offers the existing [TIMES.jl](#) prototype repository to be used and transferred to ETSAP, as an institutional repository.

7.1. Project Workpackages

WP	Workpackage	Key tasks	Lead role	Exit criteria
WP1	Foundation & architecture (M1–4)	Performance-engineered prototype; core LP equations; DataBase-DataFrames (DuckDB, SQLite...) data layer; feasible solve and CI test suite on a TIMES demonstration model (e.g. Demo12); architecture for full feature set	Julia architect	Feasible solve achieved; model-build $\leq 1.5 \times$ GAMS on the demonstration model; CI passing
WP2	Integration & validation (M5–9)	VEDA/xl2times interface; report writer (output post-processing); full validation suite; programmer's manual	GAMS expert + Julia architect	5 diverse partner models validated; VEDA round-trip
WP3	Core equation library (M10–14)	Full TIMES equation set; complete pre-processing pipeline; reduction-algorithm equivalent; multi-region	TIMES/GAMS expert + Julia perf. engineer	Numerical agreement on 3 partner models; $\leq 1.2 \times$ GAMS
WP4	Community release (M15–18)	Partner beta; training workshop & materials; JuMPy integration; ETSAP handover & governance	Project lead	Beta complete; formal handover to ETSAP maintenance
WPO	Coordination & advisory (M1–18)	Project management; JuMP/OpenMOD advisory; ExCo reporting; risk management	Project lead	Milestone reporting at each ExCo

7.2. Gantt Chart timeline of workpackages, deliverables, tasks and milestones.

The indicative 18-month schedule below shows workpackage spans (■) and the milestone month for each phase gate (◆).

WP \ Month	1–3	4–6	7–9	10–12	13–15	16–18
WPO Coordination	■	■	■	■	■	■◆
WP1 Foundation	■	■◆				
WP2 Integration & validation		■	■◆			
WP3 Equation library			■	■	■◆	

WP \ Month	1–3	4–6	7–9	10–12	13–15	16–18
WP4 Community release					■	■◆

A critical historical precedent within ETSAP provides a realistic warning: the **Markal-to-TIMES rewrite**. That project was originally budgeted for two to three years but **ultimately took five years** to complete

7.3. Migration Project Budget

The budget is built around four senior specialist roles plus coordination and advisory support over 18 months, with a 15% contingency. Indicative day-rate and FTE assumptions should be finalised at the tendering stage.

Budget line	Basis	Estimated cost (€)
TIMES/GAMS domain expert	1 FTE × 18 months	260,000–300,000
GAMS expert (pre-processing / converter)	1 FTE × 12–18 months	180,000–240,000
Julia/JuMP performance engineer	1 FTE × 18 months	240,000–280,000
Julia/JuMP software architect	1 FTE × 18 months	240,000–280,000
Coordination & project management	0.3 FTE × 18 months	60,000–80,000
Advisory (JuMP core team, & other senior GAMS→JuMP modellers)	Consultancy days	30,000–50,000
Software, AI (Claude) tokens/Subscriptions, infrastructure, travel, dissemination,	Direct costs	25,000–40,000
Subtotal	—	1,035,000–1,270,000
Contingency (15%)	—	155,000–190,000

Budget line	Basis	Estimated cost (€)
Indicative total	18 months	≈ €1.1–1.3 million

Appendix: Consultation Participants

Name	Institution	Role / Expertise	Date
Oscar Dowson	JuMP	JuMP lead developer; code review	Apr 2025
Gary Goldstein	DecisionWare / ETSAP	TIMES co-founder; ETSAP–GAMS liaison	Jul 2025
Anna Krook–Riekkola	Luleå University of Technology	Senior TIMES user; policy modelling	Jul 2025
Felix Lippkau	University of Stuttgart (IER)	PhD researcher; heavy TIMES/VEDA user	Jul 2025
Evangelos Panos	Paul Scherrer Institute	Advanced TIMES user; large-scale models	May 2025
Pernille Seljom	IFE (Norway)	Advanced TIMES user	May 2025
Amit Kanudia	Kanors–EMR (VEDA)	VEDA developer; modelling platform	Aug 2025
Lars Hellemo	SINTEF (Norway)	Julia/JuMP developer; SparseVariables.jl	Oct 2025
Truls Flatberg	SINTEF (Norway)	Julia/JuMP migration practitioner	Oct 2025
Benoît Legat	UCLouvain / JuMP	JuMP core developer; GAMS→JuMP converter; JuMPy	Apr 2026
Luke Searle	ECCA	TIMES NZ Modeller	June 2025
James Foster	CSIRO	TIMES Australia Modeller	July 2025
Antti Lehtila	VTT	TIMES Maintainer	July 2025
Kanna Ramachandran	PSI	Swiss TIMES modeller	July 2025
Sophie Chlela	Mine Paris Tech	TIMES–FR Modeller	July 2025
Maurizio Gargiulo	E4SMA	TIMES Trainer	July 2025
Will Catton	ECCA	TIMES–NZ	July 2025
David Paoella	Breakthrough Energy	RD&D on GRIDS	July 2025

Name	Institution	Role / Expertise	Date
Diego Tejadaarango	TNO	TulipaEnergy & SpineOpt	July 2025

Discussions on the OpenMod forum:

<https://forum.openmod.org/t/feasibility-of-migrating-times-to-julia-jump-from-gams/5326/6>

Discussion on LinkedIn:

https://www.linkedin.com/posts/we-have-been-tasked-within-the-etsap-community-share-7336451658794618880-QoWV/?utm_source=share&utm_medium=member_desktop&rcm=ACoAAAWCNKgBvZFauxK8-U1-9yQCOb_t7q5rLX0