Deploying OpenIPSL Models into DIgSILENT PowerFactory via FMI Model Exchange

Hao Chang[1] Luigi Vanfretti[1]

[1]Electrical, Computer and Systems Engineering Department, Rensselaer Polytechnic Institute, Troy, NY, USA,
{changh7, vanfrl}@rpi.edu

Abstract

Translating open-source Modelica power-system models into commercial grid-simulation environments remains a practical barrier to using research-grade libraries in utility workflows. The Open-Instance Power System Library (OpenIPSL) provides validated Modelica machine, excitation, and control models, but, to the authors' knowledge, no published study has documented an end-to-end deployment into DIgSILENT PowerFactory. This paper presents such a workflow using the Functional Mock-up Interface (FMI) in *Model Exchange* mode, in which PowerFactory's integrator advances the FMU's states directly. Two coupling granularities are defined: a *controller-only* pattern that exchanges scalar AVR signals between the FMU and PowerFactory's native synchronous machine, and a *full-machine* pattern that couples the FMU to the grid through a voltage/power boundary. Both are applied to a benchmark using the OpenIPSL GENROU and SCRX models and validated against PowerFactory's built-in PSS®E SynMach SCRX. Steady states match across both patterns; controller-only transients track the reference closely, while the full-machine pattern shows a higher field-voltage overshoot, indicating sensitivity to machine-level implementation details and FMU–grid boundary dynamics. The workflow, FMUs, and project files are provided as an open source repository.

Keywords: Modelica, OpenIPSL, Functional Mock-up Interface (FMI), Model Exchange, PowerFactory, power system dynamics, model validation, synchronous machine, excitation system.

1 Introduction

OpenIPSL (Baudette et al. 2018) has become the principal open-source Modelica library for phasor-domain power-system simulation, providing validated synchronous machines, excitation systems, governors, stabilizers, and inverter-based resources benchmarked against PSS®E. Utility engineers, however, run their studies in commercial tools—most prominently DIgSILENT PowerFactory—and an OpenIPSL model developed today cannot easily be exercised in such a tool tomorrow without re-implementation or cross-tool packaging.

FMI is the obvious packaging mechanism, but its application here has been only partially explored. PowerFactory added FMI model exchange in 2022 and native Modelica modeling in 2023–2024. Prior FMI–PowerFactory work targeted Simulink-authored FMUs for wind-plant studies (Meer et al. 2019), and recent OpenIPSL FMU work from the ALSETLab group targeted OPAL-RT's ePHASORSIM for grid-forming-inverter studies (Basumallik et al. 2025).

This paper fills that gap with four contributions: (i) a documented Model-Exchange workflow exporting OpenIPSL components from OpenModelica as FMUs and integrating them into PowerFactory's Composite Model Frame; (ii) two coupling patterns with explicit trade-offs—a *controller-only* pattern exchanging scalar AVR signals and a *full-machine* pattern coupling through a voltage/power boundary; (iii) validation against PowerFactory's native PSS®E SynMach SCRX under an excitation disturbance; and (iv) an open repository of FMUs, source, and project files available at `https://github.com/ALSETLab/Modelica-FMI-PowerFactory-ModelImport-Tutorial`

2 Background and Related Work

2.1 The FMI standard

FMI (Blochwitz et al. 2011) provides a tool-independent route for exchanging dynamic models, and is the foundation of the workflow presented here. PowerFactory's current support is built on FMI 2.0, with the Model Exchange and Co-Simulation interfaces exposed through distinct ingestion paths: Model-Exchange FMUs are registered as *Modelica Model Types* and participate in the host RMS solver directly, while Co-Simulation FMUs are coupled through a separate framework introduced in PowerFactory 2022. This paper targets the Model Exchange path. The boundary-filter design used in the full-machine pattern (Section 3) follows established guidance on coupling stiff, tightly-coupled systems (Bastian et al. 2011; Gomes et al. 2018).

2.2 OpenIPSL for power-system dynamics

OpenIPSL (Baudette et al. 2018), forked from the earlier iTesla Power Systems Library (Vanfretti, Rabuzin, et al. 2016), provides validated Modelica implementations of synchronous machines, excitation systems, governors, stabilizers, and inverter-based resources, all benchmarked

DOI 10.3384/ECP219 153

against PSS®E references. Recent extensions have added specialized North-American utility models (Islam et al. 2023) and demonstrated OpenIPSL as a digital-twin substrate for identification and control (Vanfretti and Roychowdhury 2024). Despite this maturity, OpenIPSL models lack a routine deployment pathway into the commercial tools used for utility grid studies—the gap this work addresses. OpenModelica (Fritzson et al. 2018), used throughout the workflow described here, supports FMU export and is compatible with OpenIPSL.

2.3 PowerFactory's Modelica and FMI interfaces

DIgSILENT PowerFactory has incrementally added support for the Modelica Association's standards over recent releases. Generic FMI-based model exchange between PowerFactory's RMS engine and external tools was introduced in PowerFactory 2022, followed by native Modelica modeling support for both continuous- and discrete-time models in PowerFactory 2023 and 2024. The integration mechanism centers on PowerFactory's Composite Model Frame, an established slot-based wiring abstraction originally designed for DSL (DIgSILENT Simulation Language) user-defined models. An imported FMU is registered as a *Modelica Model Type* with a "compiled model" flag and a path to the `.fmu` file; the resulting object is then bound to a slot in a Composite Frame, allowing it to participate in load-flow-initialized RMS simulations alongside the tool's native components. Variables declared as `External` in the FMU's local initialization are populated by PowerFactory's load-flow solution before the dynamic simulation begins, providing the initialization handshake between the steady-state grid solution and the FMU's internal state. While the vendor documentation describes this mechanism, the published literature characterizing how to exploit it for cross-tool model exchange—particularly with open-source Modelica libraries—remains thin, and this gap is precisely what the present paper addresses.

2.4 Prior FMI-based power systems simulation

Prior FMI–PowerFactory work has used Simulink as the FMU source. Van der Meer et al. coupled PowerFactory's RMS engine to Simulink-generated FMUs for wind-plant integration studies (Meer et al. 2019), demonstrating the viability of PowerFactory as the coordinating simulation environment but not addressing open-source Modelica libraries as the FMU source.Within the OpenIPSL ecosystem, Basumallik et al. deploy a grid-forming inverter FMU into large-scale New York State studies via OPAL-RT's ePHASORSIM (Basumallik et al. 2025); their work shares this paper's motivation but targets a different host tool and does not explore the coupling-granularity spectrum or validate against a native commercial reference. The OpenIPSL → PowerFactory pathway remains undocumented in the published literature.

3 Simulation Architecture

The central design decision when porting an OpenIPSL component into PowerFactory is the *coupling granularity*: which portion of the OpenIPSL model is packaged inside the FMU. At one extreme the FMU contains only a single control element—an AVR, a governor, a stabilizer—and exchanges scalar signals with surrounding native machine and grid models. At the other extreme the FMU contains a complete machine-and-controls subsystem, and the surrounding network sees it as a power injector at a terminal bus. The two extremes exercise the FMU at qualitatively different levels and impose qualitatively different demands on initialization and signal wiring. The FMU is exported as Model Exchange in both cases, since this is the interface PowerFactory's Modelica Model Type ingestion path accepts.

3.1 Pattern A: controller-only coupling

In Pattern A the FMU contains only the OpenIPSL SCRX excitation system; the synchronous machine and all grid elements remain native PowerFactory components. The FMU sits inside the machine's control loop: it consumes the compensated terminal voltage and returns the commanded field voltage.

Four scalar signals cross the boundary. Three are inputs: `ECOMP` (compensated terminal voltage), `ECOMP0` (its steady-state value), and `EFD0` (the steady-state field voltage). The output is `EFD`, the commanded field voltage. The two `...0` signals are required by the OpenIPSL SCRX block to initialize its integrator state consistently with the network, and both are direct outputs of PowerFactory's load-flow solution. The initialization handshake therefore reduces to declaring these inputs with `External` causality in the FMU's local-initialization section; PowerFactory populates them automatically.

Pattern A's interface surface is minimal, its compile time short, and its algebraic coupling loose. It is well suited to studies that prototype new control laws against a trusted machine and grid. The limitation is that the FMU's machine model is never exercised, so studies of new machine representations such as those in (Islam et al. 2023) cannot use this pattern.

3.2 Pattern B: full-machine coupling

Pattern B exports a complete machine-and-controls subsystem as the FMU: a GENROU round-rotor machine, the same SCRX excitation as Pattern A, and a boundary block with first-order filters. The surrounding grid is unchanged, but the GEN1 unit is replaced by an interface frame containing a VoltageSensor, the FMU, and a load slot. From the grid's perspective the FMU is a controlled power injector at the GEN1 bus.

Four scalar signals cross the boundary. The inputs are `vr` and `vi`, the terminal voltage components at the GEN1 bus. The outputs are `Pext` and `Qext`, driving a load slot that closes the loop. First-order filters on the out-

DOI 10.3384/ECP219 153

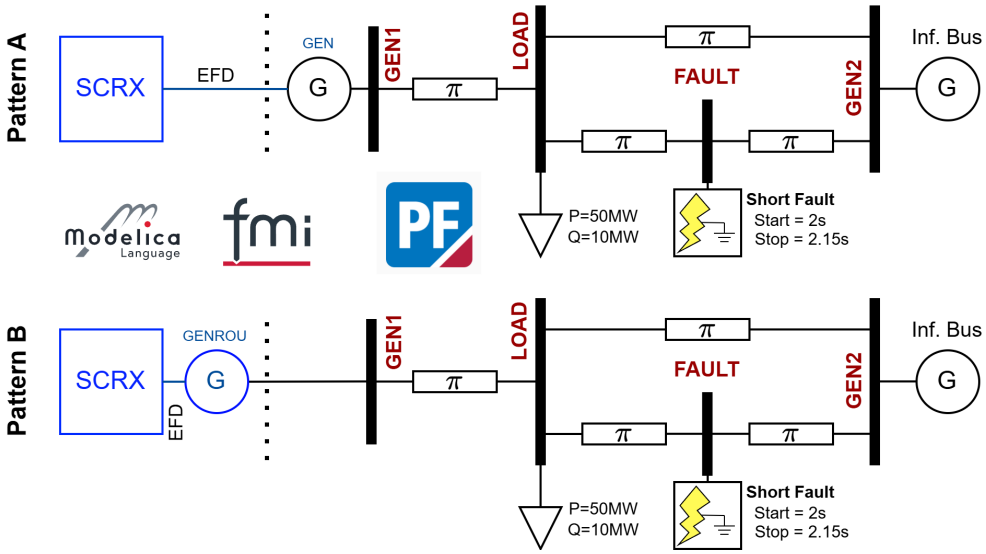


Figure 1. Two FMI Model-Exchange coupling patterns for deploying OpenIPSL models into PowerFactory

puts smooth the algebraic relationship between voltage and power, reducing the risk of algebraic loops at the FMI boundary—a precaution motivated by the coupling stability analysis in (Gomes et al. 2018).

Initialization is more involved than in Pattern A. The GENROU machine carries six dynamic states plus the SCRX internal states, none of which appear directly in PowerFactory's load-flow output. The handshake proceeds in two stages: the load flow solves for steady-state terminal voltage and pre-disturbance P/Q at GEN1; these are passed through `External`-classified inputs into the FMU, where the machine's initialization equations backsolve for the internal states. This is more sensitive to inconsistencies between PowerFactory and OpenIPSL parameter conventions (per-unit bases, saturation function forms, reactance definitions) than Pattern A, and small differences can manifest as transient discrepancies even when steady states agree.

The pattern exercises the full electromechanical dynamics of the OpenIPSL GENROU and is the appropriate choice for studies of new machine models or coupled machine–control interactions. The cost is larger FMU size and compile time, more demanding initialization, and tighter algebraic coupling that may motivate FMI 3.0's intermediate update or clocked mechanisms (Junghanns et al. 2021) in future work.

3.3 Trade-off summary

Table 1 summarizes the practical trade-offs between the two patterns across seven dimensions. The dimensions are deliberately chosen to reflect decisions a user faces *before* attempting the integration: the interface surface and FMU complexity affect how much of the OpenIPSL source must be packaged; initialization difficulty and algebraic-coupling risk affect how much debugging time the deployment will consume; communication overhead and computational cost affect whether the resulting FMU-based simulation is fast enough for the intended study; and debuggability affects whether discrepancies between the FMU response and a reference baseline can be localized to the FMU, to the grid, or to the boundary itself.

Neither pattern is uniformly preferable; the choice is study-driven. The benchmark case in Section 4 applies both patterns to the same disturbance on the same grid, using PowerFactory's native PSS®E SynMach SCRX as a common reference, in order to characterize where the two patterns agree, where they diverge, and what the divergences reveal about the underlying modeling assumptions on either side of the FMI boundary.

4 Case Study and Results

This section validates both coupling patterns against a common reference on a shared benchmark. The validation strategy is deliberately simple: the same grid, the same operating point, and the same disturbance are applied in three configurations—an all-native PowerFactory simulation using the built-in PSS®E SynMach SCRX (the reference), Pattern A with the OpenIPSL SCRX as a controller-only FMU, and Pattern B with the OpenIPSL GENROU+SCRX as a full-machine FMU—and the resulting time-domain responses are overlaid. Steady-state agreement is the necessary condition (an FMU-based simulation that does not reach the same equilibrium as the reference cannot be trusted for transient studies regardless of how its dynamics look).

4.1 Benchmark system

The benchmark system, shown in Figure 1, is a small two-machine network with an external grid equivalent. The machine under study is GEN1, connected to the GEN1 busbar; this is the unit whose excitation system (Pattern A) or whose entire machine-and-controls subsystem

DOI 10.3384/ECP219 153

Table 1. Trade-offs between the two coupling patterns.

Aspect	Pattern A (controller-only)	Pattern B (full-machine)
Interface surface	ECOMP, ECOMP0, EFD0, EFD	vr, vi, Pext, Qext
FMU contents	SCRX excitation only	GENROU + SCRX + boundary filters
Initialization	Load-flow values map to External inputs	Two-stage: load flow → External inputs → FMU back-solves internal states
Algebraic coupling	Loose (regulator dynamics dominate)	Tight (mitigated by 1st-order P/Q filters)
Step-size sensitivity	Low (single control block)	Moderate (full machine state per step)
Debuggability	High: AVR isolable from grid	Lower: machine and grid entangled
Recommended use case	Validating novel controls against a trusted machine and grid	Validating novel machine models or coupled machine–control studies

(Pattern B) is replaced by the FMU in the two coupled configurations. A second machine GEN2 represents a remote unit on the other side of a common impedance network and is held as a native PowerFactory synchronous machine in all three configurations. A lumped LOAD and a SHUNT element sit between the two machines, and the FAULT bus—where the disturbance is applied—is interconnected with the rest of the system through common impedances. An external grid equivalent at the boundary provides the system reference frame and absorbs the residual power balance.

The operating point for all three configurations is set by a single load-flow solution: GEN1 dispatches its rated active power with the AVR holding the GEN1 terminal voltage at 1.00 p.u., producing a pre-disturbance field voltage of approximately 1.434 p.u. at the SCRX output. This value is identical across the three configurations by construction. The matching pre-disturbance steady state is therefore not a coincidence but a sanity check on the handshake, and is verified explicitly at the start of each run by a brief no-disturbance smoke test before the actual study disturbance is applied.

4.2 Disturbance scenario

A single disturbance is applied at $t = 2$ s in all three configurations: a three-phase short-circuit fault at the FAULT bus. The fault is cleared at $t = 2.15$ s, giving a fault duration of 150 ms. The same disturbance is applied to the all-native PowerFactory reference case, the controller-only FMU configuration, and the full-machine FMU configuration.

The choice of a short-circuit fault is deliberate. A network fault simultaneously exercises the excitation system, the synchronous machine's electromechanical and flux dynamics, and the surrounding network's voltage and reactive-power response. This makes it a strong test of the FMI coupling patterns. In Pattern A, the disturbance propagates through the native PowerFactory machine and exposes the OpenIPSL SCRX FMU only through the regulator input–output interface. In Pattern B, the same disturbance also exercises the OpenIPSL GENROU machine inside the FMU, making the comparison sensitive to machine-state initialization, saturation behavior, per-unit conventions, and the voltage/power boundary dynamics.

The simulation is run from $t = -1$ s to $t = 10$ s, with the pre-disturbance window providing visual confirmation of steady-state agreement and the post-fault interval covering the electromechanical transient and return to a new equilibrium.

The recorded variables are the field voltage `EFD` at the GEN1 SCRX output and the terminal voltage magnitude at the GEN1 busbar. `EFD` is the most direct indicator of differences inside the excitation loop and is the more sensitive of the two signals to modeling assumptions on either side of the FMI boundary; terminal voltage is the most relevant grid-facing quantity and indicates how those internal differences propagate into the network response.

4.3 Pattern A results

Figure 2 overlays the Pattern A response (solid) against the native PSS®E SynMach SCRX reference (dashed). The top plot shows the field voltage `EFD`; the bottom shows the terminal voltage magnitude at GEN1.

Both traces originate at the same pre-disturbance steady state (`EFD` ≈ 1.434 p.u., $V_t = 1.00$ p.u.), confirming that the initialization handshake is functioning correctly. The post-disturbance trajectories then agree to within numerical tolerance: post-disturbance RMS errors are 0.039 p.u. on `EFD` and 0.003 p.u. on V_t, with the order-of-magnitude difference between the two reflecting the fact that the brief saturation peak at $t \approx 2.1$ s is the only locus of meaningful disagreement, and its effect is filtered out by the machine downstream of the regulator. Apart from that peak, the two responses are visually indistinguishable through the negative `EFD` excursion near $t = 3$ s, the damped oscillation at $t \approx 4$ s, and the return to a common steady state. The terminal voltage shows the same agreement, dipping to ≈ 0.6 p.u. and recovering with identical time constants.

The interpretation is straightforward. Because the FMU contains only the SCRX block and the downstream ma-

DOI 10.3384/ECP219 153

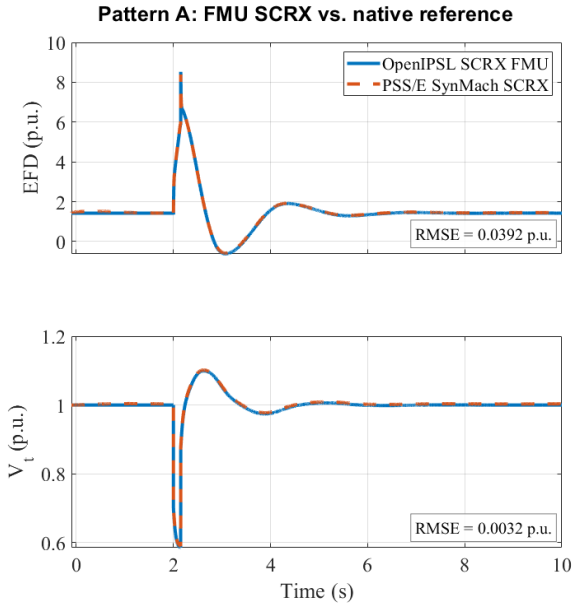


Figure 2. Controller-only coupling result

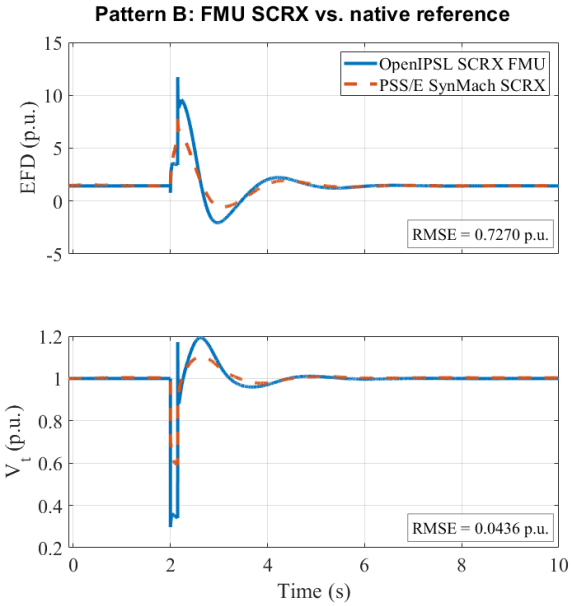


Figure 3. Full-machine coupling result

chine is the same native PowerFactory element in both configurations, the only potential source of disagreement is between the two SCRX implementations. The observed agreement therefore confirms that the OpenIPSL and native SCRX implementations match to within numerical tolerance for this disturbance — both target the same PSS®E block diagram, and the controller-only coupling exposes only their regulator input–output behavior across the FMI boundary. Pattern A behaves as a clean controller swap.

4.4 Pattern B results

Figure 3 shows the corresponding overlay for Pattern B, with the FMU now containing the full GENROU machine *and* the SCRX excitation system, coupled into the grid through the voltage/power boundary described in Section 3.2.

Pattern B's comparison tells a more nuanced story than Pattern A. Pre-disturbance, the two traces start at the same operating point (`EFD` ≈ 1.434 p.u., $V_t = 1.00$ p.u.), confirming that the two-stage initialization handshake has converged correctly. Post-disturbance, both traces also settle to the same steady state within approximately 4 s. The steady-state validation is therefore as clean as Pattern A's.

The transient response, however, diverges visibly. Post-disturbance RMS errors are 0.727 p.u. on `EFD` and 0.044 p.u. on V_t — roughly 19× and 14× larger than Pattern A's respective values. Two features dominate the disagreement. First, the FMU's `EFD` reaches a peak excursion of ≈ 11 p.u. at $t \approx 2.1$ s, while the PSS®E reference peaks at ≈ 6.5 p.u. The FMU's terminal voltage dips correspondingly deeper at the disturbance instant ($V_t \approx 0.3$ p.u. vs. ≈ 0.6 p.u.) and overshoots higher on recovery (≈ 1.18 p.u. vs. ≈ 1.10 p.u.). Second, a smaller but persistent offset between the two `EFD` traces remains through the post-transient recovery window ($t \approx 4$–6 s), with the FMU consistently slightly below the reference before the two converge near the steady state. The V_t traces are in close agreement through this same window, indicating that the residual `EFD` offset is again filtered by the machine downstream of the regulator.

Two observations summarize the divergence. The disagreement is concentrated in two windows: a narrow ≈ 1.5 s transient immediately following the disturbance, and a slower post-transient recovery from $t \approx 4$ s to $t \approx 6$ s. And the divergences across the two recorded variables are correlated in direction — higher `EFD` peak, deeper V_t undershoot, higher V_t overshoot, persistent post-transient `EFD` offset — which suggests a common underlying cause at the machine–grid boundary rather than independent disagreements between the excitation and machine subsystems.

Table 2. Initialization, simulation, and total wall-clock time across the three configurations.

Configuration	Init. time (s)	Sim. time (s)	Total (s)
Native PSS®E	0.007	0.127	0.133
Pattern A	0.010	0.123	0.133
Pattern B	0.015	0.322	0.337

4.5 Performance

Table 2 reports initialization, simulation, and total wall-clock times for the three configurations on a common workstation, with a fixed maximum integration step size of 10 ms and 11 s of simulated time ($t \in [-1, 10]$ s). Each value is the mean of three back-to-back runs; variation between runs was negligible.

Two observations stand out. First, Pattern A's runtime cost is indistinguishable from the all-native baseline: its total time matches the reference at 0.133 s, and its slightly

DOI 10.3384/ECP219 153

faster simulation time (0.123 s vs. 0.127 s) compensates for a small additional initialization overhead. The FMI Model Exchange boundary, when the FMU contains only a single control block, imposes no measurable penalty on the host's RMS solver. Second, Pattern B costs roughly 2.5× as much as the baseline (0.322 s vs. 0.127 s in simulation time). The increase is concentrated in the simulation phase rather than initialization, and reflects the work the host integrator does evaluating the full GENROU state vector and the SCRX regulator through the FMU's right-hand-side function at each step. For the disturbance studied here the absolute cost is still well under a second, but the scaling matters: a Pattern B FMU bound into a multi-machine study would amplify this overhead in proportion to the number of FMU-resident machines.

Initialization time is a small fraction of total cost in all three configurations (5–10% for the FMU-based cases, 5% for the baseline) and does not differentiate the patterns meaningfully. FMU compile time — a one-time cost paid at design iteration rather than per-simulation — is several seconds for the SCRX-only FMU and tens of seconds for the GENROU+SCRX FMU; it is not included in Table 2 because it does not affect runtime simulation cost, but it does affect development turnaround and is worth noting for practitioners.

5 Conclusions and Future Work

This paper documented an end-to-end workflow for deploying OpenIPSL models into DIgSILENT PowerFactory as Model-Exchange FMUs, with two coupling granularities: a controller-only pattern exchanging scalar AVR signals with a native synchronous machine, and a full-machine pattern coupling through a voltage/power boundary. Validation against PowerFactory's native PSS®E SynMach SCRX showed clean agreement under the controller-only pattern; the full-machine pattern matched in steady state but exhibited a higher field-voltage peak, revealing machine-level modeling differences the controller-only pattern does not expose.

The immediate next step is to disambiguate the candidate mechanisms behind the Pattern B divergence through sweeps over disturbance amplitude (saturation), filter time constant (boundary dynamics), step size (host-integrator interaction), and state-vector instrumentation (initialization residuals). The workflow extends naturally to other OpenIPSL component classes—governors and stabilizers as Pattern A analogs, wind, photovoltaic, and HVDC models as Pattern B analogs—and to multi-machine and aggregated FMUs. Further out, three Modelica Association developments (Junghanns et al. 2021) would expand its reach: FMI 3.0 clocks and intermediate update would relax the algebraic-coupling tightness behind Pattern B's boundary filters; SSP would manage studies that bundle multiple FMUs; and eFMI would extend the deployment path from simulation to embedded controller hardware.

Acknowledgements

This paper is in part, based upon work supported in whole or in part by the Elia Group's and Energinet's "Research Challenge for Innovation in Power Systems", managed by 50Hertz Transmission GmbH and Energinet, whose support is gratefully acknowledged.

References

Bastian, Jens et al. (2011-03). "Master for Co-Simulation Using FMI". In: *Proceedings of the 8th International Modelica Conference*. Dresden, Germany: Linköping University Electronic Press, pp. 115–120. DOI: 10.3384/ecp11063115.

Basumallik, Sagnik et al. (2025-09). "Enhancing Large-Scale Power Systems Simulations through Functional Mock-up Unit-based Grid-Forming Inverter Models". In: *Proceedings of the 16th International Modelica & FMI Conference*. Lucerne, Switzerland: Linköping University Electronic Press.

Baudette, Maxime et al. (2018). "OpenIPSL: Open-Instance Power System Library — Update 1.5 to "iTesla Power Systems Library (iPSL): A Modelica Library for Phasor Time-Domain Simulations"". In: *SoftwareX* 7, pp. 34–36. ISSN: 2352-7110. DOI: 10.1016/j.softx.2018.01.002.

Blochwitz, Torsten et al. (2011-03). "The Functional Mockup Interface for Tool Independent Exchange of Simulation Models". In: *Proceedings of the 8th International Modelica Conference*. Dresden, Germany: Linköping University Electronic Press, pp. 105–114. DOI: 10.3384/ecp11063105.

Fritzson, Peter et al. (2018-10). "The OpenModelica Integrated Modeling, Simulation and Optimization Environment". In: *Proceedings of the 1st American Modelica Conference*. Cambridge, MA, USA: Linköping University Electronic Press, pp. 206–219. DOI: 10.3384/ECP18154206.

Gomes, Cláudio et al. (2018-05). "Co-Simulation: A Survey". In: *ACM Computing Surveys* 51.3, pp. 1–33. DOI: 10.1145/3179993.

Islam, Md Shamimul et al. (2023). "Modeling Specialized Electric Power Generators, Excitation Systems and Prime Movers Used by North American Utilities". In: *Proceedings of the 15th International Modelica Conference*. Aachen, Germany: Linköping University Electronic Press.

Junghanns, Andreas et al. (2021-09). "The Functional Mock-up Interface 3.0 — New Features Enabling New Applications". In: *Proceedings of the 14th International Modelica Conference*. Linköping, Sweden: Linköping University Electronic Press, pp. 17–26. DOI: 10.3384/ecp2118117.

Meer, Arjen A. van der et al. (2019-09). "Towards Scalable FMI-based Co-simulation of Wind Energy Systems Using PowerFactory". In: *2019 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe)*. Bucharest, Romania: IEEE, pp. 1–5. DOI: 10.1109/ISGTEurope.2019.8905616.

Vanfretti, Luigi, Tin Rabuzin, et al. (2016). "iTesla Power Systems Library (iPSL): A Modelica Library for Phasor Time-Domain Simulations". In: *SoftwareX* 5, pp. 84–88. ISSN: 2352-7110. DOI: 10.1016/j.softx.2016.05.001.

Vanfretti, Luigi and Chayan R. Roychowdhury (2024). *Power System Modeling for Identification and Control Applications Using Modelica and OpenIPSL*. Technical Report TR2024-112. Cambridge, MA, USA: Mitsubishi Electric Research Laboratories (MERL).

DOI **10.3384/ECP219 153**

Discontinuity-Aware Solvers for Vapor-Compression Cycles

Shreyas Ekanathan[1, 3] Avinash Subramanian[1] Oscar Smith[1] Yingbo Ma[4] Christopher Rackauckas[1] Christopher R. Laughman[2*]

[1]JuliaHub, Inc. {shreyas.ekanathan, avinash.subramanian, oscar.smith, chris.rackauckas}@juliahub.com
[2]Mitsubishi Electric Research Laboratories, Cambridge, MA, USA laughman@merl.com
[3]University of Texas at Austin
[4]Independent Researcher

Abstract

Physics-based models of vapor-compression cycles are often formulated as large, stiff DAE systems with derivative discontinuities introduced at the points of phase change. We use an implementation of a discontinuity-aware ODE and DAE solver scheme that uses domain-specific, user-declared information to compute appropriate time steps for more efficient integration. This scheme is made open-source within `DifferentialEquations.jl`. We demonstrate how discontinuity information can be specified symbolically in a declarative modeling language such as `Dyad` and passed to the solver to address the numerical challenges posed. We apply this discontinuity-aware technique to two applications: First, a scalar ODE with two discontinuities, and then a model of a full vapor-compression cycle. The discontinuity-aware solver yields significantly fewer step rejections and improved solver times, particularly with the `Rodas5P` solver at loose tolerances, as compared to standard error control, suggesting that exposing physical discontinuity information from equation-oriented models to adaptive solvers can improve practical simulation performance without requiring reassembly of the model. *Keywords: Vapor-compression cycles, HVAC, Differential Equations, Dyad*

1 Introduction

Vapor-compression cycles (VCCs) represent a key technology that serves as a foundation for many heating and cooling systems used in residential, commercial, and industrial buildings today, such as in heat pumps and air-conditioning systems. These systems play an essential role in maintaining comfort in buildings and regulating industrial thermal processes, and also provide an important capability in the continuing trend of electrification as a means of integrating renewable energy sources into the electrical grid. Given ongoing climate trends, these systems are becoming more common throughout the world in both existing and emerging markets.

Considering the wide geographic diversity of equipment installations and the range of applications, model-based design processes for equipment that incorporates vapor-compression cycles is becoming increasingly important to manage their dynamic behavior in the face of changing operating conditions. Whereas common design processes for these cycles tend to focus on experimental trial-and-error methodologies, system-level modeling and simulation is a valuable tool when used to pursue high system-of-systems efficiencies and reduce the dependence of the design process on time-consuming and expensive experimental laboratory work. These tools can improve the design process across many different system aspects, including refrigerant selection, component sizing, control system design, and initial design evaluations in the larger system-of-systems context.

A variety of approaches have been used to describe the spatially and temporally varying behavior of these systems as a set of computational models (Rasmussen 2012; Rasmussen and Shenoy 2012). While the underlying physics of these systems might be most accurately characterized as a set of partial differential algebraic equations (PDAEs), these are often simplified to form either a set of ordinary DAEs or a set of ordinary differential equations (ODEs). DAEs often arise in these systems due to the algebraic relations between the refrigerant properties, as well as from algebraic equations from flow and energy constraints formed when interconnecting component-based models. This modeling approach has been popularized by the Modelica language (Modelica Association 2023) and recently by the Dyad language (Rackauckas, Abdelrehim, et al. 2025), in which component models are created using an equation-oriented declarative language, and then interconnected to form larger system-level models. We employ this approach to construct these VCC models, which are based on well-known previous work (Qiao et al. 2015).

While a substantial body of work exists for the numerical solution of general DAEs (Brenan et al. 1995; Petzold 1982), the numerical solution of DAE models of VCCs pose a number of distinct challenges. These models are often quite large, due to the discretization of the heat exchangers into a number of control volumes to approximate the behavior of the partial differential equations describing the conservation of mass, momentum, and energy within the refrigerant flow and across the pipe wall. In addition, these models contain a variety of nonlinear relationships,

DOI 10.3384/ECP219 159

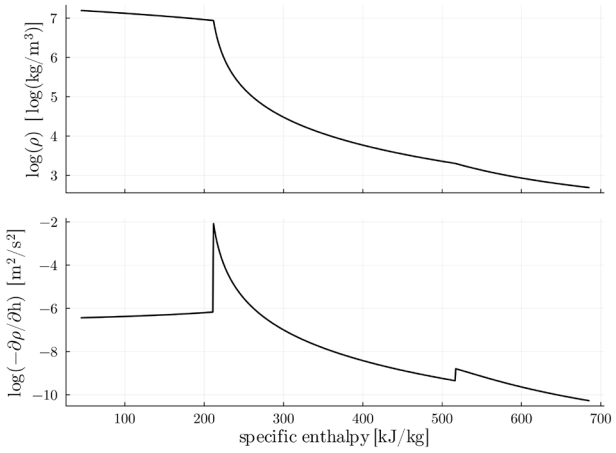


Figure 1. Plot of $\log(\rho)$ and $\log(-\partial\rho/\partial h|_P)$ for the refrigerant R32 at 1 MPa over a range of specific enthalpies, illustrating the derivative discontinuities at the bubble and dew points.

such as between the refrigerant properties (e.g., temperature, density) or in the closure relations relating the pressure drop to the refrigerant flow, which must be accommodated when solving these models.

The C^0 behavior of the refrigerant in the two-phase (condensing or evaporating) flows also introduces numerical challenges to the solver. This behavior is observed when integrating the differential equations related to the conservation of mass, which are often formulated in terms of the density (as the cross-sectional area of the tubes in the heat exchangers are constant). As seen in the example provided in Figure 1, this density has discontinuous derivatives at the boundaries of the two-phase region, known as the bubble point (between the liquid and two-phase regions) and the dew point (between the two-phase and vapor regions). Numerical solvers that integrate the differential equations of the model are often designed under the assumption that the underlying model is smooth, and may either become extremely slow when encountering such derivative discontinuities, or even crash or produce erroneous output in these situations. This is because the solver steps past the discontinuity, incurs a large local error, rejects the step, reduces the step size, and repeats this process many times before successfully crossing the trouble spot (Gear and Østerby 1984).

Several techniques have been developed for solving dynamical systems with discontinuities (Dieci and Lopez 2012), which arise in two distinct varieties: time-triggered and state-triggered. Time-triggered discontinuities occur at prescribed times that are known in advance and so are easier to handle, as the integrator can step to the discontinuity location and restart integration (Mørk et al. 2025). State-triggered discontinuities, on the other hand, require the determination of the time location of the discontinuity during integration. While most numerical integration methods are constructed on the assumption that the solution is smooth, these discontinuities violate this assumption. Gustafsson illustrates the effect of these discontinuities by showing that a naive integrator can spend a lot of computation on repeated step rejections to handle the large error introduced, whereas an informed integrator may directly step to it (Gustafsson 1994).

A number of works have addressed the numerical integration of dynamical systems with state-triggered discontinuities. Carver (Carver 1978) and Cellier (Cellier 1979) published seminal work on marking the discontinuity using zero-crossing functions, which are checked by the integrator for sign changes that indicate when the discontinuity is crossed. After crossing the discontinuity, the integrator is restarted with the chosen set of equations for the branch. One distinct mode of failure for this approach is that if a discontinuity is crossed twice within an integrator time step, or if two separate discontinuities lie close together, they may be missed. Park and Barton address this shortcoming by presenting an approach to search the entire timestep by leveraging interval arithmetic techniques to guarantee detection of all discontinuities (Park and Barton 1996).

Otter et al. discuss the implementation of this zero-crossing function technique in Modelica compilers, where a conservative approach is taken: every relation (with operators such as less than, greater than etc.) between continuous variables is assumed to introduce a discontinuity, and therefore automatically triggers a state event unless the modeler mentions otherwise (Otter et al. 1999). Fredriksson et al. present an implementation in Assimulo that can be attached to any integrator (Fredriksson et al. 2014) and is compatible with the FMI standard. In addition, rather than relying on a sign-change, they adopt a domain-based formulation whereby a change between the domains $g < 0$ and $g \geq 0$ is detected. This approach handles a failure mode of the sign-change test for the case where a discontinuity is triggered by the function reaching zero and staying there (as can happen in practical examples such as cutting power to a clutch). In other related work, Tummescheit and Eborn implement crossing functions for multiphase thermal-fluid flow with chemical reactions (with external property models) that allow numerically robust detection of new phases by using a "deviation index" which is the normalized distance to a phase-separation surface (Tummescheit and Eborn 2002).

In the present work, several fundamental ingredients of the solver algorithm are taken from this existing literature. We also use zero-crossing functions are used to mark discontinuities which are then detected by relying on a sign-change, and the location of the discontinuity is determined by performing a root finding search on the integrator's dense interpolant. However, the solver algorithm implements several unique features. First, the solver uses the zero-crossing functions to discover a discontinuity even on step rejections, not merely on step acceptances as is done in alternative approaches. This scheme, designed to handle the anticipated high error in rejected steps, is a key contribution of this paper. For example, the retried

DOI **10.3384/ECP219 159**

step size after rejection is set at half the rejected step size, which empirically yields the best results for the cases studied. Finally, the proposed algorithm is fully composable with any adaptive methods (such as `Rodas5P`, `Tsit5`, and `RadauIIA5` in the Julia SciML ecosystem).

We make the following contributions:

1. We implement a discontinuity-aware ODE and DAE solver that uses domain-specific (user-declared) information to compute appropriate time steps for more efficient integration. This solver is provided open-source in the Julia SciML ecosystem within `DifferentialEquations.jl` (Rackauckas and Nie 2017). Existing schemes only take advantage of such information on accepted steps, while we extend the scheme to function on rejected schemes as well.

2. We apply this solver algorithm to both a scalar ODE with two discontinuities as well as a vapor-compression cycle. We show that discontinuity-awareness results in significantly fewer step rejections and improved solver times, particularly with the `Rodas5P` solver at loose tolerances, compared to solvers using standard error control methods.

3. Lastly, we demonstrate how the domain-specific discontinuity information can be specified symbolically in a declarative modeling language (such as Dyad) and passed on to a numerical solver ecosystem.

The paper is organized as follows: Section 2 presents the two exemplar systems, consisting of a scalar ODE system with two discontinuities and a vapor-compression cycle model with varying compressor controls after a zero-flow startup. Section 3 describes the discontinuity-aware solver algorithm and contrasts with the standard solver, Section 4 presents the results on the two case studies, and Section 5 presents conclusions and directions for future work.

2 Exemplar Systems

2.1 Scalar ODE system

First, we study the following discontinuous ODE system with two state discontinuities:

$$\dot{x} = f(x,t) = \begin{cases} 2x, & x < 1 \\ x + 0.2, & 1 \le x < 2 \\ -4x + 12, & x \ge 2 \end{cases} \tag{1}$$

with initial condition $x(0) = 0.1$ and $t \in [0, 2.5]$.

The solution is plotted in Figure 2, illustrating the fact that the derivative discontinuities produce kinks in the solution at $x = 1$ and $x = 2$. This simple system allows us to explain the stepping dynamics and differences versus a standard ODE solver, as discussed in Section 4.1.

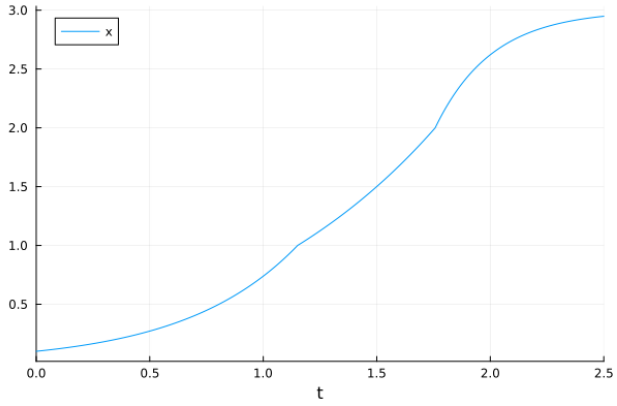


Figure 2. Solution to the toy problem. Note the effects of the derivative discontinuity as the solution crosses $x = 1$ and $x = 2$.

2.2 Vapor-compression cycle model

Figure 3 presents a vapor-compression cycle modeled in Dyad. It consists of a positive-displacement compressor, an evaporator and a condenser modeled as a tube-fin heat exchangers with a cross-flow configuration, and a linear expansion valve. The compressor and valve are described via algebraic relations since their dynamics are much faster than the rest of the system. The tube-fin heat exchanger comprises a 1D refrigerant thermal-fluid flow model with a finite-volume discretization along the flow axis consisting of mass, momentum and energy balance equations. Pressure and enthalpy states are selected as differential states to be integrated, while other refrigerant properties are computed from these states using fitted spline-based property models. The heat transfer through the tube walls to the air side is modeled using appropriate efficiency correlations, and the moist air flow has a cross-flow configuration over the finned tubes. A complete description of model is available in (Laughman and Qiao 2017) with a summary in the Appendix.

The Dyad model is developed using the `Dyad HVAC Components` library (JuliaHub 2026), in which reusable subcomponents are developed and interconnected. The Dyad compiler then reduces the full model to the form of a semi-explicit DAE (3) through alias elimination, index reduction, and tearing. This DAE is then passed to a numerical solver from an ecosystem such as `DifferentialEquations.jl` (Rackauckas and Nie 2017) to solve the equations of the form

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}\big(\mathbf{x}, \mathbf{y}; u(t)\big) \tag{2}$$

$$\mathbf{0} = \mathbf{g}\big(\mathbf{x}, \mathbf{y}; u(t)\big), \tag{3}$$

with $u(t)$ representing the controller inputs (such as compressor speed and valve position), $\mathbf{x}$ representing the differential states (such as refrigerant enthalpies and pressures in each control volume), and $\mathbf{y}$ denoting algebraic states (e.g., heat duties, temperatures and moist-air states). For this case study, compressor speed inputs are chosen so

DOI 10.3384/ECP219 159

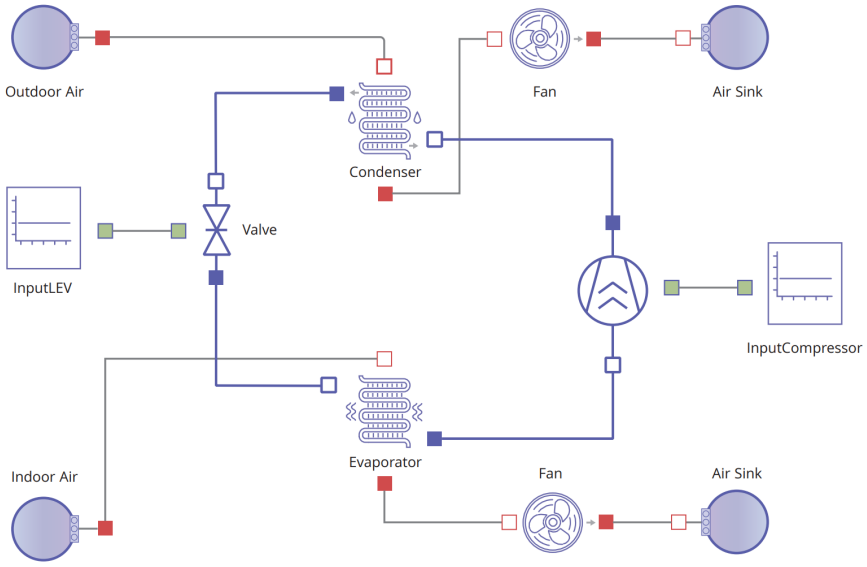


Figure 3. Vapor-compression cycle model in Dyad

as to drive the system through a cold-start, low-load, peak-load, and partial-shutdown regimes in succession without ever reaching a steady-state, so that several system states repeatedly traverse the refrigerant's two-phase region where discontinuities arise.

Dyad provides a native mechanism for attaching continuous events to an assembled system model. These events are specified declaratively using Dyad's symbolic language, and thus provide an ergonomic interface for the modeler to communicate domain-specific information to the ODE/DAE solver (1). These continuous events are then passed as callback functions to the integrator and are triggered when the integrated state satisfies the corresponding predicate. Mathematically, these can be written in terms of guard functions $g_j(\mathbf{x})$ which indicate points of discontinuity. We attach two such guards in each refrigerant control volume, one at each phase boundary:

$$\begin{aligned} g_{\text{bub}}(\mathbf{x}) &= h - h_{\text{bub}}(P) = 0 \\ g_{\text{dew}}(\mathbf{x}) &= h - h_{\text{dew}}(P) = 0, \end{aligned} \tag{4}$$

where $h_{\text{bub}}(P)$ and $h_{\text{dew}}(P)$ are the bubble and dew enthalpies evaluated at the particular pressure P, and h is the enthalpy state.

```
component RefrigerantMedium
  # parameters go here
  # variables go here (including P and h
    )
relations
  # property relations go here
  discontinuityAt(h = h_bub(P))
  discontinuityAt(h = h_dew(P))
end
```

Listing 1. Dyad syntax for declaring the phase-boundary events of Eq. (4) in a refrigerant medium model. Parameters, variables and property relations are omitted.

In this case study, we use eight control volumes per heat exchanger giving rise to thirty two events. The discontinuity-aware solver evaluates this guard information if a step is rejected, which is useful in determining subsequent candidate time steps.

3 Discontinuity-Aware Solver Algorithm

The discontinuity-aware solver is partially inspired by the work on delay differential equations (DDEs) by Guglielmi and Hairer (Guglielmi and Hairer 2008), which proposes a scheme for handling the breaking points in DDEs. These breaking points can be treated similarly to discontinuities in the right hand side of ODEs and DAEs, motivating the scheme presented here.

The new scheme exposes discontinuity information to the step-size controller thereby enabling cleaner navigation around them. The right flowchart in Figure 4 presents the improved adaptive time-step control logic, which is the core feature of the discontinuity-aware solver. In comparison, the left flowchart presents the time-step control without discontinuity-awareness. Of particular note is the fact that the current state of the software only runs on accepted steps, and this is our baseline for comparison. The novelties we introduce are manifest in the behavior on rejected steps, as described below.

Both solvers first proceed by taking a trial step as appropriate for the method, giving a test value $\hat{x}$. After taking the step, the solver uses a lower-order Runge-Kutta solution formed from the same stage evaluations to derive a lower-order estimate $\tilde{x}$. A normalized error estimate for the step is thus derived as

$$E = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{\hat{x}_i - \tilde{x}_i}{\text{atol} + \text{rtol}\cdot\max(|\hat{x}_i|, |\tilde{x}_i|)}\right)^2}. \tag{5}$$

DOI 10.3384/ECP219 159

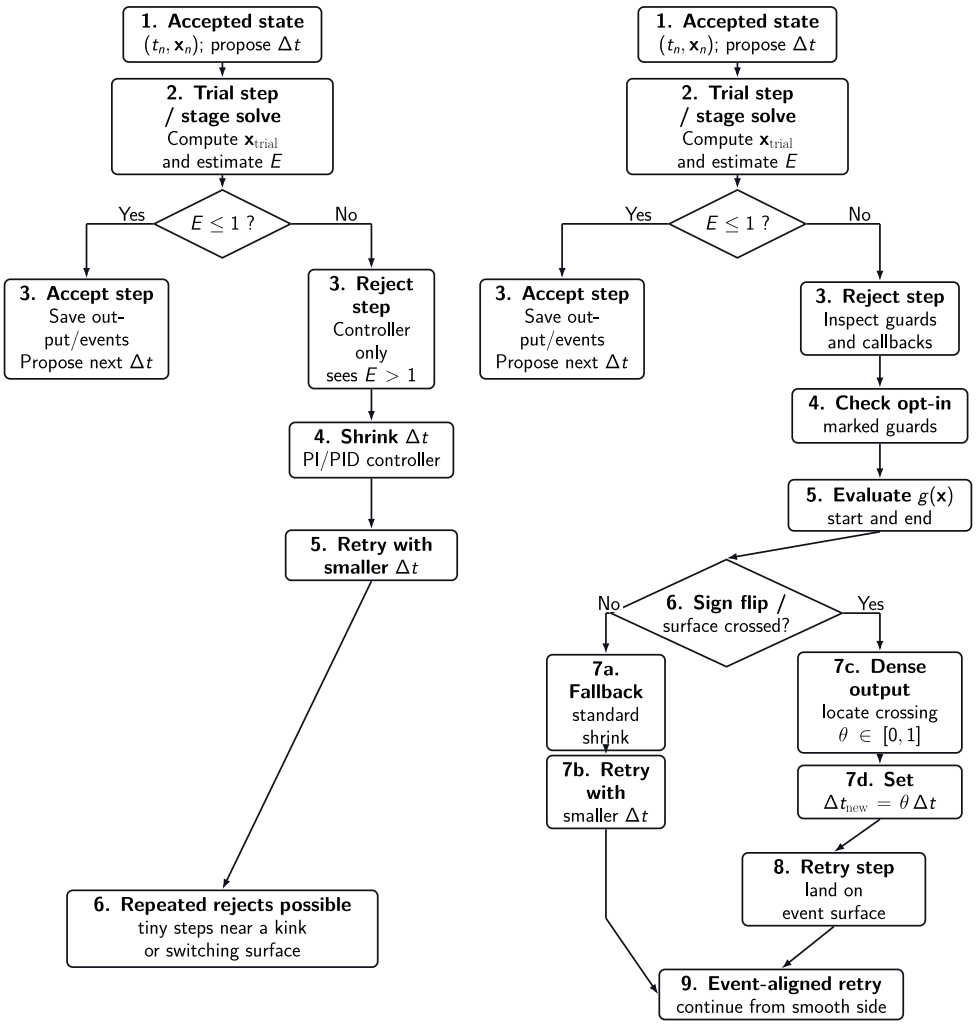


Figure 4. Adaptive time-step control with and without discontinuity-aware logic. The left branch applies the standard rejected-step controller. The right branch uses the marked guard functions together with a nonlinear solve on the dense interpolant to recompute a candidate timestep on the earliest detected discontinuity crossing.

In the event that the error is less than 1, both schemes will accept the step and proceed to the next step. However, the key difference occurs on step rejections, which we will now further explore.

3.1 Traditional step-size controllers

Many step-size controller schemes exist, but the general idea of using heuristic measures to predict the appropriate step size for the next step is often consistent from scheme to scheme. For instance, the Gustafsson controller scheme (Gustafsson 1991) computes the new step size via the following heuristic:

$$h_{\text{new}} = \frac{f_{\text{safe}} \cdot h}{E^{1/(p+1)}}, \tag{6}$$

where f_{safe} is a safety factor typically taken around 0.9, h was the current step size attempted, E was the error of the current step, and p is the order of the method.

This scheme works well when the system is smooth, but derivative discontinuities lead these heuristic estimates to perform very poorly, with clusters of step rejections around the discontinuities. For this reason, we introduce our discontinuity-aware step-size controller to handle these situations more robustly.

3.2 Discontinuity-aware step-size controller

We expose discontinuities via callbacks (Rackauckas, Nie, and SciML Contributors 2024), which describe discontinuity functions in two main forms:

1. Continuous callbacks: A single discontinuity function $g(x)$, which triggers an event at points where $g(x) = 0$.
2. Vector Continuous callbacks: A vector of conditions, allowing multiple discontinuity surfaces to be simultaneously monitored.

DOI **10.3384/ECP219 159**

These are exposed to the step size controller via the integrator interface, allowing the identification of discontinuities in our scheme. Upon opting to reject a step, we perform the following analysis on the callbacks:

Identifying event occurrences: Consider a step from t to $t+h$, with $f(t) = x_t$, $f(t+h) = x_{t+h}$. For a given callback $g(x)$, a root of g occurs if $g(x_t)$ and $g(x_{t+h})$ have different signs by the intermediate value theorem. Then, $g(x_t) \cdot g(x_{t+h}) < 0$ provides a sufficient condition for detecting the occurrence of an event. We evaluate this guard on all callbacks and only proceed to the subsequent steps on those that had event occurrences; if no events are detected, we fall back to the standard step-size rejection scheme previously described.

Identifying the event locations: For state-dependent callbacks previously described, we seek the time t_{event} at which $g(f(t_{\text{event}})) = 0$. This time is determined by using the dense interpolant for the solver, either derived specially for the integrator or using a Hermite interpolation, and an interval nonlinear solve is done to determine the location of the event using a bisection-like search. This gives a point in time at which the event occurred, and we choose the first such event to step to directly. We note that the dense interpolant is built from data already computed during the rejected trial step, so does not require either full function evaluations, Jacobian evaluations or matrix factorizations. However, given that the error was already high on the step, the interpolant will not be perfect, and hence we expect some numerical error with the computation of the event, but even with this error, this scheme causes the timestep to be chosen much closer to the discontinuity than a standard PI timestepper which is unaware of the discontinuity's existance.

Given this error, we choose to mildly bias the result of the solve, θ, via the following formula:

$$\theta = (0.5 + 0.4 \cdot \sin(\pi \cdot (\theta - 0.5)))$$

Or equivalently in polynomial form (to reasonable error, for faster computational purposes),

$$\theta = 0.5 + 1.2372 \cdot (\theta - 0.5) - 1.7487 \cdot (\theta - 0.5)^3$$

The motivation for this biasing is that we want each step to maximize the information yielded, hence an extremely large step (close to the prior rejected step) or a very small step is suboptimal. Therefore, we choose to try and bias our steps towards roughly `dt`$/2$, while also maintaining use of the nonlinear solvers results.

Resumption of Smooth Integration: Upon crossing the discontinuity, we reset error history of the controller. This is done to indicate the resumption of integration in a new domain, in which prior assumptions may not apply. This allows us to more efficiently determine the appropriate step-sizes for this regime.

In addition to this error reset, we also restrict the first step after a discontinuity forced step. This is done out of caution, given that we expect that we have just navigated the discontinuity, but in the event that we have not, we want a limited step to again maximize the information gain from this rejected step.

More technically, suppose our first rejected step ends at time t. Then, after our discontinuity step, we restrict the subsequent step to $\min(dt, (t - t_{curr})/4$, where dt is the proposed step-size. Thus, if our intended step takes us beyond the area where we think the discontinuity is, to prevent unnecessary failures, we limit this first step before continuing integration as normal.

We further note that this discontinuity-aware time-step controller is composable with a variety of solvers in the `DifferentialEquations.jl` (Rackauckas and Nie 2017) ecosystem.

4 Results and Discussion

In this work, we study the application of this scheme to three algorithms: `Rodas5P` (a Rosenbrock method applicable to both DAEs and ODEs), `RadauIIA5` (a fully-implicit Runge-Kutta method applicable to both DAEs and ODEs), and `Tsit5` (an explicit Runge-Kutta method applicable to ODEs) for both of the exemplar models.

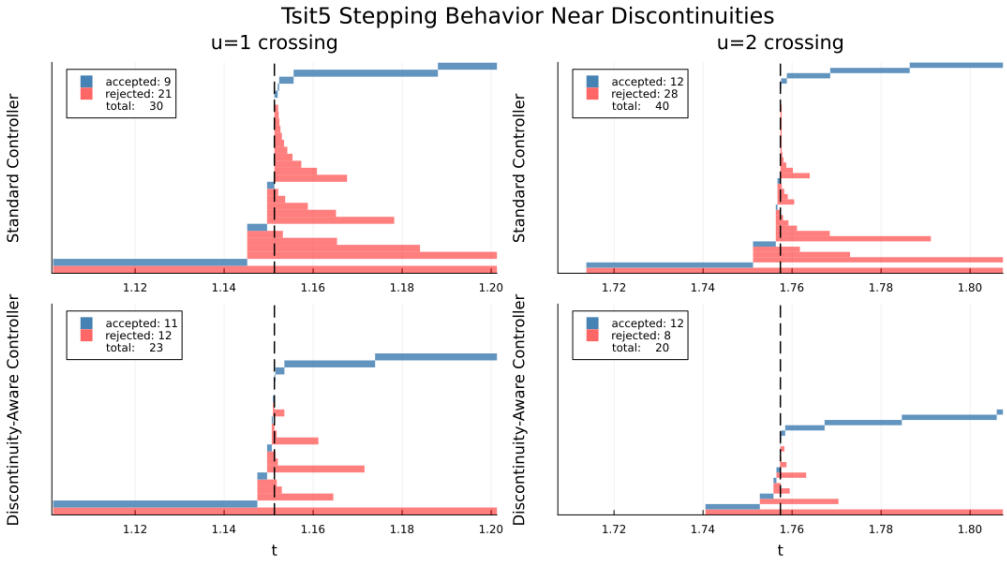
We note that in both the standard and discontinuity-aware solver configurations, the solver has access to zero-crossing functions at all times. The distinction is that the standard controller only uses this information on accepted steps, while our scheme additionally exploits it on rejected steps.

4.1 Scalar ODE system

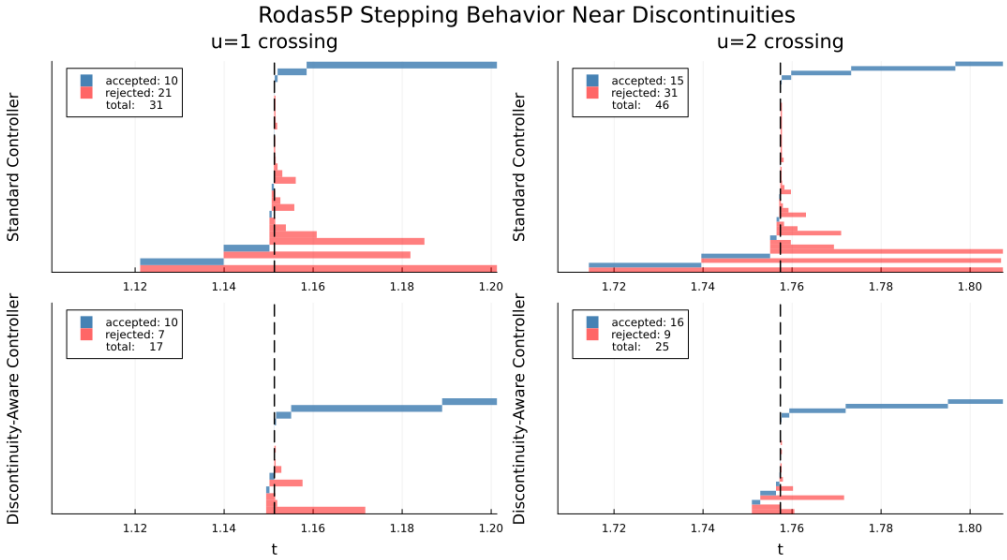
Figure 5 presents the time step plots of the `Tsit5` and `Rodas5P` solvers with a discontinuity-aware controller versus a standard controller. On each plot, the left pane shows the time steps taken near the $u = 1$ discontinuity while the right pane shows this near the $u = 2$ discontinuity. The y-axis shows the number of trial time steps. Each horizontal bar spans a proposed time step with the bars classified by whether the time step was accepted (blue) or rejected (red). Studying the right pane of Figure 5 (a), we observe that the first step is rejected in both cases. Both the discontinuity-aware and standard solvers shrink the next proposed time step. However, only the discontinuity-aware solver computes an appropriately-sized time step; while it is still rejected, gives this controller much more information to proceed. The standard solver shrinks the time step proportional to the error of the step, resulting in the rejection of the next time step being rejected before the succeeding one is accepted. This pattern repeats several times as we note that the standard solver has significantly more (red) rejected time steps. This trend also continues with Figure 5 (b) as we observe that the discontinuity-aware solver requires significantly fewer trials to pinpoint the discontinuity and integrate past it so as to solve the ODE efficiently.

Figure 6 presents several solver metrics: wall time (ms), number of function evaluations, and number of

DOI **10.3384/ECP219 159**



(a) `Tsit5`



(b) `Rodas5P`

Figure 5. Time steps of the `Tsit5` solver and the `Rodas5P` solver. The y-axis shows the number of trial time steps, each horizontal bar spans a proposed time step with the bars classified by whether the time step was accepted (blue) or rejected (red). The locations of the discontinuities are marked by the vertical dashed line (note that these are only known in the state space; the corresponding location of the discontinuity in time is not known before solution)

rejected steps for both `Tsit5` and `Rodas5P` solvers with discontinuity-aware and standard time-step controllers. For both `Tsit5` and `Rodas5P`, we note that discontinuity-awareness results in significantly fewer rejected steps. Similarly, discontinuity-awareness results in fewer function evaluations which are a major contributor to solver times. Figure 7 presents a work-precision diagram: The x-axis denotes the L2-norm error of the solution versus a ground-truth solution while the y-axis denotes the solver time. The results on the work-precision diagrams build on this as the performance of the discontinuity-aware solver is consistently faster than the standard solver.

An interesting point to note is that the benefit for the implicit method, `Rodas5P`, appears larger than the benefit for the explicit method, `Tsit5`. This gives useful insight into when the discontinuity detection scheme adds value: on an implicit method, with a higher cost per rejected step, the discontinuity-aware step-size controller has a positive effect. On the other hand, while the scheme still helps for an explicit solver, we see a more modest improvement, as the rootfinding overhead introduced by the discontinuity detection scheme is non-trivial relative to the actual step cost.

DOI **10.3384/ECP219 159**

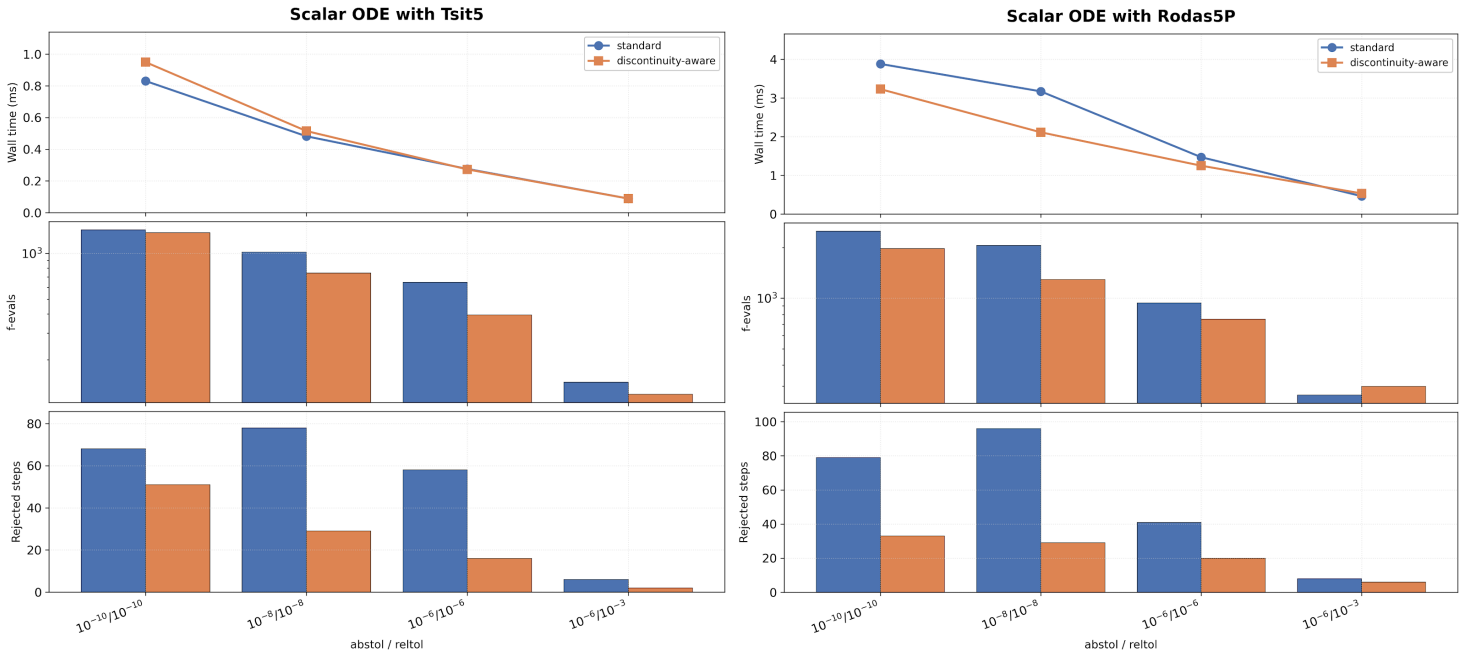
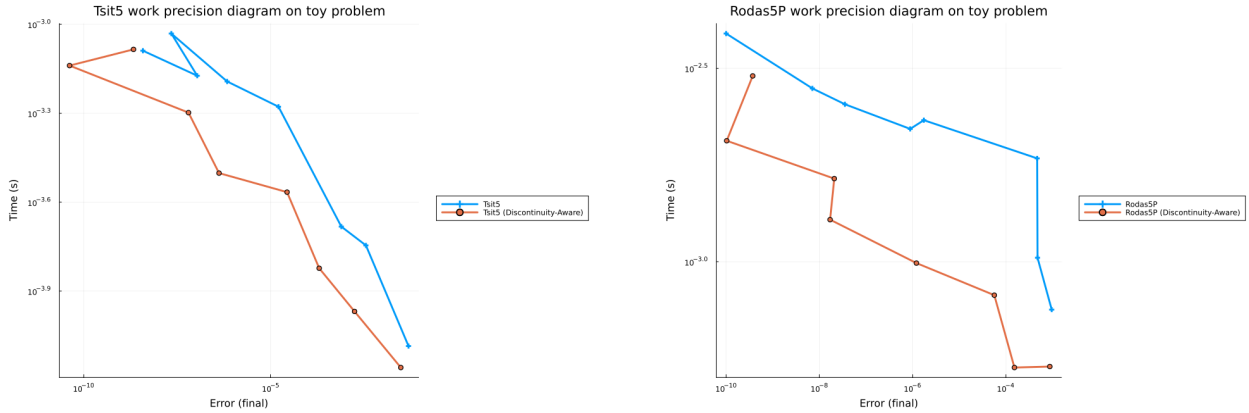


Figure 6. Wall time (ms), number of function evaluations and number of rejected steps at different absolute and relative tolerances with a discontinuity-aware and standard time-step controller. Metrics for `Tsit5` are shown on the left while `Rodas5P` are on the right.



(a) `Tsit5` work-precision diagram on the toy problem.

(b) `Rodas5P` work-precision diagram on the toy problem.

Figure 7. Work-precision diagrams for `Tsit5` and `Rodas5P` on the toy problem. Both plots are taken with reltol ranging from 10^{-2} to 10^{-9}, and abstol ranging from 10^{-4} to 10^{-11}. The vertical axis shows solve time and the horizontal axis shows the final error in the solution.

4.2 Vapor-compression cycle

Figure 8 presents the solver metrics for the vapor-compression cycle with the `Rodas5P` and the `RadauIIA5` solver. Note that our results are shown as averages over 15 runs at each tolerance level, as the solver performance is strongly dependent on the solver initialization; some solves perform poorly with discontinuity-awareness and cleanly otherwise, while the opposite also occurs. We therefore analyze general trends for the clarity of results, rather than the individual run variance.

We first analyze the results for `Rodas5P`. Considering the loose tolerance regimes, we observe performance improvements with the discontinuity-aware step-size controller, with 2x speedups on average across the low tolerances. The poor performance of the standard solver in these regimes likely stems from the fact that the standard controller takes large steps at loose tolerances and incurs high error around discontinuities. While the loose tolerances allow the solver to accept steps despite high error near discontinuities, the large error estimates cause the controller to dramatically shrink subsequent steps (even though the steps are accepted), leading to inefficient navigation of the discontinuity region.

DOI **10.3384/ECP219159**

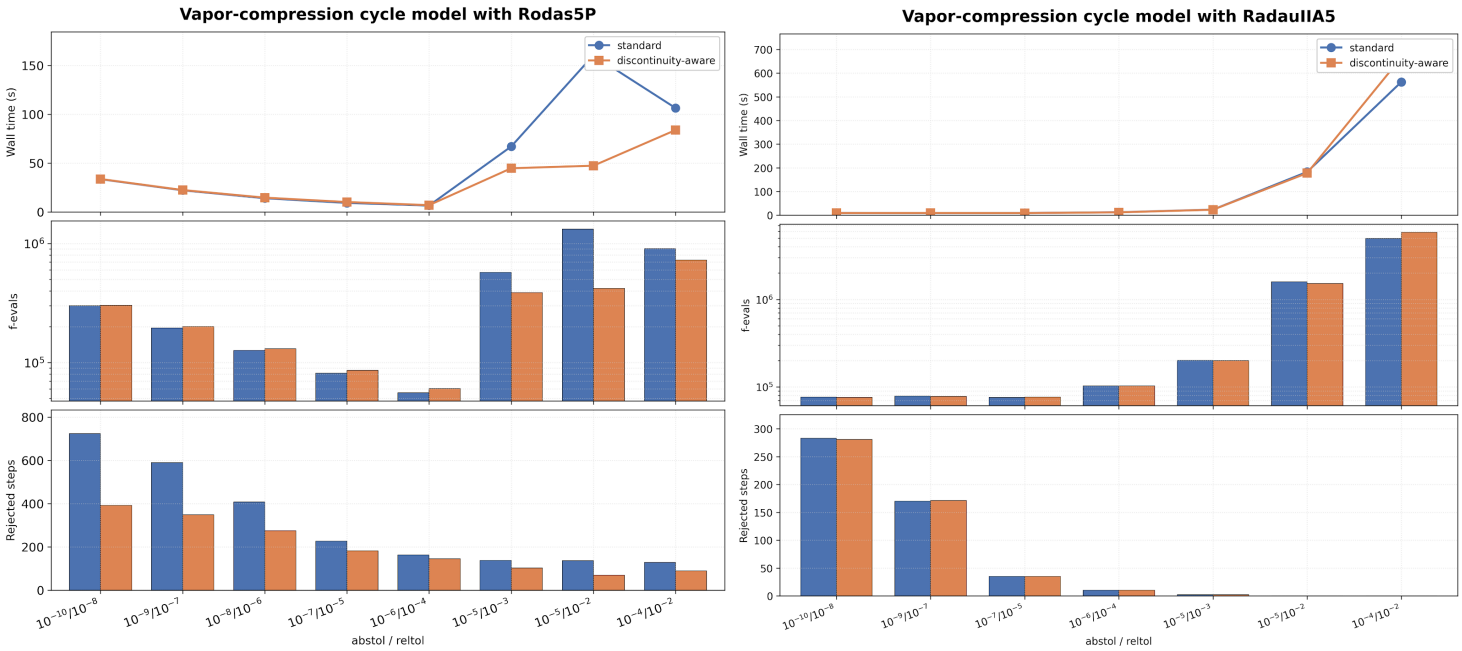


Figure 8. Wall time (ms), number of function evaluations and number of rejected steps at different absolute and relative tolerances with a discontinuity-aware and standard time-step controller. Metrics for `Rodas5P` are shown on the left while `RadauIIA5` are on the right. Metrics are calculated as the average over 15 runs with varying initialization conditions.

Further consideration of these results suggests that the full value of the discontinuity-aware scheme lies in its ability to prevent "stagnations", or larger clusters of tiny accepted steps around discontinuities, which cause solve times to blow up. This behavior was seen in the toy problem above, and the table below shows the tail behavior of these test runs on the full model.

	Median		P75		P90		Max	
abstol / reltol	**ND**	**D**	**ND**	**D**	**ND**	**D**	**ND**	**D**
Default	13.2	9.8	38.8	13.3	445.8	81.5	916.9	804.5
10^{-4} / 10^{-2}	42.0	69.1	132.2	117.1	168.5	170.4	651.3	217.3
10^{-5} / 10^{-2}	34.7	32.5	122.9	50.0	322.7	98.8	1245.9	150.5
10^{-5} / 10^{-3}	22.6	24.9	59.8	27.8	223.0	30.5	304.3	380.4
10^{-6} / 10^{-4}	6.3	6.9	6.4	7.0	6.9	7.2	9.4	7.6

Table 1. Rodas5P solve time (s) distributions over 15 runs on the complete HVAC model. ND = standard controller, D = discontinuity-aware controller. P75 represents the 75th percentile, while P90 is the 90th percentile.

Strong evidence of the discontinuity-aware scheme's benefits at high tolerances is manifest in this tail behavior. For example, the default tolerance is 5x faster at the 90th percentile of runs, and $10^{-5}/10^{-3}$ is 8x faster at the same percentile. Furthermore, the maximum solve time at $10^{-5}/10^{-2}$ is likewise 8x faster with discontinuity-awareness. The 75th percentile is about 2x faster across all of the tolerances.

While the differences are negligible in the median performance, but the discontinuity-aware scheme performs significantly better when failures and difficulties occur. This serves as key evidence for the benefits of this scheme, as the bounds for the worst case solve time are tighter than they are for the existing schemes. The successful recovery of a solve that took 20 minutes at $10^{-5}/10^{-2}$ in only 2.5 minutes with our scheme is the strongest indicator of this claim.

In comparison, we see that the discontinuity-aware scheme still rejects far fewer steps with `Rodas5P` at tight tolerances, but to negligible benefit, and even minor slowdowns. At these tight tolerances, both schemes are forced to take smaller steps; this setting favors the standard scheme, as the cost of rejections decreases. The benefits of our algorithm are minor here, but the overhead is also limited.

We note an interesting contrast between the results with `Rodas5P` and `RadauIIA5`. The differences between the standard and discontinuity-aware solvers are negligible across the board with `RadauIIA5`, with solve times being close to identical and step rejection percentages likewise very similar. Interestingly, the discontinuity-aware scheme often increases both step acceptances and step rejections, but the cost of this appears to be mitigated with each individual step being cheaper. This is apparent when considering the function evaluations with `RadauIIA5`, as there are minimal increases and even decreases in function evaluations despite the increase in steps. This is likely because of the more efficient navigation of discontinuities by the discontinuity-aware scheme, allowing the Newton iterations to converge more easily and hence reducing function evaluations.

At tight tolerances, `RadauIIA5` appears to be more robust than `Rodas5P`. This difference can be understood as a product of the fact that `RadauIIA5` is a

DOI 10.3384/ECP219 159

fully implicit Runge-Kutta method, while `Rodas5P` is linearly implicit. `Rodas5P` benefits dramatically from discontinuity-awareness at loose tolerances because its single linearization per step is particularly vulnerable to discontinuities, as it takes large steps and hence incurs severe error. On the other hand, `RadauIIA5` shows little benefit because the full Newton iteration more effectively handles the discontinuity internally, converging to a valid solution even across discontinuities, at the cost of more iterations but without catastrophic step rejection. Again, this is likely because both schemes are already exposed to discontinuities through the callback mechanism, and differ in how they utilize this information on rejected steps. For fully implicit methods like `RadauIIA5`, this appears to yield minimal benefit, while there are large gains for linearly implicit methods like `Rodas5P`. At tight tolerances, both take smaller steps, yielding similar results.

5 Conclusions

We have presented a discontinuity-aware ODE and DAE solver in which user-declared information is used to compute appropriate time steps at discontinuities, and is now available in the open-source `DifferentialEquations.jl` ecosystem. We also demonstrated how this information can be specified symbolically in the Dyad declarative modeling language and passed to the solver.

Benchmarking on a scalar ODE and a vapor-compression cycle DAE model shows that the scheme yields significantly fewer step rejections and improved solver times, with the largest gains observed for `Rodas5P` at loose tolerances, where we see speedups of up to 8x in the tail behavior of HVAC models. The contrast between `Rodas5P` and `RadauIIA5` reveals an important practical guideline: discontinuity-awareness provides the greatest benefit for linearly implicit methods at loose tolerances, where large step sizes make the solver particularly vulnerable to derivative discontinuities, while fully implicit methods are more naturally robust. Furthermore, the analysis of `Rodas5P` shows the critical value of the discontinuity-aware scheme in preventing large solve failures, which somewhat mitigates the challenges of difficult integration problems.

References

Brenan, Kathryn Eleda, Stephen L Campbell, and Linda Ruth Petzold (1995). *Numerical solution of initial-value problems in differential-algebraic equations*. SIAM.

Carver, MB (1978). "Efficient integration over discontinuities in ordinary differential equation simulations". In: *Mathematics and Computers in Simulation* 20.3, pp. 190–196.

Cellier, François E (1979). "Combined continuous/discrete system simulation by use of digital computers". In.

Dieci, Luca and Luciano Lopez (2012). "A survey of numerical methods for IVPs of ODEs with discontinuous right-hand side". In: *Journal of Computational and Applied Mathematics* 236.16. 40 years of numerical analysis: "Is the discrete world an approximation of the continuous one or is it the other way around?", pp. 3967–3991. ISSN: 0377-0427. DOI: https://doi.org/10.1016/j.cam.2012.02.011. URL: https://www.sciencedirect.com/science/article/pii/S0377042712000684.

Fredriksson, Emil, Christian Andersson, and Johan Åkesson (2014). "Discontinuities handled with events in Assimulo". In: *10th International Modelica Conference*. Linköping University Electronic Press, pp. 827–836.

Gear, Charles William and Ole Østerby (1984). "Solving Ordinary Differential Equations with Discontinuities". In: *ACM Transactions on Mathematical Software* 10.1, pp. 23–44. DOI: 10.1145/356068.356071.

Guglielmi, Nicola and Ernst Hairer (2008). "Computing breaking points in implicit delay differential equations". In: *Advances in Computational Mathematics* 29.3, pp. 229–247.

Gustafsson, Kjell (1991). "Control theoretic techniques for step-size selection in explicit Runge-Kutta methods". In: *ACM Transactions on Mathematical Software* 17, pp. 533–554. DOI: 10.1145/210232.210242.

Gustafsson, Kjell (1994). "Traps and pitfalls in simulation". In: *SIMS (Scandinavian Simulation Society) Simulation Conference*.

JuliaHub (2026). *Dyad HVACComponents Library*. https://help.juliahub.com/dyad/dev/stdlib/HVACComponents/. URL: https://help.juliahub.com/dyad/dev/stdlib/HVACComponents/.

Laughman, Christopher R and Hongtao Qiao (2017). "On the influence of state selection on mass conservation in dynamic vapour compression cycle models". In: *Mathematical and Computer Modelling of Dynamical Systems* 23.3, pp. 262–283.

Modelica Association (2023-04). *Modelica – A Unified Object-Oriented Language for Systems Modeling. Language Specification Version 3.6*. Tech. rep. Linköping: Modelica Association. URL: https://specification.modelica.org/maint/3.6/MLS.pdf.

Mørk, Jenny M., Tor Nordam, and Siren Rühs (2025). "Handling discontinuities in numerical ODE methods for Lagrangian oceanography". In: *EGUsphere*. preprint. DOI: 10.5194/egusphere-2025-2109.

Otter, Martin, Hilding Elmqvist, and Sven Erik Mattsson (1999). "Hybrid modeling in Modelica based on the synchronous data flow principle". In: *Proceedings of the 1999 IEEE International Symposium on Computer Aided Control System Design (Cat. No. 99TH8404)*. IEEE, pp. 151–157.

Park, Taeshin and Paul I Barton (1996). "State event location in differential-algebraic models". In: *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 6.2, pp. 137–165.

Petzold, Linda (1982). "Differential/algebraic equations are not ODE's". In: *SIAM Journal on Scientific and Statistical Computing* 3.3, pp. 367–384.

Qiao, Hongtao, Vikrant Aute, and Reinhard Radermacher (2015). "Transient modeling of a flash tank vapor injection heat pump system – Part I: Model development". In: *International Journal of Refrigeration* 49, pp. 169–182. ISSN: 0140-7007. DOI: https://doi.org/10.1016/j.ijrefrig.2014.06.019.

Rackauckas, Christopher, Anas Abdelrehim, et al. (2025). *Software-Defined Machines with Dyad*. Preprint. ResearchGate. DOI: 10.13140/RG.2.2.15807.57760. URL: https://www.researchgate.net/publication/398814645_Software-Defined_Machines_with_Dyad.

DOI **10.3384/ECP219 159**

Rackauckas, Christopher and Qing Nie (2017). "Differentialequations. jl–a performant and feature-rich ecosystem for solving differential equations in julia". In.

Rackauckas, Christopher, Qing Nie, and SciML Contributors (2024). *DifferentialEquations.jl Callback Functions*. https://docs.sciml.ai/DiffEqDocs/stable/features/callback_functions/. Accessed: 2026-05-31.

Rasmussen, Bryan P (2012). "Dynamic modeling for vapor compression systems—Part I: Literature review". In: *HVAC&R Research* 18.5, pp. 934–955.

Rasmussen, Bryan P and Bhaskar Shenoy (2012). "Dynamic modeling for vapor compression systems—Part II: Simulation tutorial". In: *HVAC&R Research* 18.5, pp. 956–973.

Tummescheit, Hubertus and Jonas Eborn (2002). "Chemical reaction modeling with ThermoFluid/MF and MultiFlash". In.

A Vapor-compression cycle model

A.1 Tube-fin heat exchangers

Both the condenser and the evaporator are tube-fin heat exchangers comprised of the refrigerant-side flow, the tube wall, and the air-side flow.

Refrigerant flow. The refrigerant flow is 1D thermal-fluid flow. Mass, momentum, and energy conservation are then governed by

$$\frac{\partial(\rho A)}{\partial t}+\frac{\partial(\rho A v)}{\partial x}=0, \tag{7}$$

$$\frac{\partial(\rho v A)}{\partial t}+\frac{\partial(\rho v^2 A)}{\partial x}=-A\frac{\partial P}{\partial x}-F_f, \tag{8}$$

$$\frac{\partial(\rho u A)}{\partial t}+\frac{\partial(\rho v h A)}{\partial x}=vA\frac{\partial P}{\partial x}+vF_f+\frac{\partial Q}{\partial x}. \tag{9}$$

Discretising each refrigerant circuit yields an ordinary differential equation of the form for the mass of volume j, for instance,

$$\frac{d(\rho_j V_j)}{dt} \;=\; \dot{m}_k-\dot{m}_{k+1}, \tag{10}$$

with the finite volume enthalpy flow evaluated by upwind descretization. The frictional pressure drop in both the single- and two-phase regions is taken as a scaled quadratic relation,

$$\Delta P \;=\; K\,\frac{(\Delta P)_0}{\dot{m}_0^2}\,\dot{m}^2, \tag{11}$$

and the refrigerant heat-transfer coefficient is assigned a constant value within each phase. The derivative discontinuities at the bubble and dew curves discussed in Section 2.2 enter the assembled DAE (3) through refrigerant property computations.

Tube wall. Conductive heat transfer through the property wall takes the form

$$\begin{aligned}(m_w c_w+m_f c_f)\frac{\mathrm{d}T_w}{\mathrm{d}t} &= \alpha_{\mathrm{ref}}A\,(T_r-T_w)\\ &\quad+\dot{m}_a[c_{p,a}(T_{a,\mathrm{in}}-T_{a,\mathrm{out}})+\\ &\quad(\omega_{a,\mathrm{in}}-\omega_{a,\mathrm{out}})\,\Delta h_{fg}.\end{aligned} \tag{12}$$

Air-side flow. The air stream crosses the fin side and exchanges both sensible and latent heat with the wall using replaceable heat and mass transfer correlations

A.2 Compressor

The compressor is a positive-displacement machine modelled algebraically, with no mass or energy storage. Its behaviour is set by two performance maps fitted to manufacturer data — a volumetric efficiency, which fixes the refrigerant mass flow, and an isentropic efficiency, which fixes the discharge enthalpy:

$$\eta_v(f,p_r,\ldots)=\frac{\dot{m}_{\mathrm{comp}}}{\rho_{\mathrm{suc}}V_f}, \tag{13}$$

$$\eta_{\mathrm{is}}(f,\ldots)=\frac{h_{\mathrm{dis,isen}}-h_{\mathrm{suc}}}{h_{\mathrm{dis}}-h_{\mathrm{suc}}}, \tag{14}$$

where V_f is the swept volume rate at shaft frequency f, $h_{\mathrm{dis,isen}}$ is the discharge enthalpy of the equivalent isentropic compression, ρ_{suc} is the suction-side density, and both efficiencies are maps in f and the pressure ratio p_r.

A.3 Expansion valve

The LEV is modeled as an isenthalpic, algebraic orifice without mass or energy storage. The mass flow follows the standard orifice relation

$$\dot{m} \;=\; C_v(\theta)\,A_v\,\sqrt{\rho_{\mathrm{in}}\,\Delta P}, \tag{15}$$

in which $C_v(\theta)A_v$ is the effective flow coefficient — a smooth, differentiable function of the valve position θ — ρ_{in} is the upstream density, and ΔP is the pressure drop across the valve.

DOI **10.3384/ECP219 159**

Toward Equation-Based Building Energy System Modeling in Julia: A Modular Prototype and Verification Study

Ruizhi Song[1] Mingzhe Liu[1] Zhiyao Yang[1] Zheng O'Neill[1]

[1] J Mike Walker' 66 Department of Mechanical Engineering, Texas A&M University

{ruzsng, mingzhe.liu, z.yang, zoneill}@tamu.edu

Abstract

A unified environment for equation-based modeling, optimization, and scientific machine learning makes Julia a promising platform for building energy control applications. Realizing the full potential of this emerging ecosystem, however, requires domain-specific modeling prototypes that use modeling as the entry point to an integrated workflow. This paper presents a lightweight Julia prototype comprising a set of modular weather, building thermal, and Heating, Ventilation, and Air-Conditioning (HVAC) components. The prototype is verified against reference models from the Modelica Buildings Library. Results demonstrate numerical agreement across cases of increasing system complexity, supporting the use of equation-based modeling for building energy systems and establishing a foundation for future building-domain modeling and control workflows in the Julia ecosystem.

Keywords: Equation-based modeling, HVAC, Building energy simulation, ModelingToolkit

1 Introduction

Building energy system modeling and control benefit from approaches that remain extensible across the full pipeline, from physical model formulation to simulation and control implementation (Wetter & Sulzer, 2024).

Compared with imperative programming implementations (e.g., C/C++, Fortran), equation-based modeling enables direct representation of governing equations and their organization into reusable hierarchical components without prescribing a fixed computational sequence (Wetter et al., 2016). This declarative structure facilitates symbolic analysis of the assembled system and leverages structural information, such as algebraic constraints, differential couplings, and event conditions, to support efficient numerical solution and large-scale nonlinear optimization. As a result, it enhances modeling flexibility and computational performance when supported by suitable symbolic transformations and numerical methods (Åkesson et al., 2010; Cellier & Kofman, 2006; Elmqvist et al., 2014).

As a representative equation-based modeling language, Modelica is a mature, openly standardized language with broad international support across industries. In the building domain, this foundation has been reflected by libraries such as the Modelica Buildings Library (MBL) (Wetter et al., 2014), which is a prominent and well-established open-source library for buildings, HVAC, and control-oriented applications (Qiu et al., 2024), suggesting Modelica as an important reference ecosystem in building energy system simulation.

Other equation-based modeling environments are also developing, such as ModelingToolkit (MTK) (Ma et al., 2022) in Julia, a high-performance language for scientific computing (Bezanson et al., 2017). MTK is an acausal equation-based modeling language embedded in a high-level language, allowing component-based models to be written directly in Julia and used together with surrounding non-modeling code. In practice, this integration enables equation-based models to be used with numerical solvers, optimization, machine learning and differentiable-programming tools, thereby supporting scientific machine learning (SciML) workflows.

The Julia SciML ecosystem has made Julia-based toolchains attractive for a range of applications. However, as the ecosystem continues to develop, important domain-specific gaps remain. Building-domain modeling prototypes that combine weather, envelope, and HVAC subsystems remain limited in Julia. Previous Julia-based studies have investigated vapor compression cycle models in connection with state and parameter estimation, sensitivity-related analysis, and surrogate generation (Deshpande & Laughman, 2023; Laughman et al., 2023; Rackauckas et al., 2022), or on case-specific dedicated outdoor air system simulations (Song et al., 2025) with limited modularity and generalizability. Hybrid architectures that couple equation-based models with machine-learning components have also been investigated, including physics-enhanced neural ODEs in a Modelica-based workflow (Hofmann & Mikelsons, 2025) and NeuralFMUs, which embed an FMU within a neural-network topology and train the resulting hybrid model in Julia (Thummerer et al., 2021). Both approaches could start from an existing equation-based model. Against this background, the availability of a domain-specific modeling prototype can facilitate full-pipeline exploration, since modeling serves as the entry point to subsequent workflows such as optimization, control, and machine learning.

DOI 10.3384/ECP219 170

Motivated by the gap, this paper presents a lightweight, open-source building energy system modeling prototype, BuildEML (v0.1.0), implemented in Julia language and openly available at https://github.com/BE-HVACR/BuildEML.jl. The prototype includes a limited set of weather, thermal, and HVAC models, and its implementation is comparatively verified against four reference cases from the MBL. The goal is to provide an initial domain-specific equation-based modeling prototype for building applications in the Julia ecosystem and to support subsequent evaluations and extensions.

2 Methods

2.1 Modeling and simulation workflow

In the presented work, the modeling and simulation workflow (Figure 1) combines symbolic model construction using ModelingToolkit.jl (Ma et al., 2022) v10.21.0 with time-domain simulation using DifferentialEquations.jl (Rackauckas & Nie, 2017) v7.17.0.

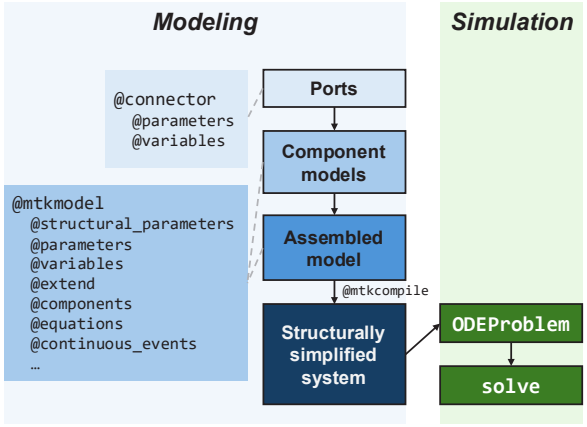


Figure 1. Schematic of the modeling and simulation workflow.

In this work, physical models are built via ModelingToolkit.jl as symbolic systems. Such systems can be constructed either through macro-based definitions, such as `@connector` (and `@mtkmodel` in MTK v10), or through function-based definitions that return equivalent symbolic systems represented by the `System` type. At the interface level, `@connector` is used to define variables exchanged between components, such as stream, flow, or regular variables. Physical components are then built using `@mtkmodel`, which organizes structural parameters, model parameters, variables, subsystems, and equations into reusable model units and supports model extension. This representation supports a hierarchical modeling workflow across connectors, individual components, and complete assembled systems.

After the full system is assembled and before numerical simulation, the symbolic model is transformed into a simulation-ready form through automatic symbolic compilation and structural simplification, which can be invoked, for example, via the macro `@mtkcompile`. This step examines the equation–variable structure, applies transformations such as alias elimination, tearing, and index reduction, and can help identify structurally underdetermined or overdetermined formulations before time integration (Ma et al., 2022). Modelica compilers perform extensive symbolic transformations of the equations when generating simulation code (Brück et al., 2002). In ModelingToolkit, these transformations are invoked through Julia functions operating on symbolic system objects in the host language, so the simplified system and the subsequently constructed numerical problem are Julia objects accessible to surrounding code.

For simulation, the structurally simplified system is converted into a numerical problem, typically an `ODEProblem`, for solution using DifferentialEquations.jl. DAE systems can also be converted to an `ODEProblem` when they can be expressed in mass-matrix form. The simplified system can be inspected through its unknowns and generated mass matrix to identify the retained differential and algebraic variables. Based on this structure, initial conditions are assigned to the differential states, with initial guesses provided for the algebraic variables when needed, together with the simulation time span and any parameter values that differ from their defaults. To numerically solve the problem, the solve function is called with a method selected from the broad range of solvers available in DifferentialEquations.jl for stiff and nonstiff systems, together with options such as output times and error tolerances.

2.2 Prototype design

For code organization, the BuildEML.jl package is structured as a top-level module. Within the src directory (Figure 2), its functionality is divided into several submodules, each implemented through Julia source files and organized into category folders when needed. The submodules are connected through using and re-export relationships. The `examples` directory then assembles, instantiates, and numerically solves these models in specific example cases.

(1) The Utils submodule provides supporting utilities, including interpolation, smoothing, unit conversion, and helper functions for converting between date-based settings and simulation time in seconds.

(2) The Media submodule defines the physical interfaces and simplified media-related functions. Air- and water-side connectors are defined separately, with moist-air psychrometric relations and basic water properties. This HVAC-oriented implementation is simpler than using general fluid ports with replaceable medium packages, and related discussion is also provided in Section 4.

- The moist-air connector `AirPort` carries four variables: pressure (Pa), outflow temperature (K), outflow humidity ratio (kg/kg), and dry-air mass flow rate (kg/s). Here, outflow temperature and humidity ratio are declared as *stream* variables, i.e., they

DOI 10.3384/ECP219 170

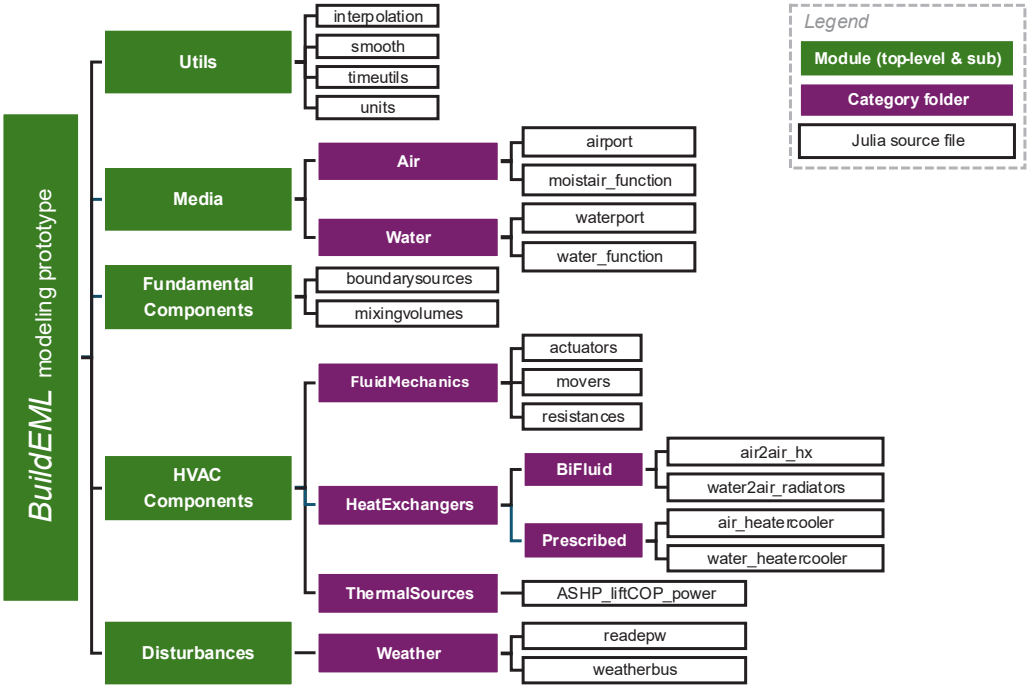


Figure 2. Module, category-folder, and source-file organization of the BuildEML.jl modeling prototype**.**

represent port outflow properties and are not directly equalized by connections. When a component requires the inflowing air properties at a port, it explicitly queries the connected upstream values via an instream function (used here as the prescribed mechanism to access inlet properties), while mass flow rate is declared as a *flow* variable with sign convention (> 0 means flowing into a component) and conservation is enforced upon connections. The stream and flow semantics follow the convention used by Modelica. The Modelica fluid connectors typically carry specific enthalpy and mass fractions as stream variables, which supports more general and replaceable fluid media. While temperature and humidity ratio are carried here, since they are the quantities normally used in HVAC calculations and are considered sufficient for the studied cases.

- The water-side connector `WaterPort` uses a reduced set (pressure, outflow temperature, and water mass flow rate) with the same stream/flow conventions.
- **(3) The FundamentalComponents submodule** contains boundary sources and mixing volumes. Mixing-volume models represent lumped fluid control volumes with mixing, storage, and the associated mass and energy balances. For example, `AirMixingVolumeNode2` is used for the zone air, while `WaterMixingVolumeNode2` is used in the water heater and radiator. In the current version, the mixing-volume models use a two-port formulation with predefined inlet and outlet roles.
- Boundary-source models provide prescribed boundary conditions for the fluid networks. On the air side, for example, separate source and sink models, such as `AirSource_mflow`, `AirSource_pT`, and `AirSink_p` are used, where the suffixes indicate the prescribed boundary quantities, such as mass flow rate or pressure and temperature. On the water side, `WaterBoundaryNode2_p` is used to provide the pressure boundary for the hydronic loop.

(4) The HVACComponents submodule contains higher-level applied model/system definitions for HVAC processes. It is noted that, for each equipment category, the current implementation covers a minimum set of model formulations, namely those required by the verification cases.

- The FluidMechanics folder contains component models for flow actuation and transport, including dampers for air, valves for water, fans, pumps, and flow resistances.
- The BiFluid HeatExchangers folder includes `Constant_HX` for air-to-air sensible heat recovery and radiator for water-to-air heat transfer, where the radiator model is a multi-element model with a customizable number of elements.
  The Prescribed HeatExchangers folder includes `SensibleCooler_T`, which prescribes the air outlet temperature for sensible cooling, and `WaterHeaterCooler_Q`, which imposes a prescribed heat input or extraction on the water side.
- The ThermalSources folder contains models that convert thermal load to electric power, currently represented by a constant-COP formulation and a lift-dependent air-source heat pump (ASHP) formulation.

DOI 10.3384/ECP219 170

(5) The Disturbances submodule includes weather boundary conditions and schedule-based internal heat gain inputs. For the weather data, EPW files (EnergyPlus weather file format (Crawley et al., 2001)) serve as input. `ReadEPW` reads and preprocesses the file into a standardized DataFrame with time, thermal, radiation, and wind variables. `WeatherBus` then constructs an ODESystem that uses a `ContinuousClock` and interpolation blocks to convert the weather data into continuous time-dependent signals.

Several basic model/system definitions are adopted directly from ModelingToolkitStandardLibrary.jl and used in the prototype, including thermal-network components (e.g., `HeatCapacitor` and `ThermalResistor`), prescribed boundary and source models (e.g., `PrescribedTemperature` and `PrescribedHeatFlow`), control blocks (e.g., `LimPID`), and clock/signal blocks (e.g., `ContinuousClock`, `RealInput`, and `RealOutput`).

2.3 Verification cases

This study uses the SimpleHouse example in MBL v11.0.0, as the reference model (Figure 3). The SimpleHouse represents a compact but fairly complete building energy system (Modelica Buildings Library, 2024). It includes weather, building, heating, and cooling/ventilation blocks, with both air and water loops.

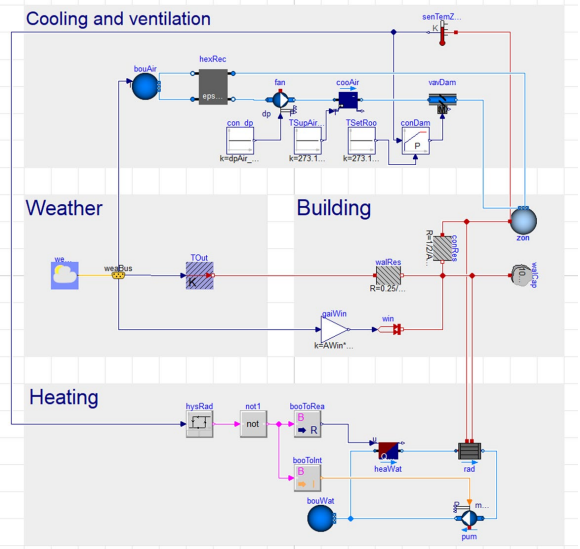


Figure 3. The graphical representation of the `Buildings.Examples.SimpleHouse` model in MBL.

There are generally two model paths for the SimpleHouse examples in the MBL: `Buildings.Examples.Tutorial.SimpleHouse` consists of seven progressively extended variants (SimpleHouse0–6), each building upon the previous case by adding components; Buildings.Examples.SimpleHouse is a standalone comprehensive model comparable in scope to SimpleHouse6, but with some changes. Table 1 summarizes the four cases used for verification.

Table 1. Summary of the four MBL SimpleHouse cases used for verification. Case 's' denotes the standalone SimpleHouse model.

Case	Key feature	Outputs shown in the tutorial
0	Weather block.	Outdoor temperature.
3	Weather block, RC envelope with a window, and zone air volume coupled through an interior convective resistance.	Zone air temperature.
5	Case 3 with a hot-water heating loop (water heater, pump, and radiator) under hysteresis control (zone 21–23 °C).	Zone air temperature.
s	Integrated building and HVAC system with hysteresis-controlled heating (zone 20–22 °C) and PID-controlled cooling and ventilation.	Zone air temperature.

The weather block transforms TMY weather into model inputs. The building block represents an RC zone model that captures indoor thermal dynamics through the zone air, envelope heat transfer, and solar gains. The heating block consists of a hydronic loop with a heater, pump, radiator, and simple on/off control logic. The cooling and ventilation block includes outdoor air supply, heat recovery, sensible cooling, fan, damper, and Proportional-Integral-Derivative (PID) control logic.

Because these four blocks capture the core interactions among weather, building thermal response, and HVAC operation, including both air-side and water-side loops, SimpleHouse was chosen as the verification case for a lightweight Julia prototype covering the essential building energy dynamics. The implementation in this work preserves the same overall structure and qualitative behavior at the component level. Specific deviations are further discussed in Section 4.

The verification compares simulation results from four representative SimpleHouse variants in MBL with the corresponding models assembled using the proposed prototype in Julia. The four cases are SimpleHouse0, SimpleHouse3, SimpleHouse5, and the standalone SimpleHouse model. More specifically, SimpleHouse0 is used to verify the weather block, particularly the transformation of weather-file data into time-dependent input signals through interpolation. SimpleHouse3 extends the verification to the coupled indoor thermal dynamics driven by weather, opaque-wall heat transfer, and window gains. SimpleHouse5 is selected to assess the coupled behavior of the house model and the hydronic

DOI 10.3384/ECP219 170

heating loop under hysteresis-based heating control. Finally, the standalone SimpleHouse model is used to examine the integrated building-HVAC system, including ventilation and cooling under PID control.

The model parameters are kept consistent with the SimpleHouse settings in the MBL (Table 2). All cases use the Chicago O'Hare International Airport TMY3 weather file. It is noted that humidity or latent control and internal heat gains are not considered in the SimpleHouse examples, although both the MBL reference and the prototype developed in this work retain the underlying capability.

Simulations are performed using the DASSL solver for MBL reference models in Dymola 2023 (Brück et al., 2002), and Rodas5P (Steinebach, 2023) for models in this Julia prototype, with a tolerance of 1e-6 and an output interval of 900 seconds. In the MBL reference, the default simulation horizon is 1e6 seconds. In this study, cases are extended to an annual horizon (31,536,000 seconds) to provide a more complete comparison. For SimpleHouse5 and the standalone SimpleHouse (cases with heating hysteresis control), the individual switching cycles are additionally shown over a 12-hour window inside the heating season.

Table 2. Summary of representative physical settings for the selected SimpleHouse verification cases, excluding inherited settings. Case 's' denotes the standalone SimpleHouse model.

Case	Physical setting	Value
3	Opaque wall area	100 m²
	Wall thickness	0.25 m
	Wall thermal conductivity	0.04 W/(m·K)
	Wall density	2,000 kg/m³
	Wall specific heat capacity	1,000 J/(kg·K)
	Window area	2 m²
	Zone air volume	192 m³
	Interior convective heat transfer coefficient	2 W/(m²·K)
	Small infiltration mass flow rate	1e-5 kg/s
5	Nominal heating capacity	3,000 W
	Nominal water mass flow rate	0.1 kg/s
	Radiator inlet/outlet water temperature	60/40 °C
	Nominal room air temperature for radiator	20 °C
	Number of radiator elements	5
	Heating hysteresis band	21-23 °C
s	Opaque wall area	100 m²
	Window area	5 m²
	Solar heat gain coefficient	0.3
	Zone air volume	300 m³
	Nominal heating capacity	700 W
	Nominal water mass flow rate	0.0167 kg/s
	Heating hysteresis band	20-22 °C
	Nominal air mass flow rate	0.2 kg/s
	Nominal air pressure difference	200 Pa
	Heat recovery sensible effectiveness	0.85
	Cooling supply-air temperature setpoint	20 °C
	Zone temperature setpoint for cooling control	24 °C
	Minimum damper opening	0.25

Pointwise singed error or signed relative error, root mean squared error (RMSE), and mean bias error (MBE) are computed between the two sets of results.

3 Results

SimpleHouse0 focuses on the weather block and uses Akima spline interpolation, consistent with the MBL reference case. As shown in Figure 4, the RMSE between the two signals over the annual simulation stays below 1e-5 °C for the outdoor dry-bulb temperature and below 1e-5 W/m² for the global horizontal irradiance. Signed relative error is not used for this case since the outdoor air temperature crosses 0 °C, where relative error becomes ill-conditioned. The results confirm the consistency of the EPW reading, time alignment, and interpolation.

DOI 10.3384/ECP219 170

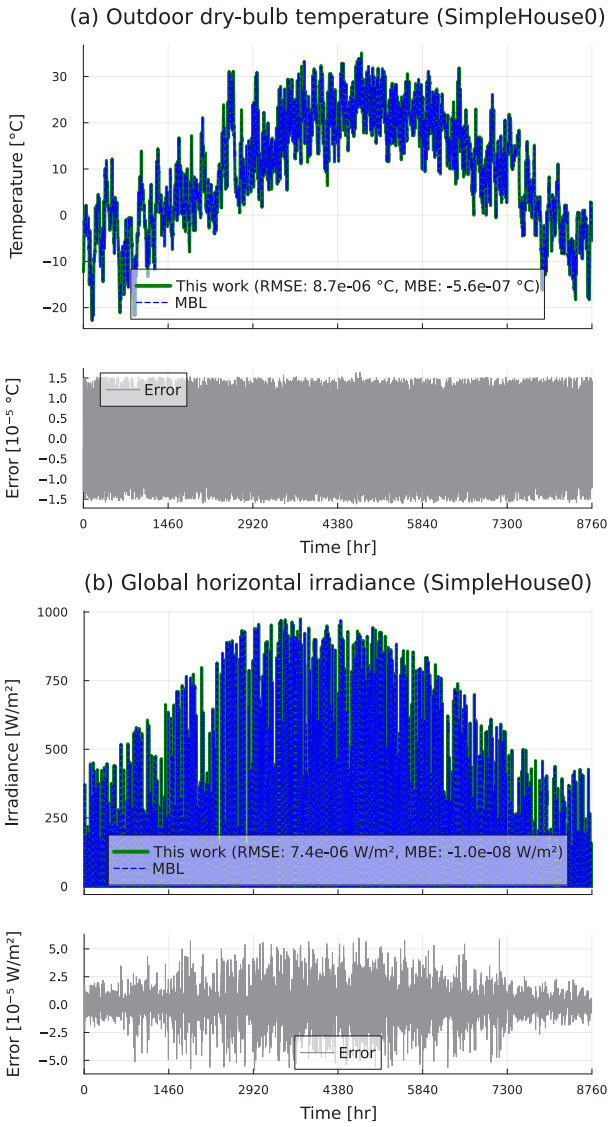


Figure 4. SimpleHouse0 verification results for outdoor dry-bulb temperature (a) and global horizontal irradiance (b).

SimpleHouse3 represents a free-floating building case in which the opaque envelope, window solar gains, and zone air volume are coupled through conductive and convective heat transfer driven by the weather boundary conditions. The resulting zone air temperature reflects the combined effects of weather boundary conditions, opaque-wall heat transfer, window gains, and zone-air thermal dynamics. As shown in Figure 5, the annual simulation of zone air temperature has an RMSE/MBE of 0.01 °C/-0.01 °C, so the envelope and zone-air formulation in the prototype produces consistent thermal response compared with the reference.

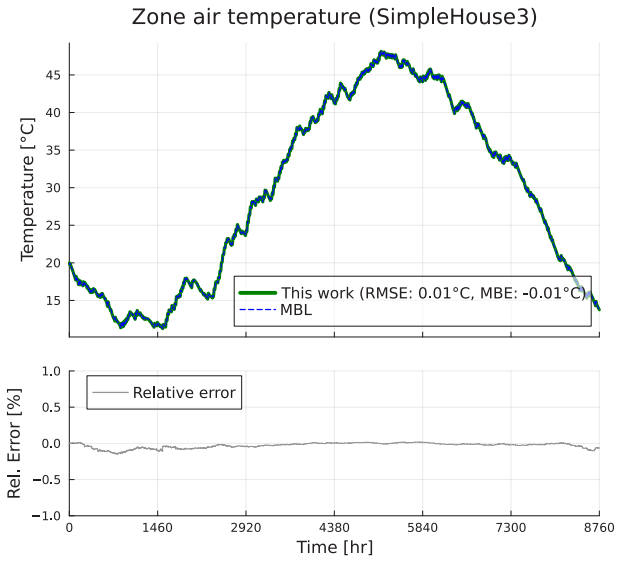


Figure 5. SimpleHouse3 verification results for zone temperature.

SimpleHouse5 adds a hydronic heating loop with hysteresis-based heating control over a 21–23 °C temperature band. The heating loop includes a 3,000 W water heater, a fixed-flow pump, and a five-element radiator coupled to the zone air and wall thermal mass.

As shown in Figure 6., both models show the same cycling behavior of the zone air temperature, with an annual RMSE/MBE of 0.54 °C/-0.01 °C between the two results. The RMSE is mainly attributed to phase differences between the hysteresis cycles. Here, the Julia model uses a smooth continuous approximation of the switching logic, while Modelica resolves discrete switching events. Since the control band spans 2 °C, an RMSE below 1 °C is consistent with a phase-shifted but similarly bounded response. The moving-average error, computed by averaging the pointwise temperature errors within a centered ±25-sample window, remains close to zero with slight systematic bias. Figure 6b resolves the individual switching cycles within a 12-hour window in the heating season. Both models cycle inside the 21–23 °C band with a similar period, and the cycles are offset in time, so the pointwise error alternates in sign while the response remains bounded by the band in both cases. During non-heating periods, the heating loop is inactive, so the zone temperature response is similar to the free-floating behavior in SimpleHouse3.

DOI 10.3384/ECP219 170

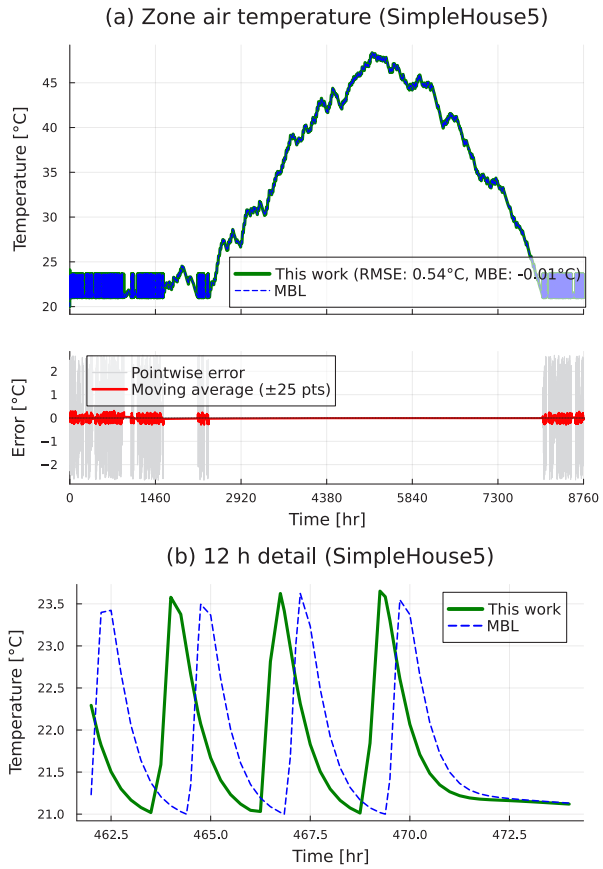


Figure 6. SimpleHouse5 verification results for zone temperature over the annual horizon (a) and a 12-hour heating window (b).

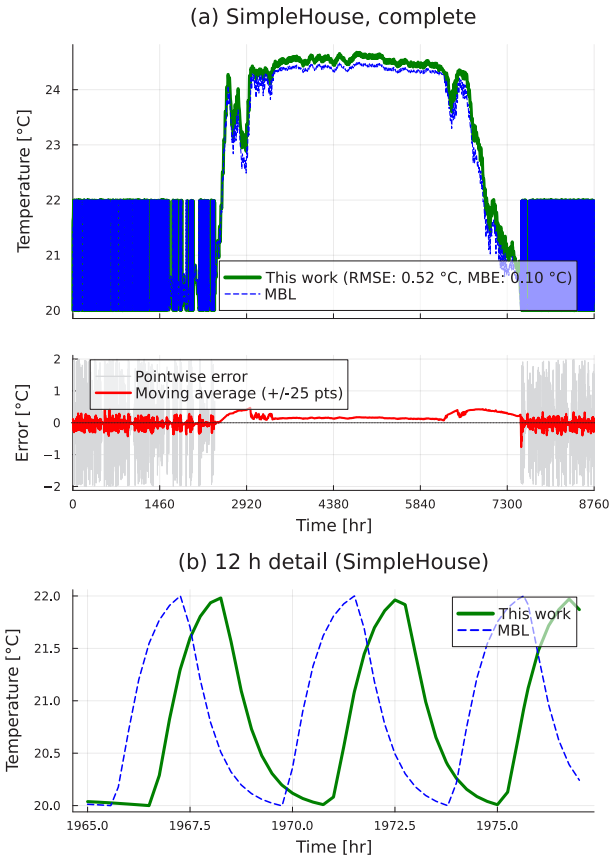


Figure 7. Standalone SimpleHouse verification results for zone temperature over the annual horizon (a) and a 12-hour heating window (b).

The standalone SimpleHouse case represents full operation, including the hysteresis-based heating loop as in SimpleHouse5 and an added ventilation and cooling loop. Unlike the tutorial SimpleHouse0–6 series, this case uses a different set of nominal parameters, including a 20–22 °C heating hysteresis band, a 700 W hydronic heating loop, a 100 m² wall area, and a 300 m³ zone volume. The cooling loop uses a sensible cooler with a fixed 20 °C supply-air temperature, while a PID controller tracks the 24 °C zone temperature setpoint by modulating the damper opening. The fan maintains a fixed pressure setpoint, and the resulting airflow through the cooling branch is therefore governed by the combined fan pressure and damper position.

As shown in Figure 7a, the annual RMSE/MBE between the two results is 0.52 °C/0.10 °C. Figure 7b resolves the switching cycles of the heating loop over a 12-hour window. The heating periods mainly show oscillatory errors as in SimpleHouse5, while the cooling-dominated period shows a small positive bias, i.e., slightly warmer than the reference. This suggests that the remaining discrepancy is mainly associated with the simplified air-side cooling loop and PID-controlled cooling implementation.

4 Limitations and future work

This work is prototype-oriented, and the identified limitations suggest several directions for future development.

The simplified air/water connector design used in this prototype is considered adequate for the present HVAC-focused cases, but it does not yet offer the generality of the replaceable-medium formulation used in MBL. Extending this layer to support more general media and connector definitions is therefore an important next step.

The boundary and mixing-volume models are also built around predefined flow configurations. Air-side boundaries are represented through separate source and sink models, while the water loop relies on a pressure-boundary formulation. The current mixing-volume and flow-component models primarily fixed inlet and outlet roles. The instream function is employed to access upstream stream variables, such as outflow temperature and outflow humidity ratio, because stream variables are not directly propagated through port connections, and reverse-flow is not supported at present. Future work could extend this layer to support Modelica-

DOI 10.3384/ECP219 170

style actualStream behavior and to handle reverse-flow formulations.

Some component formulations in the current implementation may require more robust regularization and initialization handling. In the present water heating model, the internal mixing volume uses a steady-state energy balance. Under zero-flow conditions, this balance becomes degenerate when both the water mass flow rate and heat input approach zero. A small residual pump flow of 1e-4 times the nominal flow rate is therefore retained when heating is disabled. A more systematic treatment of zero-flow behavior is left for future work.

Hybrid control is another limitation of the current prototype. MBL SimpleHouse instantiates the `Hysteresis` block of the Modelica Standard Library, which holds the switching state in a discrete Boolean variable. In this prototype the controller is implemented directly, and the switching state is held in a continuous variable governed by a first-order approximation of the switching logic (as in Appendix A). Relations in Modelica equations generate state events unless they are declared explicitly (e.g., `noEvent`) (Modelica Association, 2023), while in MTK such expressions are evaluated continuously under the default compilation and events are declared explicitly. MTK supports event-handling mechanisms, while providing a hysteresis or discrete control component for building control applications requires further investigation.

Additionally, the current workflow is code-centric and follows the MTK v10 interfaces used during development. The MTK is still actively developing, with newer versions such as v11 introducing additional features and interface changes that create both opportunities and a need for version aware prototype development. Graphical modeling support and other user-facing tools are also beyond the scope of this work.

The assembled model is available as an `ODESystem` and the corresponding `ODEProblem`, which could be the interfaces used by the optimization and machine learning packages of the Julia ecosystem. Control-oriented workflows built on these interfaces, together with compiler and solver performance benchmarking, are directions for future work.

5 Conclusions

This paper presents a lightweight building energy system modeling prototype to explore equation-based modeling and simulation in Julia. This prototype is structured into modular weather, thermal, and HVAC component models and is verified against four MBL SimpleHouse variants representing increasing system complexity. The verification results demonstrate that the implementation is numerically sound across all four reference cases. Weather data follows numerical precision, while annual zone-temperature RMSE/MBE values are 0.01/-0.01 °C for SimpleHouse3 with coupled envelope and zone-air dynamics; 0.54/-0.01 °C for the heating-control case in SimpleHouse5, and 0.52/0.10 °C for the standalone SimpleHouse case with integrated heating, cooling, and ventilation operation. These results indicate that the prototype accurately captures key building energy dynamics while providing an initial building-domain example of equation-based modeling in Julia. Future work will focus on enhancing the model completeness and robustness, expanding validation efforts, and investigating applications in control-oriented and optimization-based work, thereby supporting the continued development of building energy system modeling within the Julia SciML ecosystem.

Acknowledgements

This research was partially supported by the National Science Foundation under Grant No. 2318720.

Appendix A

This appendix summarizes the main media-related closure relations, constant parameters, and component-level governing equations used in the prototype.

The moist-air specific enthalpy h (J/kg$_{da}$) is calculated as:

$$h(T, w) = c_{p,\mathrm{da}}(T - T_0) + w\, h_g(T) \quad (1)$$

$$h_g(T) = h_{g,0} + c_{p,v}(T - T_0) \quad (2)$$

where T is the temperature (K); w is the humidity ratio (kg$_w$/kg$_{da}$); $c_{p,\mathrm{da}}$ =1005.45 J/(kg$_{da}$·K) is the dry-air specific heat capacity; $c_{p,v}$=1860 J/(kg$_w$·K) is the water-vapor specific heat capacity; h_g is the water-vapor specific enthalpy; $h_{g,0}$ =2501 kJ/kg$_w$ is the reference water-vapor specific enthalpy at T_0; and T_0 =273.15 K is the reference temperature.

The moist-air specific heat capacity $c_{p,\mathrm{ma}}$ (J/(kg$_{da}$·K)) is calculated as:

$$c_{p,\mathrm{ma}}(w) = c_{p,\mathrm{da}} + c_{p,v}\, w \quad (3)$$

The pressure-dependent dry-air density ρ (kg$_{da}$/m^3) is calculated as:

$$\rho(p) = \rho_{air,ref}\, \frac{p}{p_{air,ref}} \quad (4)$$

where p is the air pressure (Pa); $\rho_{air,ref}$=1.2 kg$_{da}$/m^3 is the reference dry-air density, and $p_{air,ref}$=101325 Pa is the reference air pressure.

The governing equations of the air mixing volume use the pressure-dependent density approximation and allow transient dry-air mass storage within the volume:

$$V\frac{d\rho}{dt} = \dot{m}_1 + \dot{m}_2 \quad (5)$$

DOI 10.3384/ECP219 170

$$\rho V \frac{dw}{dt} = \dot{m}_1(w_1 - w) + \dot{m}_2(w_2 - w) + \dot{m}_{w,add} \quad (6)$$

$$\rho V[c_{p,ma}(w)\frac{dT}{dt} + h_g(T)\frac{dw}{dt}] = \dot{m}_1(h_1 - h) + \dot{m}_2(h_2 - h) + \dot{Q} \quad (7)$$

where V is the control-volume size (m^3); $\dot{m}_1$ and $\dot{m}_2$ are the inlet and outlet mass flow rates (kg_{da}/s); w_1, w_2, h_1, and h_2 are the corresponding stream humidity ratios and specific enthalpies (kg/kg_{da}, kg/kg_{da}, J/kg_{da}, J/kg_{da}, respectively); $\dot{m}_{w,\mathrm{add}}$ is an added water mass flow rate (kg/s) representing a moisture source term, and $\dot{Q}$ is the net heat flow input (W), including energy associated with the added moisture when applied.

The governing equations of the water mixing volume, based on a simplified incompressible constant-density assumption for the water side, are:

$$\dot{m}_1 + \dot{m}_2 = 0 \quad (8)$$

$$\rho_w V c_{p,w} \frac{dT}{dt} = \dot{m}_1 c_{p,w} T_1 + \dot{m}_2 c_{p,w} T_2 + \dot{Q} \quad (9)$$

where ρ_w=990 kg/m^3 and $c_{p,w}$=4186 J/(kg·K) are the water density and specific heat capacity.

The continuous approximation of the hysteresis control is:

$$\tau \frac{dy}{dt} = \begin{cases} 1 - y, & T_{\mathrm{room}} \le T_{\mathrm{low}} \\ 0, & T_{\mathrm{low}} < T_{\mathrm{room}} < T_{\mathrm{high}} \\ -y, & T_{\mathrm{room}} \ge T_{\mathrm{high}} \end{cases}$$

where $y \in [0,1]$ is the continuous control output; τ=1s is the controller time constant used by the implementation; T_{low} and T_{high} are the lower and upper switching thresholds; and $T_{\mathrm{room}}$is the measured zone air temperature expressed in K.

References

Åkesson, J., Årzén, K.-E., Gäfvert, M., Bergdahl, T., & Tummescheit, H. (2010). Modeling and optimization with Optimica and JModelica.org—Languages and tools for solving large-scale dynamic optimization problems. *Computers & Chemical Engineering*, *34*(11), 1737–1749. https://doi.org/10.1016/j.compchemeng.2009.11.011

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, *59*(1), 65–98. https://doi.org/10.1137/141000671

Brück, D., Elmqvist, H., Mattsson, S. E., & Olsson, H. (2002). *Dymola for Multi-Engineering Modeling and Simulation*. 55.

Cellier, F. E., & Kofman, E. (Eds.). (2006). Differential Algebraic Equation Solvers. In *Continuous System Simulation* (pp. 319–396). Springer US. https://doi.org/10.1007/0-387-30260-3_8

Crawley, D. B., Lawrie, L. K., Winkelmann, F. C., Buhl, W. F., Huang, Y. J., Pedersen, C. O., Strand, R. K., Liesen, R. J., Fisher, D. E., Witte, M. J., & Glazer, J. (2001). EnergyPlus: Creating a new-generation building energy simulation program. *Energy and Buildings*, *33*(4), 319–331. https://doi.org/10.1016/S0378-7788(00)00114-6

Deshpande, V. M., & Laughman, C. R. (2023). Multi-pass Extended Kalman Smoother with Partially-known Constraints for Estimation of Vapor Compression Cycles. *IFAC-PapersOnLine*, *56*(2), 6751–6758. https://doi.org/10.1016/j.ifacol.2023.10.381

Elmqvist, H., Matsson, S. E., & Olsson, H. (2014). *Parallel Model Execution on Many Cores*. https://ep.liu.se/en/conference-article.aspx?series=ecp&issue=96&Article_No=38

Hofmann, A., & Mikelsons, L. (2025). Towards Integration of PeN-ODEs in a Modelica-based workflow. *Proceedings of the 16th International Modelica&FMI Conference*, 435–444. https://doi.org/10.3384/ecp218435

Laughman, C., Deshpande, V., Qiao, H., Bortoff, S., & Chakrabarty, A. (2023, August 21). Digital twins for vapor compression cycles: Challenges & opportunities. *Roceedings of the 26th IIR International Congress of Refrigeration*. https://doi.org/10.18462/iir.icr.2023.0765

Ma, Y., Gowda, S., Anantharaman, R., Laughman, C., Shah, V., & Rackauckas, C. (2022). *ModelingToolkit: A Composable Graph Transformation System For Equation-Based Modeling* (arXiv:2103.05244). arXiv. https://doi.org/10.48550/arXiv.2103.05244

Modelica Association. (2023). *Modelica—A unified object-oriented language for systems modeling: Language specification version 3.6* (3.6) [Manual]. Modelica Association. https://specification.modelica.org/maint/3.6/MLS.html

Modelica Buildings Library. (2024). *Buildings.Examples.Tutorial.SimpleHouse*. https://simulationresearch.lbl.gov/modelica/releases/v11.0.0/help/Buildings_Examples_Tutorial_SimpleHouse.html

Qiu, K., Yang, J., Gao, Z., & Xu, F. (2024). A review of Modelica language in building and energy: Development, applications, and future prospect. *Energy and Buildings*, *308*, 113998. https://doi.org/10.1016/j.enbuild.2024.113998

Rackauckas, C., Gwozdz, M., Jain, A., Ma, Y., Martinuzzi, F., Rajput, U., Saba, E., Shah, V. B., Anantharaman, R., Edelman, A., Gowda, S., Pal, A., & Laughman, C. (2022). Composing Modeling And Simulation With Machine Learning In Julia. *2022 Annual Modeling and Simulation Conference (ANNSIM)*, 1–17. https://doi.org/10.23919/ANNSIM55834.2022.9859453

Rackauckas, C., & Nie, Q. (2017). DifferentialEquations.jl – A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia. *Journal of Open Research Software*, *5*(1). https://doi.org/10.5334/jors.151

Song, R., Liu, M., Yang, Z., & O'Neill, Z. (2025, August 4). Energy Performance Analysis of a Dedicated Outdoor

DOI **10.3384/ECP219 170**

Air System with a Desiccant Wheel through Julia-Based Modeling. *2025 ASHRAE Annual Conference*. 2025 ASHRAE Annual Conference. https://doi.org/10.63044/s25ene46

Steinebach, G. (2023). Construction of Rosenbrock–Wanner method Rodas5P and numerical benchmarks within the Julia Differential Equations package. *BIT Numerical Mathematics*, *63*(2), 27. https://doi.org/10.1007/s10543-023-00967-x

Thummerer, T., Kircher, J., & Mikelsons, L. (2021). NeuralFMU: Towards Structural Integration of FMUs into Neural Networks. *14th Modelica Conference 2021*, 297–306. https://doi.org/10.3384/ecp21181297

Wetter, M., Bonvini, M., & Nouidui, T. S. (2016). Equation-based languages – A new paradigm for building energy modeling, simulation and optimization. *Energy and Buildings*, *117*, 290–300. https://doi.org/10.1016/j.enbuild.2015.10.017

Wetter, M., & Sulzer, M. (2024). A call to action for building energy system modelling in the age of decarbonization. *Journal of Building Performance Simulation*, *17*(3), 383–393. https://doi.org/10.1080/19401493.2023.2285824

Wetter, M., Zuo, W., Nouidui, T. S., & Pang, X. (2014). Modelica Buildings library. *Journal of Building Performance Simulation*, *7*(4), 253–270. https://doi.org/10.1080/19401493.2013.765506

Willard, J., Jia, X., Xu, S., Steinbach, M., & Kumar, V. (2022). *Integrating Scientific Knowledge with Machine Learning for Engineering and Environmental Systems* (arXiv:2003.04919). arXiv. https://doi.org/10.48550/arXiv.2003.04919

DOI **10.3384/ECP219 170**

Implementation of OpenIPSL in Dyad

John Batteh[1] Luigi Vanfretti[2]

[1]JuliaHub, USA, john.batteh@juliahub.com

[2]Department of Electrical, Computer and Systems Engineering,
Rensselaer Polytechnic Institute, USA, vanfrl@rpi.edu

Abstract

This paper presents a translation of the Open Instance Power System Library (OpenIPSL) into Dyad, an acausal modeling language that compiles to ModelingToolkit.jl in Julia. Details are provided regarding the translation process leveraging agentic AI capabilities and validation approach. The benefits of having both a system simulation specific AI agent and the library running in multiple tools are covered. The resulting library OpenIPSLComponents is also extended with additional components to demonstrate the potential for applications in data center power systems. Initial simulations for a gas turbine trip scenario are presented.

Keywords: power systems, phasors, data centers

1 Introduction

Power system transient stability simulation has traditionally relied on domain-specific tools such as PSS/E® (Siemens PTI 2017) and PSAT (Milano 2005). The Open Instance Power System Library (OpenIPSL) brought these capabilities into the Modelica ecosystem, enabling equation-based, acausal modeling of power systems using general-purpose simulation environments (de Castro *et al.* 2023). OpenIPSL has been used in a wide range of power system applications including traditional power grids (de Castro Fernandes *et al.* 2019), microgrids (Fachini *et al.* 2023), hydroelectric systems (Winkler 2018), and controls (Vanfretti and Laughman 2024).

Given the successful application of OpenIPSL in Modelica tools, there is a strong motivation to bring its proven capabilities for power system simulation into the Dyad (JuliaHub 2026) ecosystem. Dyad is a new language designed for the simulation of multi-domain physical systems and controls. Like Modelica, Dyad is acausal, object-oriented, and equation-based. Dyad models are compiled into Julia code for simulation with ModelingToolkit.jl (Ma *et al.* 2021). Thus, models in Dyad can leverage automatic differentiation and a robust suite of transient and steady state solvers available in ModelingToolkit. In addition, Dyad models can interface directly with the Julia/SciML ecosystem to provide a unified platform for physical modeling combined with physics-informed AI including neural networks, reduced order models/surrogates, and automated model generation. Julia also provides a native deployment platform for web applications and digital twins. As a language, Dyad is designed to work together with AI and includes built-in agentic capabilities. While the focus of this paper is not on the Dyad language or its underlying software architecture, relevant elements of the Dyad implementation and agentic AI capabilities specifically related to the translation and extension of the OpenIPSL library are discussed. For more details on the Dyad language and syntax, the interested reader is referred to the online Dyad documentation (JuliaHub 2026).

The remainder of this paper is organized as follows. Section 2 describes the model translation process, including the role of the Dyad AI agent and the scope of the translated library. Section 3 presents the validation process for several key OpenIPSL models. It also addresses unique considerations arising between the Dyad and Modelica languages, along with key benefits for the Modelica community. Section 4 introduces new models implemented in Dyad and presents initial power system simulation results for a data center application. Section 5 concludes and outlines considerations for future work.

2 Model Translation

This section documents the process to translate OpenIPSL from Modelica into Dyad.

2.1 Process Overview

OpenIPSL is an extensive library comprising hundreds of component models, subsystems, and pre-configured test systems across its core packages. Figure 1 shows the Modelica package structure of OpenIPSL 3.1 along with an expanded view of the package structure in the Electrical package. The goal of this initial effort is not to translate the entire library into Dyad but to provide a Proof of Concept (POC) by focusing on some initial critical components and examples to support a specific application.

Given the extent of the library, it is not practical to translate the code manually into Dyad. To support an automated process for the model translation, the Dyad AI agent was used to translate the model. The Dyad AI agent is a custom agent for system simulation provided along with Dyad. The agent includes appropriate resources and skills for the Dyad language along with supporting technologies like ModelingToolkit and Julia. The agent also utilizes a custom harness developed specifically for system modeling and model-based engineering workflows

DOI 10.3384/ECP219 180

in Dyad. It can be used to support an array of typical system modeling and simulation processes including model creation, model and code manipulation, execution, post processing, analyses, and even debugging. The agent's resources also include custom modeling and debugging skills to ensure that models can be compiled and are tested as part of any model creation process. The Dyad agent provides access to several different Large Language Model (LLM) backends based on evaluation of the LLMs on a suite of gradable test problems to evaluate capability, accuracy, and token efficiency. Given the underlying LLMs existing capability to understand Modelica code and the specific skills provided around Dyad for model creation and debugging, the agent can be leveraged to support the translation process.

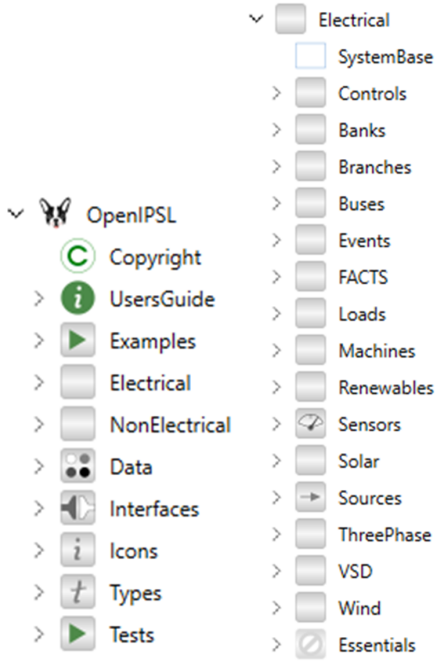


Figure 1. OpenIPSL package structure

Prior to the translation, a few guiding principles were established:

- The goal of the translation process is a faithful reproduction of OpenIPSL not just in the model code but also in terms of model organization, graphics, and modeling paradigms (*i.e.* graphical approach vs. code)
- There can be implementation differences between the Modelica and Dyad versions due to differences in the syntax and features of the languages, but these differences should not result in different modeled physics or numerical results
- Wherever possible, features of the Dyad language will be utilized in implementations given the requirement to maintain the same resulting model behavior
- Models that do include differences in the underlying implementation will be given a different name to distinguish them from the original OpenIPSL component
- Translated models must include at least one test case for use in verification and validation of resulting components

2.2 Translation Using the Dyad AI Agent

To create the Dyad implementation of OpenIPSL, the Dyad AI agent with Opus 4.6 was leveraged to perform the translation work. The translation process was performed step by step to allow for user review prior to proceeding. The first step is to create the connector. Figure 2 shows the resulting equivalent connector implementation in Dyad with the real and imaginary components of the voltage and current phasors. The `potential` and `flow` keywords are used to indicate the type of variable on the connector. The graphical metadata for the component icon SVG and the default color for connections is also shown.

```
"Connector for quasi-static single-phase electrical systems using phasor representation."
connector Pin
  "Real part of voltage phasor"
  potential vr::Voltage
  "Imaginary part of voltage phasor"
  potential vi::Voltage
  "Real part of current phasor"
  flow ir::Current
  "Imaginary part of current phasor"
  flow ii::Current
metadata {
  "Dyad": {
    "labels": [
      { "label": "instance_name", "x": 0, "y": 750, "layer": "diagram" }
    ],
    "icons": {
      "default": "dyad://OpenIPSLComponents/Pin.svg"
    },
    "path": { "stroke": "#0000ff" }
  }
}
end
```

Figure 2. Connector implementation in Dyad

With the connector defined, it is possible to start translating components. The process to translate a specific component is as follows:

1. Provide the Modelica code in the project workspace
2. Prompt the agent by pointing to the Modelica code and any specific tests that should be used for validation
3. Review any proposed formulations with the agent and iterate as needed on the formulation
4. Review the generated code and test case results
5. Iterate as needed
6. Prompt the agent to create a report document the model creation and validation process and results

A sample agent prompt is shown in Figure 3. OpenIPSL includes not only the component models but also the test suite used to validate individual models. While the Dyad AI agent will create test cases automatically as part of its model generation protocol, here the agent is simply

DOI 10.3384/ECP219 180

pointed to the existing test case in OpenIPSL so that it creates not only the component but also the same exact test used to validate the component's behavior. The resulting test models are shown in Figure 4. The models are equivalent though the Dyad Standard Library does not currently have a vectorized sum component, so the agent used a series of scalar sums to provide equivalent functionality.

```
Review OpenIPSL.Electrical.Loads.PSAT.PQ in the resources folder.  Propose an implementation in Dyad.
Validate using OpenIPSL.Tests.Loads.PSAT.LoadTestPQ.
```

Figure 3. Sample agent prompt for component translation

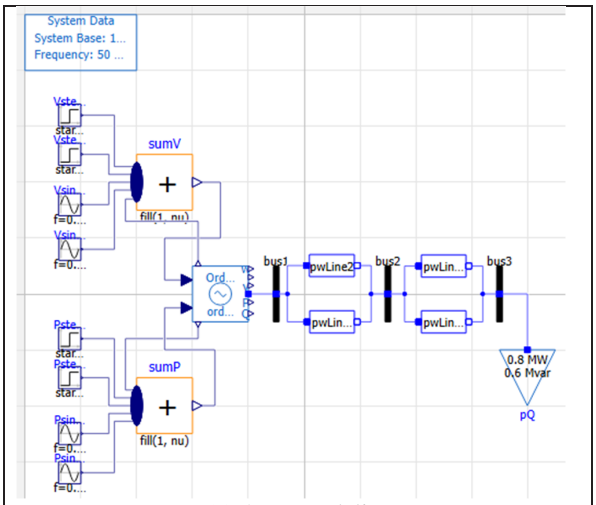


Figure 4. LoadTestPQ in OpenModelica and Dyad

2.3 Translation Scope

Given the extent of OpenIPSL, the initial translation focus for the POC was to include a range of components and examples that can provide the foundation for an initial power system application with gas turbine (GT) generators. Table 1 shows a summary of the models translated to date. The partial translation into Dyad currently covers 70 components across 23 packages. The translation spans the core power system modeling stack: synchronous machines in both PSAT (Order3/4/6) and PSS/E (GENCLS, GENROU, GENROE) conventions, 11 load models with constant-power, voltage-dependent, frequency-dependent, and sheddable variants, excitation systems (IEEET1, ESST1A, ST5B) and GT-governor models (GAST, IEEEG1), transmission lines and transformers, protection devices, and the supporting control blocks (lags, lead-lags, limiters, gates) used to assemble higher-order controllers. Two multi-machine examples from the OpenIPSL tutorial — a 3-machine 9-bus system and a single-machine infinite-bus (SMIB) system with GT-governor and Automatic Voltage Regulator (AVR) control — have also been translated and validated against OpenModelica simulation results.

Table 1. Translated models

OpenIPSL Package	Components
Interfaces	Generator
Electrical.Essentials	pfComponent
Electrical.Buses	Bus, InfiniteBus
Electrical.Branches	PwLine
Electrical.Branches.PSAT	TwoWindingTransformer
Electrical.Branches.PSSE	TwoWindingTransformer
Electrical.Machines.PSAT	BaseMachine, Order3, Order4, Order6
Electrical.Machines.PSSE	BaseMachine, GENCLS, GENROU, GENROE
Electrical.Loads.PSAT	BaseLoad, PQ, PQvar VoltageDependent, FrequencyDependent
Electrical.Loads.PSSE	BaseLoad, Load, LoadExtInput LoadFreqDepSheddable, LoadSwitch, LoadVariation
Electrical.Controls.PSAT	AVR Type II, AVR Type III, PSS Type II
Electrical.Controls.PSSE.ES	BaseExciter, RotatingExciter, ESST1A IEEET1, ST5B
Electrical.Controls.PSSE.TG	GAST, IEEEG1
Electrical.Banks.PSSE	Shunt, CSVGN1
Electrical.Events	Breaker, BreakerSignalReal, PwFault FrequencyRelay, ProtectionBreaker, ProtectionRelay
Electrical.Sensors	PwCurrent, PwVoltage
Electrical.Sources	VoltageSourceRelInput
NonElectrical.Continuous	SimpleLag, SimpleLagLim, SimpleLagLimVar LeadLag, LeadLagLim, DerivativeLag LimIntegrator, IntegratorNoInit
NonElectrical.Functions	SE (saturation)
NonElectrical.Logical	HVGate, LVGate, SelectLogic3, SwitchReal
NonElectrical.Nonlinear	CeilingBlock, FrequencyCalc
Examples.Tutorial.Example_3	Example_3, Gen1, Gen2, Gen3
Examples.Tutorial.Example_4	SMIBPartial, GeneratorTurbGovAVR InfiniteBus, SMIB

DOI 10.3384/ECP219 180

3 Selected Validation Results

This section provides validation results for a few selected translated models.

3.1 LoadTestPQ

The *LoadTestPQ* test case shown in Figure 4 consists of a three-bus network with a 3rd-order synchronous machine (PSAT Order3) at Bus 1, parallel transmission lines connecting Buses 1–2–3, and a constant PQ load at Bus 3. Small-signal perturbations are applied to the machine's field voltage and mechanical power inputs via step and sinusoidal signals. The system has no infinite bus, so the rotor angle is free to drift.

The model was simulated in both OpenModelica and in Dyad. The OpenModelica simulation used the DASSL solver at its default tolerance of 10^{-6}. The Dyad simulation also used its default solver, an auto-switching algorithm that selects among explicit (Tsit5, Vern7) and implicit (Rosenbrock23, Rodas5P, FBDF) methods based on problem stiffness with abstol = reltol = 10^{-6} to try and approximate the same tolerance as the Modelica simulation. Output from both solutions were obtained at identical output time points to eliminate interpolation artifacts. Figure 5 shows the comparison across six key variables. The rotor angle and machine speed traces are visually indistinguishable, with maximum errors of 4.2×10^{-4} rad and 1.3×10^{-6} pu respectively. The transient EMF (e'q) and bus voltage magnitudes agree to within 1.8×10^{-5} pu — differences that appear as visible separation only because the plot axes span less than 0.001 pu. Machine active power agrees to 7.6×10^{-8} pu, and the PQ load outputs are identical. All errors are at or below the solver tolerance floor.

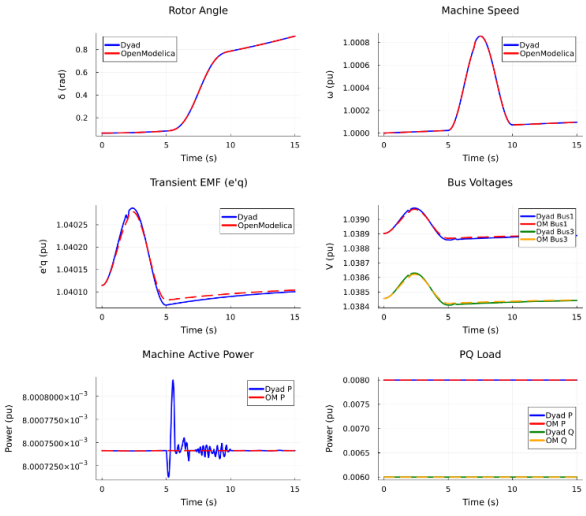


Figure 5. Validation results for LoadTestPQ

3.2 Example_3

Example_3 in OpenIPSL shown in Figure 6 is a 9-bus 3-generator test system with a three-phase-to-ground fault at bus 9. The fault occurs at 3s and ends at 3.1s. This model exercises a broader set of translated components than the *LoadTestPQ* case including the PSAT Order4 synchronous machines with AVR Type II excitation systems, PSAT two-winding transformers, voltage-dependent loads, and a three-phase fault. With no turbine-governor model and zero mechanical damping (D = 0), the system is near the marginal stability limit (two complex eigenvalues with less than 2% damping) and highly sensitive to both model fidelity and solver accuracy.

The fault implementation *PwFault* in Modelica and in Dyad are equivalent and use `if` statements for the switching admittance. The Dyad/ModelingToolkit solver framework does not currently provide equivalent automatic event handling for conditional expressions in component equations as in Modelica. To work around this issue, the simulation was split into three segments: pre-fault (t = 0–3.0 s), during-fault (t = 3.0–3.1 s), and post-fault (t = 3.1–20.0 s). At each segment boundary (fault onset and clearance), a Newton solve determined algebraic state values consistent with the new network topology while preserving all differential states (rotor angles, speeds, flux linkages, and AVR internal states) exactly from the end of the preceding segment thus replicating the essential characteristics of the Modelica event handling. The Dyad AI agent suggested and implemented this simulation approach. Figure 7 shows the comparison between Dyad and OpenModelica for rotor angles and bus voltages over 20 s with the Dyad absolute and relative tolerance = 10^{-6}, matching OpenModelica's DASSL default. All six traces (three generators, three buses) match closely through the pre-fault oscillations, the fault-induced voltage depression, and the post-fault recovery and deceleration. It should be noted that automatic event handling in Dyad is not only planned but also implemented and being tested for future release.

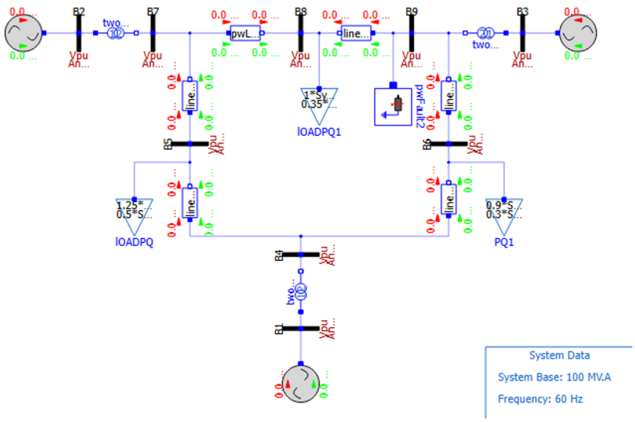


Figure 6. Example_3: 9-bus 3-generator system in Modelica

DOI 10.3384/ECP219 180

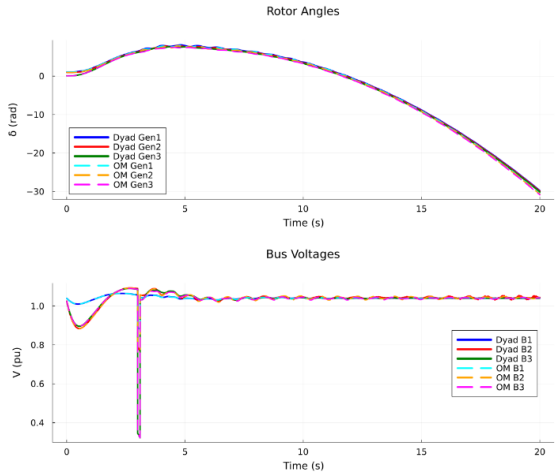


Figure 7. Validation results for Example_3

3.3 Evaluating AI-Assisted Translation

Because the translation process was executed with a human in the loop over a series of sessions, there are no exact quantitative metrics that can be cited regarding the success rate, translation efficiency, *etc.* However, a qualitative evaluation of the process can be provided.

The Dyad AI agent proved extremely capable of translating OpenIPSL from Modelica into Dyad. Since the custom harness requires that a generated model both compiles and simulates, there were no times when the agent left a model in an unworkable state for human intervention. In addition, the agent is physically aware and understands acausal modeling. Thus, the agent automatically performs checks for conservation, reasonable values, etc. In addition, OpenIPSL also includes a substantial test suite, so the translation process included not just translating the components but also replicating the tests for numerical comparison between the tools. Certainly, the prompting approach can have a large impact on the overall translation experience. A standard approach to effectively plan with the agent and review proposed formulations prior to their generation as code was adopted. This approach is highly recommended as it allows corrections and iterations before code is generated, thereby decreasing the time and tokens required to execute the translation process (since the harness requires a more extensive evaluation loop when making changes to models already in code).

A few issues were encountered as part of the translation process. As noted in the subsequent section, there were times when models did not validate. However, the agent's debugging capabilities, including the ability to independently validate calculations in Julia, were extremely helpful to understand and even directly identify the issues in the code bases. There were certainly times when several iterations were required to converge on the desired translation approach, especially in cases where there may not have been language elements that matched identically between Modelica and Dyad. Because the same sorts of prompts were generated repeatedly for translating different models, it would have been a much more streamlined process to create a custom skill for the agent to apply repeatedly to each new model translation request. Not only would the skill cut down on the prompting but also would help to ensure consistency in approach from session to session. However, these issues are more about enhancements to the process rather than issues that did not allow the process to work in the first place. Overall, it is very easy to say that this translation and validation process would have taken far longer (and perhaps have not even been possible) without the use of the Dyad AI agent.

3.4 Additional Insights from Dyad AI Agent

As part of the validation effort, there were several times when the models did not validate well. The Dyad AI agent proved extremely valuable in identifying the underlying issues and helping to diagnose the problem and solutions. In particular, there were several issues identified in the original Modelica models based on analysis and debugging from the agent:

- In the SMIB test base class, the value for angle_0 was provided in degrees rather than radians in the Modelica code. The agent identified that the value converted to radians would satisfy the steady state power flow solution exactly and thus flagged it as a unit conversion error (and thus a bug) in the original model.
- In the PSS/E generators GENROE and GENROU, the agent identified that there were discrepancies between some of the equations and particularly their signs. The agent also identified that the values for the fluxes PSIq and PSIppq in GENROU have different signs than the initial values, but those initial values may be used as start values. This finding was a result of a change in sign that was not consistently applied in the original Modelica code.
- In the ST5B component, the agent identified that there were several blocks that were not connected and that some of the blocks were using the same parameterization but likely should not have been. These findings identified missing logic in the model and led to the conclusion that the Modelica model was not complete and had not been verified with the original source.

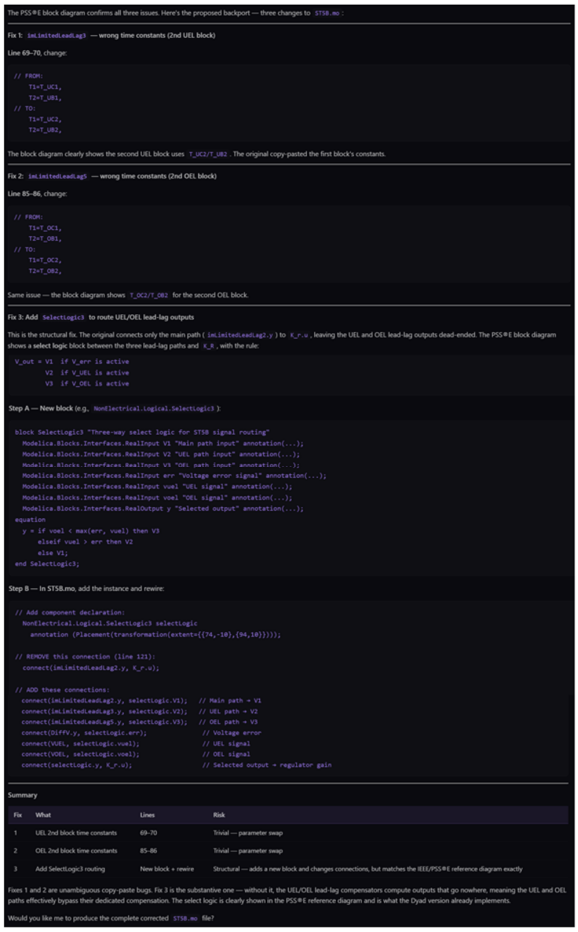
In the Modelica community, there has always been the premise that having Modelica code that is tested on multiple Modelica tools makes for a stronger, more robust code base and a healthier ecosystem as a whole. In porting OpenIPSL to Dyad, this same opportunity holds true in addition to allowing the testing of the code in even new ways (*i.e.* with additional solvers, numerical techniques, symbolic techniques, *etc.*). This sort of testing can lead

DOI **10.3384/ECP219 180**

not only to reporting of issues and bugs but even potential improvements or enhancements to the library that can improve the experience across tools and languages.

To demonstrate this cross-language workflow and generative AI capability for languages with a clear syntax, consider the use case to back-propagate the fixes for the ST5B component in Dyad to Modelica. A sample prompt is shown in Figure 8. The agent output and summary of the generated code after confirmation are shown in Figure 9. The agent created a new logical component and updated the ST5B component with the new component, fixes, and appropriate connections. It would certainly even be feasible for the agent to automatically create pull requests in the open-source GitHub repositories with these changes (with appropriate testing). With this workflow and powerful generative AI capability for both Dyad and Modelica, it is possible to ensure not only that code bases stay in sync but also that the Dyad and Modelica communities can work together seamlessly.

Figure 8. Agent prompt for back-propagation of ST5B changes to Modelica



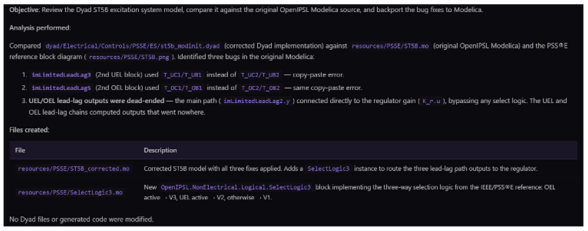


Figure 9. Agent output for back-propagation of ST5B changes to Modelica

4 Data Center Application

The motivation to translate OpenIPSL into Dyad is to support power system applications leveraging the Julia framework for model deployment and digital twins. One potential application for OpenIPSL is in the data center space. Large-scale data centers are among the fastest-growing categories of electrical load, with individual campuses now routinely exceeding 100 MW of demand. Many hyperscale and colocation facilities are sited in locations where utility grid capacity is limited or permitting timelines are long, leading operators to pursue islanded or semi-islanded power architectures served by on-site generation — typically natural gas turbines or reciprocating engines.

In an islanded configuration the generators must supply the full facility load without the frequency and voltage support of a large, interconnected grid. This configuration makes the system acutely sensitive to generation-loss contingencies: when a single unit trips, the surviving machines must absorb the entire load deficit, and the resulting frequency excursion can be significantly deeper than in a grid-connected plant. If the excursion exceeds the under-frequency load-shedding (UFLS) threshold, protection relays will disconnect a portion of the data center load — an outcome that directly impacts IT service availability. Understanding the interaction between generator dynamics, governor response, protection relay timing, and load composition is therefore critical for data center power system design and control.

This application was chosen for an initial application of the model in Dyad as it provides an opportunity to leverage the core power system modeling capabilities in OpenIPSL but also implement new models in Dyad.

4.1 New Components

The GT trip study requires several capabilities not available in the standard OpenIPSL library: protection relays with configurable timing and latching, signal-driven breakers, and composite load models with load-shedding inputs. New models were implemented in Dyad as extensions of OpenIPSL.

A generic *ProtectionRelay* component was implemented that provides pickup-threshold detection with configurable delay, integrator-based timing, and an irreversible ratchet latch that models the manual-reset behavior of real relays. A new *ProtectionBreaker*

DOI **10.3384/ECP219 180**

component wraps the relay with a mode-selectable monitor signal (reverse power, overcurrent, under/overvoltage). The breaker can be used to represent ANSI devices 27/32/50/51/59 which operate in various protection modes. In Dyad an enumeration is used in conjunction with the case/switch syntax as shown in Figure 10 to provide the appropriate monitor signal for the relay based on the mode. The relay drives a *BreakerSignalReal* component — an admittance-based breaker whose conductance transitions between closed and open states based on a real signal. A new *FrequencyRelay* component was implemented that uses the *FrequencyCalc* model in OpenIPSL which derives bus frequency from the Cartesian voltage phasor using a wrapping-free formulation, avoiding the 2-argument arctangent (atan2) discontinuities, and trips when the frequency deviation exceeds a pickup threshold for a sustained duration.

```
switch mode
  case ReversePower
    monitor = -(s.vr * s.ir + s.vi * s.ii)
  case OverCurrent
    monitor = sqrt(s.ir^2 + s.ii^2)
  case OverVoltage
    monitor = sqrt(s.vr^2 + s.vi^2)
  case UnderVoltage
    monitor = 2.0 * pickup - sqrt(s.vr^2 + s.vi^2)
end
```

Figure 10. Case switch logic in ProtectionBreaker

A composite *DataCenterLoad* was implemented in Dyad that aggregates the total facility power from an IT load and a cooling load. The IT load is modeled as constant power with a voltage cutoff (representing power supply unit shutdown) and no frequency dependence (representing uninterruptible power supply (UPS) isolation from grid frequency). The cooling load uses the full PSS/E ZIP (constant impedance, current, and power) voltage decomposition with low-voltage behavior and frequency dependence representing motor-dominated HVAC systems. Each sub-load has an independent shed input and configurable sheddable fraction, enabling tiered load shedding.

4.2 System Model

The data center model shown schematically in Figure 11 and in the Dyad model in Figure 12 comprises a generator group, a transmission network, a main step-down transformer, and a composite data center load. The generator group subcomponent includes 6 GTs each modeled using the GENROU generator, GAST GT-governor model, and the ST5B exciter model. The aggregated data center load discussed previously provides the IT load modeled as constant power and the cooling load as a frequency and voltage dependent load to represent variable-speed drives on chillers, pumps, and fans. Both loads have a sheddable fraction based on an input signal. Two protection functions are included: UFLS that shed the designated fractions of IT and cooling load and a reverse-power breaker (ANSI 32) that disconnects the tripped turbine when absorbing power.

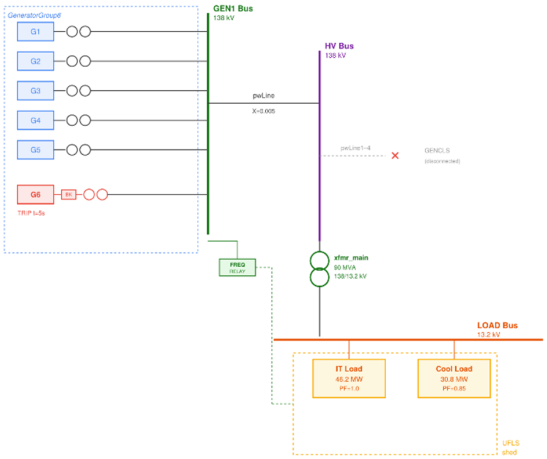


Figure 11. Turbine trip model schematic

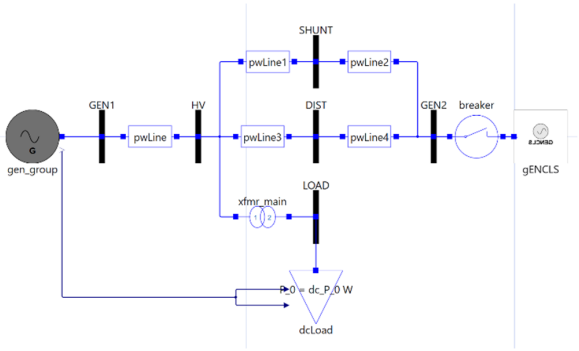


Figure 12. Dyad model

4.3 Results

Two scenarios are simulated in Dyad over a 15 s window to illustrate the effect of generator redundancy on system resilience. In the first case, all 6 generators are online and share the load equally. In the second case, the fifth generator is not used and thus the 5 remaining generators share the load equally. In both cases, generator 6 trips at 5 s.

Figure 13 shows the system response when G6 trips with all six generators initially online. Upon loss of G6's mechanical input at t = 5 s, the five remaining generators decelerate as they absorb the 12.8 MW deficit. The frequency nadir reaches 59.53 Hz ($\Delta\omega = -0.0078$ pu) at t $\approx$ 6.0 s — above the 59.4 Hz UFLS threshold. Because the under-frequency relay is never triggered, no load is shed and the IT load factor remains at 1.0 throughout the simulation. The governors on G1–G5 increase mechanical power output over the following seconds. Each unit settles at approximately 15.2 MW (68% of rated), and the system reaches a new droop-governed steady state at 59.71 Hz ($\Delta\omega = -0.0049$ pu). The 0.29 Hz steady-state frequency offset reflects the 5% droop characteristic distributing the

DOI 10.3384/ECP219 180

additional load across the five remaining machines. G6, having lost its prime mover, decelerates further and briefly absorbs a small amount of power from the network. However, because G6's power absorption remains below the reverse-power relay pickup for the required delay, the breaker does not trip and G6 remains connected, spinning down to near-zero power exchange.

Figure 14 shows the more severe scenario with only 5 generators online. Pre-trip, each of the five active generators carries 15.3 MW (69% of rated). When G6 trips at t = 5 s, only four generators remain to serve the 77 MW load, requiring approximately 19.3 MW each (87% of rated) — within machine capability but producing a deeper transient. The frequency nadir reaches 59.33 Hz ($\Delta\omega = -0.0112$ pu) at $t \approx 6.0$ s, breaching the 59.4 Hz UFLS threshold. The under-frequency relay begins integrating at approximately t = 5.4 s when the frequency first crosses the 1% pickup, and the trip signal latches at $t \approx 6.0$ s after the 0.3 s time delay is satisfied. Load shedding disconnects 10% of IT load and also 10% of cooling load, reducing total demand by approximately 12.9 MW. The reverse-power relay on G6 detects motoring at $t \approx 6.4$ s and opens the G6 breaker, cleanly disconnecting the tripped unit from the network. The protection sequence — frequency relay first, then reverse-power breaker — reflects the different pickup conditions and time delays of the two relays. With the shed load removed and G6 disconnected, the four remaining generators stabilize at approximately 15.9 MW each (72% of rated). The post-event steady-state frequency recovers to 59.96 Hz ($\Delta\omega = -0.0007$ pu), substantially closer to nominal than in the 6-generator case. This recovery is a direct consequence of load shedding: the 12.9 MW reduction in demand restores the generation–load balance at a frequency much closer to 60 Hz than the droop-only equilibrium would produce.

The results illustrate a characteristic trade-off in islanded data center design. The N+1 configuration (6 generators for a 5-generator load) rides through the GT trip without any load shedding but settles at a depressed frequency due to droop control. In the degraded configuration (one unit already out for maintenance), the same trip causes a deeper frequency excursion that crosses the UFLS threshold, shedding 12.9 MW of IT load. While the load shedding restores frequency closer to nominal, it represents a partial loss of IT service that would be unacceptable for many data center operators.

The GT trip scenarios presented here deliberately omit several controls and hardware systems that would be present in a production data center installation. Thus the simulated frequency excursions and load-shedding events are more severe than would typically be observed in practice, and the results should be interpreted as a stress test of the underlying generation and protection architecture rather than a prediction of operational performance. Modifications to the frequency control and inclusion of additional hardware such as UPS with battery ride through and battery energy storage systems (BESS) would certainly impact the observed responses.

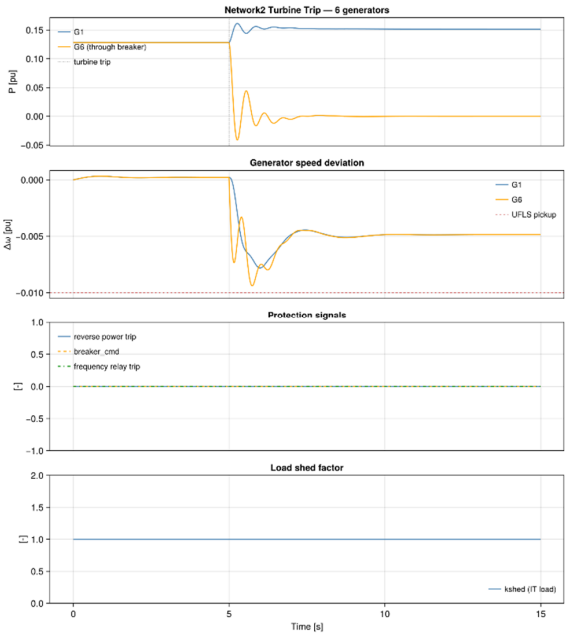


Figure 13. Results from 6 generator simulations

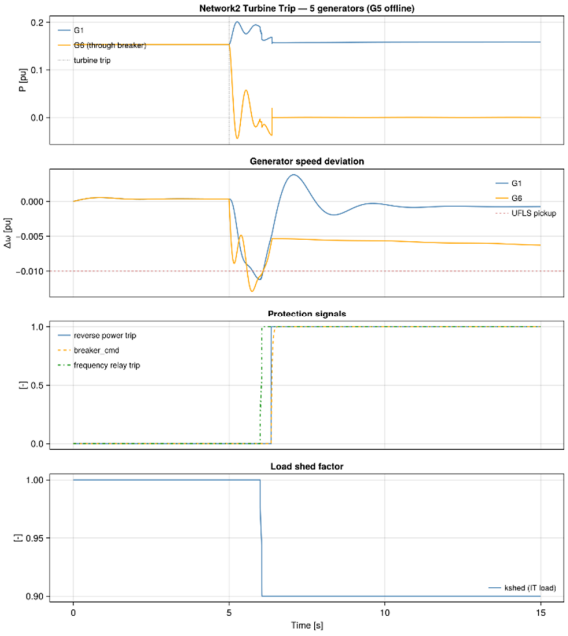


Figure 14. Results from 5 generator simulations

5 Conclusions

This paper presents an initial POC translation of OpenIPSL into Dyad. A representative subset of OpenIPSL was translated leveraging the Dyad AI agent,

DOI 10.3384/ECP219 180

and initial validation of key components and system models were shown. As a first potential application, the Dyad models were used to model a data center power system. Simulations were conducted showing a GT trip with different numbers of GTs initially active. Future work is planned to complete the translation of OpenIPSL into Dyad and continue to build out modeling capabilities to support data center power applications.

References

de Castro, M., Winkler, D., Laera, G., Vanfretti, L., Dorado-Rojas, S.A., Rabuzin, T., Mukherjee, B., and M. Navarro (2023). "Version [OpenIPSL 2.0.0] - iTesla Power Systems Library (iPSL): A Modelica library for phasor time-domain simulations". SoftwareX, Volume 21, 2023, 101277, ISSN 2352-7110, DOI: 10.1016/j.softx.2022.101277.

de Castro Fernandes, M., Vanfretti, L., de Oliveira, J.G., and M. Baudette (2019). "A Fundamental Time-Domain and Linearized Eigenvalue Analysis of Coalesced Power Transmission and Unbalanced Distribution Grids using Modelica and the OpenIPSL ", *Proceedings of the 13th International Modelica Conference*, Regensburg, Germany, March 4–6, 2019, DOI: 10.3384/ecp19157617

Fachini, F., Bhattacharjee, S., Aguilera, M., Vanfretti, L., Laera, G., Bogodorova, T., Moftakhari, A., Huylo, M., and A. Novoselac (2023). "Exploiting Modelica and the OpenIPSL for University Campus Microgrid Model Development", *Proceedings of the 15th International Modelica Conference 2023*, October 9-11, Aachen, Germany. DOI: 10.3384/ecp21181147.

JuliaHub (2026). Dyad v3.0.0. URL: https://help.juliahub.com/dyad/dev/

Ma, Y., Gowda, S., Anantharaman, R., Laughman, C., Shah, V., and C. Rackauckas. (2021), "ModelingToolkit: A Composable Graph Transformation System For Equation-Based Modeling," arXiv:2103.05244

Milano, F. (2005). "An Open Source Power System Analysis Toolbox," IEEE Transactions on Power Systems, vol. 20, no. 3, pp. 1199-1206. URL: https://ieeexplore.ieee.org/document/1490569/

Siemens PTI (2017). "PSS®E 34.2.0 Model Library," Siemens PTI, Schenectady, NY, 2017.

Vanfretti, L. and C.R. Laughman (2024). "Power System Modeling for Identification and Control Applications using Modelica and OpenIPSL," *2024 IEEE Conference on Control Technology and Applications (CCTA),* Newcastle upon Tyne, United Kingdom, 2024, pp. 525-532, DOI: 10.1109/CCTA60707.2024.10666594.

Winkler, D. (2018). "Analysing the stability of an Islanded hydro-electric power system", *Proceedings of The American Modelica Conference 2018*, October 9-10, Samberg Conference Center, Cambridge MA, USA, Issue 154, 2019-02-26, pp. 112-120, ISSN 1650-3740. DOI: 10.3384/ecp18154103

DOI **10.3384/ECP219 180**

A Single-Element Multi-Point Compliance for Rigid-Body Connections

Behzad Danaei[1] Srikanth Sivaramakrishnan[1]

[1]General Motors, Charlotte Technical Center, Charlotte, NC, USA, behzad.danaei@gmail.com

Abstract

Many engineering assemblies connect two rigid bodies at multiple mounting points using compliant elements (e.g., vehicle subframes to body/chassis via bushings). Modeling such systems with one 6-DOF bushing per mounting point is conceptually straightforward but couples many stiff parallel force paths into the rigid-body equations, adding kinematic loops and severe numerical stiffness that hamper robustness and computational efficiency. This paper derives a single-element multi-point compliance (SEMPC) that aggregates an arbitrary set of local translational/rotational stiffnesses into a single 6×6 spatial stiffness at a body reference frame. The approach preserves accuracy to first order in relative orientation, adds no algebraic complexity compared with an ordinary point bushing, and provides a linear map to distribute the net wrench back to the underlying mounting points for sensing/diagnostics. We extend the formulation with a second-order (inertance–damper–spring) filter in wrench space to support stability in high stiffness values and real-time step sizes. A Modelica component formulation is given, along with examples demonstrating the usage of this model.

Keywords: Multi-point compliance, MultiBody Dynamics, Spatial stiffness

1 Introduction

Multiple compliant attachments between two rigid bodies are ubiquitous in engineering assemblies: subframe-to-chassis interfaces, engine mounts, exhaust hangers, and double-wishbone control arms with distinct front/rear bushings. The conventional modeling approach instantiates one 6-DOF bushing per attachment point. As a motivating example, consider a single rigid body connected to ground through six such bushings (the configuration analysed in the case study). The assembly still has only six mechanical degrees of freedom, yet it does not map cleanly onto a single 6-DOF element, and the two natural formulations each carry a penalty.

If the compliant elements are treated as pure force elements, the deflection at every mount is computed algebraically from the body coordinates and fed back as a force into the rigid-body equations of motion. At the very high stiffness values typical of automotive mounts, this direct force coupling makes the system strongly coupled and numerically stiff—the dominant free-response frequencies are driven into the kilohertz range—which forces impractically small step sizes for explicit fixed-step integration and degrades robustness in multi-domain, real-time simulations.

If, instead, the compliance deflection is selected as a state—a natural and often desirable choice, since it yields an explicit state form and avoids the stiff algebraic force coupling above—a different problem arises when several such elements connect the same two bodies. Bodies A and B share a single relative pose, so only one deflection can be an independent state; the remaining deflections are kinematically tied to it and must satisfy loop-closure equations. These closed-loop kinematic constraints are position constraints that are nonlinear (they involve rotation composition on $SO(3)$) and reintroduce nonlinear algebraic systems into the model. In other words, the redundant parallel paths form a closed kinematic loop precisely at the level of state selection. The single-element formulation proposed here avoids both penalties: it aggregates all mounts into one 6-DOF element with a single set of deflection states, so it is simultaneously free of the stiff algebraic force coupling and of the loop-closure constraints created by redundant parallel compliances.

The theoretical foundations for this class of problems are well established. Spatial stiffness and compliance modeling via screw theory and parallel spring synthesis has a long trajectory in robotics and mechanism design, with realization and classification results for spatial stiffness matrices—including translational–rotational coupling—firmly in place (Huang and Schimmels 1998a; Huang and Schimmels 2001; Huang and Schimmels 1998b). In vehicle dynamics, 6-DOF bushing models and hydro-mounts have been widely studied, including nonlinear and hysteretic extensions (Ok et al. 2008; Fredette et al. 2019; Ueda et al. 2016; Lee et al. 2003). On the simulation side, the Modelica MultiBody library provides the standard foundation for 3D rigid-body modeling (Otter et al. 2003). Despite this depth of prior work, no existing approach directly addresses the efficient aggregation of an arbitrary set of attachment points into a single, well-conditioned compliance element that is free of additional algebraic complexity and suitable for both offline and real-time simulation.

We propose a single-element multi-point compliance (SEMPC): a single 6-DOF element that encapsulates the aggregate effect of n mounting points without introduc-

DOI 10.3384/ECP219 189

ing extra algebraic unknowns or kinematic loops. The key idea is to pre-assemble a 6×6 spatial stiffness matrix from the local stiffnesses and attachment geometry, and then compute the net wrench from the relative pose at a chosen reference frame. A companion linear map recovers per-point wrenches for load analysis and diagnostics. The approach complements the established spatial stiffness literature by collapsing multi-point attachments into a single, physically interpretable element while retaining a rigorous back-mapping to individual mounts.

2 Local Mount Models and Net Wrench

Consider bodies A and B with reference frames $\{A\}$ and $\{B\}$ shown in Figure 1. Unless otherwise stated, all vectors are expressed in the local frame $\{A\}$. Let $\Delta\mathbf{r} \in \mathbb{R}^3$ and $\Delta\theta \in \mathbb{R}^3$ denote the relative translation and (small) rotation of $\{B\}$ with respect to $\{A\}$. For $i = 1, \ldots, n$, let $\mathbf{r}_{i/a}$ be the vector from the reference origin of $\{A\}$ to attachment point i on body A (equivalently, the coincident point on B in the undeformed configuration). Define $[\cdot]_\times$ as the skew operator.

For small relative rotations (first-order linearization on $SO(3)$), the displacement at point i resolved in $\{A\}$ is

$$\Delta\mathbf{r}_i = \Delta\mathbf{r} - [\mathbf{r}_{i/a}]_\times \Delta\theta. \tag{1}$$

This approximation is accurate to first order in $|\Delta\theta|$, with a truncation error of $O(|\Delta\theta|^2)$ (Goldstein et al. 2002). For typical mounting deflections, such small-angle assumptions remain valid across the operational range of most compliant connections between nominally aligned frames.

At point i, we allow a (possibly anisotropic) linear translational stiffness $\mathbf{K}_i \in \mathbb{R}^{3\times3}$ and a rotational stiffness $\mathbf{K}_{\theta,i} \in \mathbb{R}^{3\times3}$ about the local attachment. The point wrench contributions are

$$\mathbf{F}_i = \mathbf{K}_i \Delta\mathbf{r}_i, \tag{2}$$

$$\mathbf{M}_i = \mathbf{K}_{\theta,i} \Delta\theta. \tag{3}$$

The total wrench at $\{A\}$ is then

$$\mathbf{F} = \sum_{i=1}^{n} \mathbf{F}_i, \tag{4}$$

$$\mathbf{M} = \sum_{i=1}^{n} \left(\mathbf{M}_i + \mathbf{r}_{i/a} \times \mathbf{F}_i\right). \tag{5}$$

Substituting (1)–(3) and collecting terms yields the aggregate spatial stiffness relation

$$\begin{bmatrix} \mathbf{F} \\ \mathbf{M} \end{bmatrix} = \mathcal{K} \begin{bmatrix} \Delta\mathbf{r} \\ \Delta\theta \end{bmatrix} \tag{6}$$

where the aggregate 6×6 spatial stiffness matrix $\mathcal{K} \in \mathbb{R}^{6\times6}$ is

$$\mathcal{K} = \sum_{i=1}^{n} \begin{bmatrix} \mathbf{K}_i & -\mathbf{K}_i [\mathbf{r}_{i/a}]_\times \\ [\mathbf{r}_{i/a}]_\times \mathbf{K}_i & \mathbf{K}_{\theta,i} - [\mathbf{r}_{i/a}]_\times \mathbf{K}_i [\mathbf{r}_{i/a}]_\times \end{bmatrix}. \tag{7}$$

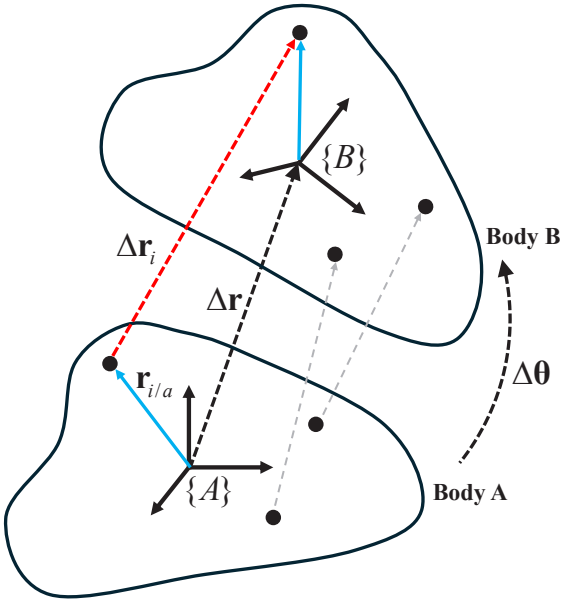


Figure 1. Bodies A and B are connected by n compliant elements (black dots) arranged in parallel, each capable of deforming along three translational and three rotational directions, transmitting forces and moments.

Equation (6) is the central result: the multi-point assembly is equivalent (to first order) to a single 6-DOF element with stiffness $\mathcal{K} \in \mathbb{R}^{6\times6}$ precomputable from geometry and local stiffnesses. Equation (6) is the wrench acting on body A resolved in $\{A\}$; by Newton's third law the wrench transmitted to body B is equal and opposite, $[\mathbf{F}_B; \mathbf{M}_B] = -[\mathbf{F}; \mathbf{M}]$, when resolved at the coincident reference point.

2.1 Per-Point Wrench Distribution (Sensing)

While the solver only needs the net wrench (6), it is often useful to estimate loads at each attachment. Define the block matrix

$$\mathcal{A}_i \triangleq \begin{bmatrix} \mathbf{K}_i & -\mathbf{K}_i [\mathbf{r}_{i/a}]_\times \\ \mathbf{0} & \mathbf{K}_{\theta,i} \end{bmatrix}. \tag{8}$$

From (2)–(3) and (6),

$$\begin{bmatrix} \mathbf{F}_i \\ \mathbf{M}_i \end{bmatrix} = \mathcal{A}_i \mathcal{K}^{-1} \begin{bmatrix} \mathbf{F} \\ \mathbf{M} \end{bmatrix}. \tag{9}$$

Equation (9) acts as a load splitter consistent with the assumed local stiffnesses and returns the wrench $[\mathbf{F}_i; \mathbf{M}_i]$ that mount i applies to body A; the equal and opposite reaction $-[\mathbf{F}_i; \mathbf{M}_i]$ is applied to body B at the same point. Moreover, $\mathcal{K}^{-1}$ and $\mathcal{A}_i$ can be treated as precomputed parameters, allowing them to be evaluated prior to runtime and thereby significantly reducing the computational cost.

DOI 10.3384/ECP219 189

2.2 Dynamic Extension: Damping and Inertance

To stabilize simulations with very stiff $\mathcal{K}$ at larger step sizes, we introduce a second-order filter in wrench space:

$$\mathcal{K}^{-1}\begin{bmatrix}\mathbf{F}\\ \mathbf{M}\end{bmatrix} = \tau^2\frac{d}{dt}\begin{bmatrix}\mathbf{v}_{\text{rel}}\\ \boldsymbol{\omega}_{\text{rel}}\end{bmatrix} + 2\zeta\tau\begin{bmatrix}\mathbf{v}_{\text{rel}}\\ \boldsymbol{\omega}_{\text{rel}}\end{bmatrix} + \begin{bmatrix}\Delta\mathbf{r}\\ \Delta\boldsymbol{\theta}\end{bmatrix}. \quad (10)$$

Here $\mathcal{K}^{-1}$ is the aggregate spatial compliance, i.e. the inverse of the spatial stiffness $\mathcal{K}$ in (7); τ is a time constant, ζ a damping ratio, $\mathbf{v}_{\text{rel}}$ and $\boldsymbol{\omega}_{\text{rel}}$ are relative twist components. Equation (10) corresponds to adding a diagonalized apparent inertia and viscous damping in the compliance coordinates, lowering the natural frequency and improving integrator robustness. Typically, the time constant τ should be selected so that the resulting compliance natural frequency lies above the highest excitation frequency that must be preserved, yet below the Nyquist limit corresponding to the chosen integration step size; in particular, when using explicit Euler integration with step size h, stability requires that τ exceed the lower bound

$$\tau > \begin{cases}\dfrac{h}{2\zeta}, & 0 < \zeta \le 1,\\[2ex] \dfrac{h}{2}\left(\zeta + \sqrt{\zeta^2 - 1}\right), & \zeta > 1.\end{cases} \quad (11)$$

3 Implementation in Modelica

As shown in Figure 2, the SEMPC is implemented as a Modelica component with `frame_a` and `frame_b` connectors from the MultiBody library. The component is composed of two cooperating sub-models: a passive `MultiMountSensor` and a `VariableTranslationRotation` actuator. Mount positions $\mathbf{r}_{i/a}$ and per-axis stiffness values (representing diagonal $\mathbf{K}_i$ and $\mathbf{K}_{\theta,i}$) are supplied as parameters.

Precomputed Parameters

Both the aggregate spatial compliance $\mathcal{K}^{-1}$ and the per-mount load splitter matrices $\mathcal{A}_i\mathcal{K}^{-1}$ are precomputed as `final parameter` constants before simulation begins. This is done by assembling $\mathcal{K}$ from (6) and inverting it once; no matrix factorization is required at runtime. Listing 1 shows the core assembly loop.

Listing 1. Aggregate spatial stiffness assembly (from `complianceMatrix` function)

```
// Assemble 6x6 spatial stiffness K from
// n mount contributions
for i in 1:n loop
  K := K + [
    diagonal(kT[i]),
      -diagonal(kT[i])*skew(r[i]-r0);
    skew(r[i]-r0)*diagonal(kT[i]),
      diagonal(kR[i])
      - skew(r[i]-r0)*diagonal(kT[i])
        *skew(r[i]-r0)];
end for;
```

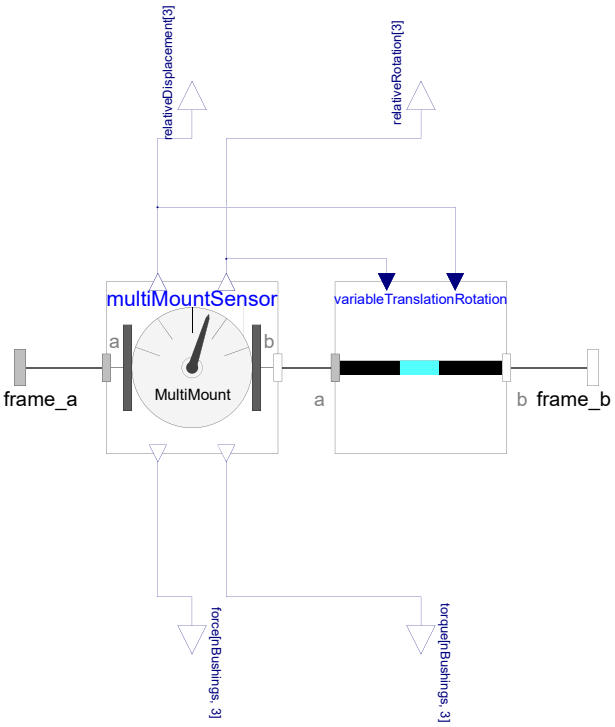


Figure 2. Overview of the graphics layer in the Modelica implementation of SEMPC.

```
// Invert once to obtain the 6x6
// spatial compliance K^{-1}
K_inv := Modelica.Math.Matrices.solve2(
  K, identity(6));
```

Sensor: Static Compliance Mapping

The `MultiMountSensor` is a zero-displacement element that reads the current reaction forces $[\mathbf{F};\mathbf{M}]$ at `frame_a` and produces:

- The static compliance displacement: $[\Delta\mathbf{r};\Delta\theta] = \mathcal{K}^{-1}[\mathbf{F};\mathbf{M}]$,
- Per-mount load outputs: $[\mathbf{F}_i;\mathbf{M}_i] = \mathcal{A}_i\mathcal{K}^{-1}[\mathbf{F};\mathbf{M}]$ via (9).

Because `frame_a` and `frame_b` are directly connected inside the sensor, it introduces no dynamic states. Listing 2 shows the key protected parameters and equations.

Listing 2. Core of `MultiMountSensor`: precomputed matrices and compliance equations

```
protected
  // Aggregate spatial compliance K^{-1}
  // (precomputed once)
  final parameter Real complianceMatrix
      [6,6] =
    Functions.complianceMatrix(nBushings,
      rBushing, rFrame,
        translationStiffness,
      rotationStiffness);
  // Per-mount load splitter A_i * K^{-1}
```


DOI 10.3384/ECP219 189

```
  // (precomputed once)
  final parameter Real loadSplitter[
      nBushings,6,6] = {
    Functions.stiffnessMatrix(rBushing[i],
        rFrame,
      translationStiffness[i],
          rotationStiffness[i])
      * complianceMatrix for i in 1:
          nBushings};
equation
  // Static compliance map:
  // [dr; dtheta] = K^{-1}*[F; M]
  [relativeDisplacement; relativeRotation]
      =
    -complianceMatrix * [frame_a.f; frame_a
        .t];
  // Per-mount load splitter (Eq. 9)
  for i in 1:nBushings loop
    [force[i]; torque[i]] =
      -loadSplitter[i,:,:] * [frame_a.f;
          frame_a.t];
  end for;
```

Actuator: Second-Order Filter

The `VariableTranslationRotation` actuator takes $(\Delta\mathbf{r}, \Delta\theta)$ from the sensor as target inputs and drives a second-order ODE (10):

$$\tau^2\ddot{\delta} + 2\zeta\tau\dot{\delta} + \delta = \mathbf{u}, \tag{12}$$

separately for the translational and rotational channels, where δ is the actual applied displacement of `frame_b` and $\mathbf{u}$ is the target input. Concretely, (12) is applied twice: once with $(\delta, \mathbf{u}) = (\Delta\mathbf{r}, \Delta\mathbf{r}_{\text{target}})$ for the translational channel and once with $(\delta, \mathbf{u}) = (\Delta\theta, \Delta\theta_{\text{target}})$ for the rotational channel, where $\Delta\mathbf{r}_{\text{target}}$ and $\Delta\theta_{\text{target}}$ are the static-compliance targets supplied by the sensor. Because each channel is a second-order system in a 3-vector, its state consists of the value and its rate; the four resulting vector states $(\Delta\mathbf{r}, \dot{\Delta\mathbf{r}}, \Delta\theta, \dot{\Delta\theta})$—twelve scalar states in total—are selected as simulation states. This explicit state form avoids algebraic loops and provides robustness at high stiffness values. The cutoff frequency f_c (with $\tau = 1/(2\pi f_c)$) and damping ratio ζ are exposed as user parameters. Listing 3 shows the filter equations and pose enforcement.

Listing 3. Second-order filter and frame enforcement in `VariableTranslationRotation`

```
final parameter SI.Time tau = 1/(2*pi*
    freqCutoff);
// ...
// Translational filter: states dr, drDot
drDot = der(dr);
der(drDot) = (r - dr)/tau^2 - 2*
    dampingRatio*drDot/tau;
// Rotational filter: states dphi, dphiDot
dphiDot = der(dphi);
der(dphiDot) = (phi - dphi)/tau^2
  - 2*dampingRatio*dphiDot/tau;
// Enforce filtered pose on frame_b
frame_b.r_0 = frame_a.r_0
  + Frames.resolve1(frame_a.R, dr);
frame_b.R = Frames.absoluteRotation(frame_a
    .R,
  Frames.axesRotations(rotationOrder, dphi,
      der(dphi)));
```

4 Case Study

To evaluate the SEMPC formulation and its practical utility, a controlled benchmark study is conducted comparing two modeling approaches over an identical problem configuration: (i) a reference model, in which each attachment point is represented by an independent 6-DOF linear spring-damper element, consistent with the conventional multi-element approach; and (ii) the SEMPC model, which aggregates all attachment points into the single compliance element proposed in this paper.

Simulation Setup

The benchmark comprises two rigid bodies (Figure 3): Body A is fixed to the inertial frame, and Body B is connected to Body A via n compliant attachment points distributed equidistantly on a circle of radius $R = 0.3\,\text{m}$ in the y-z plane, with the i-th attachment located at angular position $\theta_i = 2\pi(i-1)/n$. To introduce representative stiffness heterogeneity, the translational and rotational stiffnesses of the i-th attachment are scaled geometrically as

$$\mathbf{K}_i = \mathbf{K}_1(1+\alpha)^{i-1}, \qquad \mathbf{K}_{\theta,i} = \mathbf{K}_{\theta,1}(1+\alpha)^{i-1},$$

with base stiffnesses $\mathbf{K}_1 = \text{diag}(2.27, 1.41, 2.17) \times 10^7\,\text{N/m}$ and $\mathbf{K}_{\theta,1} = \text{diag}(2.87, 1.73, 0.84) \times 10^5\,\text{N·m/rad}$, and a per-bushing increment factor $\alpha = 0.10$.

Body B has mass $m = 100\,\text{kg}$ and principal moments of inertia $I_{xx} = I_{yy} = I_{zz} = 1\,\text{kg·m}^2$; gravitational acceleration is set to zero to isolate the compliance dynamics. A broadband wrench is applied directly to Body B: three independent uniform noise signals (sampled at 100 Hz, amplitude ± 1) are each passed through a second-order Butterworth low-pass filter with cutoff frequency $f_c = 30\,\text{Hz}$; the force channel gains are $[70, 12, 44]\,\text{kN}$ and the torque channel gains are $[1.0, 2.0, 4.0]\,\text{kN·m}$. All simulations are executed in Dymola over a 0.5 s interval using the explicit first-order Euler integrator in fixed-step mode, with no adaptive step-size control, so that numerical stability boundaries can be characterized precisely. Explicit fixed-step Euler is chosen deliberately: it is the integration scheme most representative of the hard-real-time and hardware-in-the-loop targets that motivate this work, where implicit or adaptive solvers are frequently unavailable or too costly per step, and it exposes the stability boundary most sharply for a fair comparison of the two models. Implicit schemes (e.g., backward Euler) or semi-implicit symplectic (semi-implicit Euler) schemes would relax the step-size restriction of the reference model, but at the cost of per-step linear solves or additional implementation complexity on real-time targets. The SEMPC formulation is complementary to such solvers and benefits equally from them, since

DOI 10.3384/ECP219 189

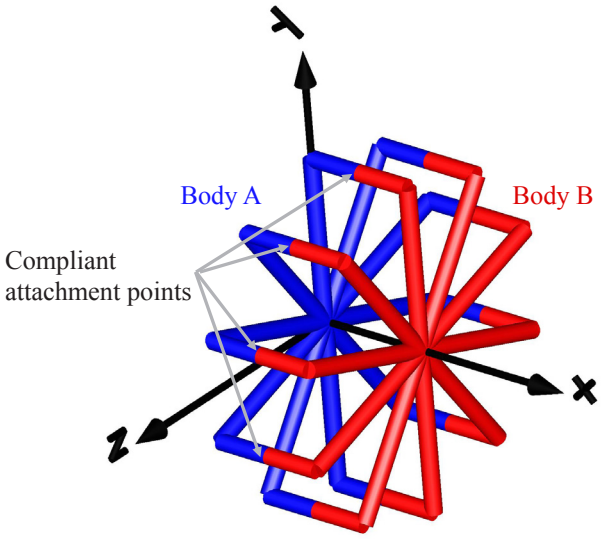


Figure 3. Benchmark simulation setup. Body A (blue) is rigidly fixed to the inertial frame. Body B (red) is connected to Body A via n compliant attachment points distributed equidistantly on a circle of radius $R = 0.3\,\mathrm{m}$ in the y-z plane. A three-component sinusoidal wrench (forces and torques applied at Body B) drives the relative compliance response used for validation.

it reduces the number of stiff states independently of the integrator; the explicit-Euler comparison here isolates and quantifies exactly that state-reduction benefit.

Study 1: Numerical Stability and Real-Time Feasibility

The first study fixes $n = 10$ and determines the largest fixed step size at which each model remains numerically stable under the explicit Euler integrator. The stiffness values employed are representative of production automotive isolation mounts; the high aggregate translational stiffness on the order of $10^8\,\mathrm{N/m}$ drives the dominant free-response natural frequencies of the reference model well into the kilohertz range, imposing a stringent Euler stability requirement.

For the reference model, the maximum stable step size is found to be $h_{\mathrm{ref}} = 1 \times 10^{-7}\,\mathrm{s}$. Simulating 1 s of physical time therefore requires 10^7 integration steps, yielding a wall-clock time approximately $40\times$ longer than real time—clearly impractical for real-time deployment. This restriction arises from the direct algebraic coupling between the stiff spring forces and the rigid-body equations of motion: the high aggregate stiffness constrains the Euler stability region to a sub-microsecond step size.

The SEMPC model eliminates this coupling by embedding the stiffness response within the second-order ODE filter, which limits the highest dynamically active frequency to $f_c = 250\,\mathrm{Hz}$. As a result, the SEMPC model is stable at a step size of $h_{\mathrm{SEMPC}} = 1 \times 10^{-3}\,\mathrm{s}$—four orders of magnitude larger than h_{ref}—and completes the same 1 s simulation approximately $25\times$ faster than real time, placing real-time and hardware-in-the-loop (HIL) deployment well within reach.

The relative displacement of Body B with respect to Body A is shown in Figure 4. The SEMPC solution closely tracks the reference across the dominant response modes. The small error is the expected consequence of the second-order filter and constitutes the fundamental accuracy–efficiency trade-off of the proposed formulation.

Study 2: Per-Mount Load Recovery

Figure 5 compares the per-mount forces and torques recovered by both models. The SEMPC per-mount estimates—computed via the precomputed load-splitter map (9)—agree closely with the reference outputs across all attachment points. Crucially, this load-recovery computation introduces no additional differential states: all splitter matrices are assembled once at initialization, and the `MultiMountSensor` sub-component operates purely as an algebraic observer at runtime. The per-mount sensing capability can therefore be deployed independently of the SEMPC actuator block with no impact on the simulation step size, model index, or run-time cost.

Study 3: Computational Scalability

The third study varies n systematically while holding all other parameters constant, in order to characterise how wall-clock simulation time scales with the number of attachment points for each approach. For the reference model, each additional bushing introduces new elastic states and tightens the Euler stability constraint—since the aggregate stiffness and dominant natural frequencies grow with n—so the required simulation time increases with n. For the SEMPC, adding mounts only modifies the offline assembly of the 6×6 aggregate stiffness matrix, an $O(n)$ initialization cost, while the runtime ODE remains largely insensitive to n.

Figure 6 shows the measured wall-clock simulation time as a function of n. The reference model exhibits a clear increase in computational cost with growing n, whereas the SEMPC approach scales much more weakly. The modest increase observed for SEMPC is primarily attributable to initialization and the per-mount load-splitting step. This near-constant scaling arises from the dimension-reduction property of the SEMPC formulation: regardless of the number of underlying mounts, the runtime dynamics are represented by a single 6-DOF element with a fixed number of ODE states.

DOI 10.3384/ECP219 189

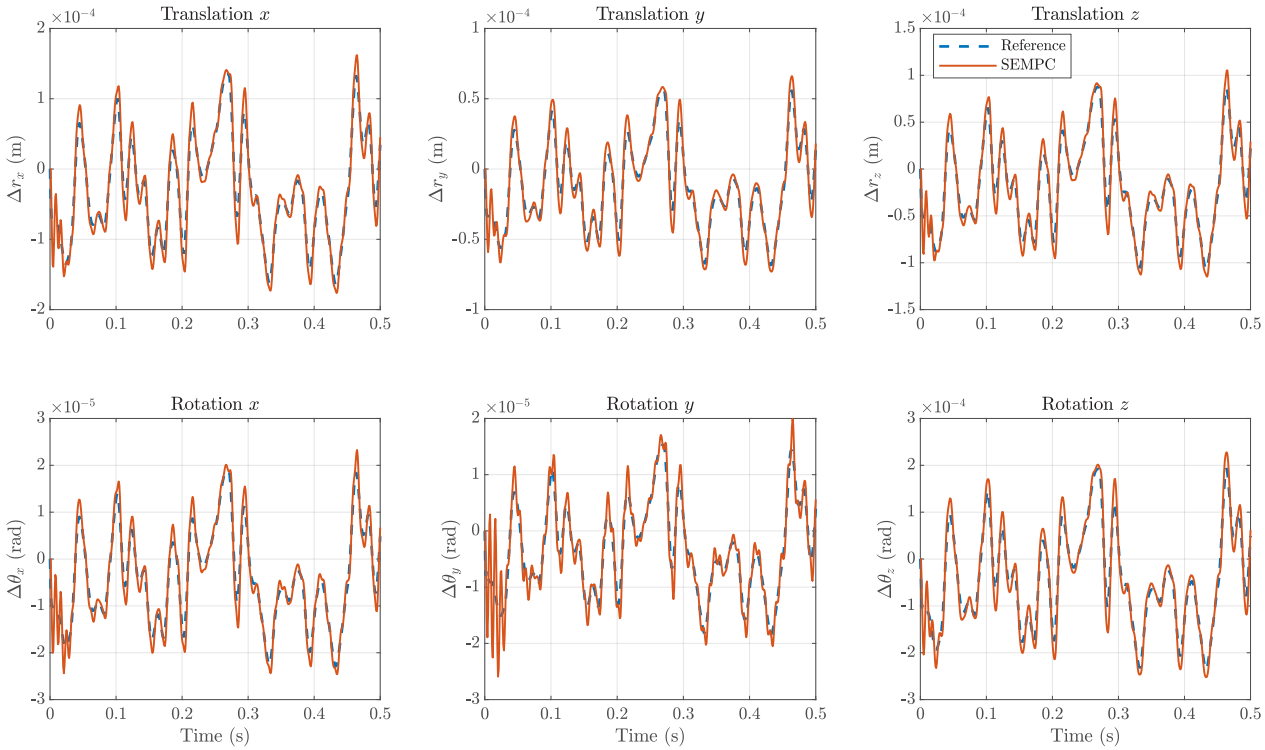


Figure 4. Study 1: Relative translational (top) and rotational (bottom) displacement of Body B with respect to Body A. Solid orange: SEMPC model; dashed blue: reference model. The SEMPC closely tracks the reference.

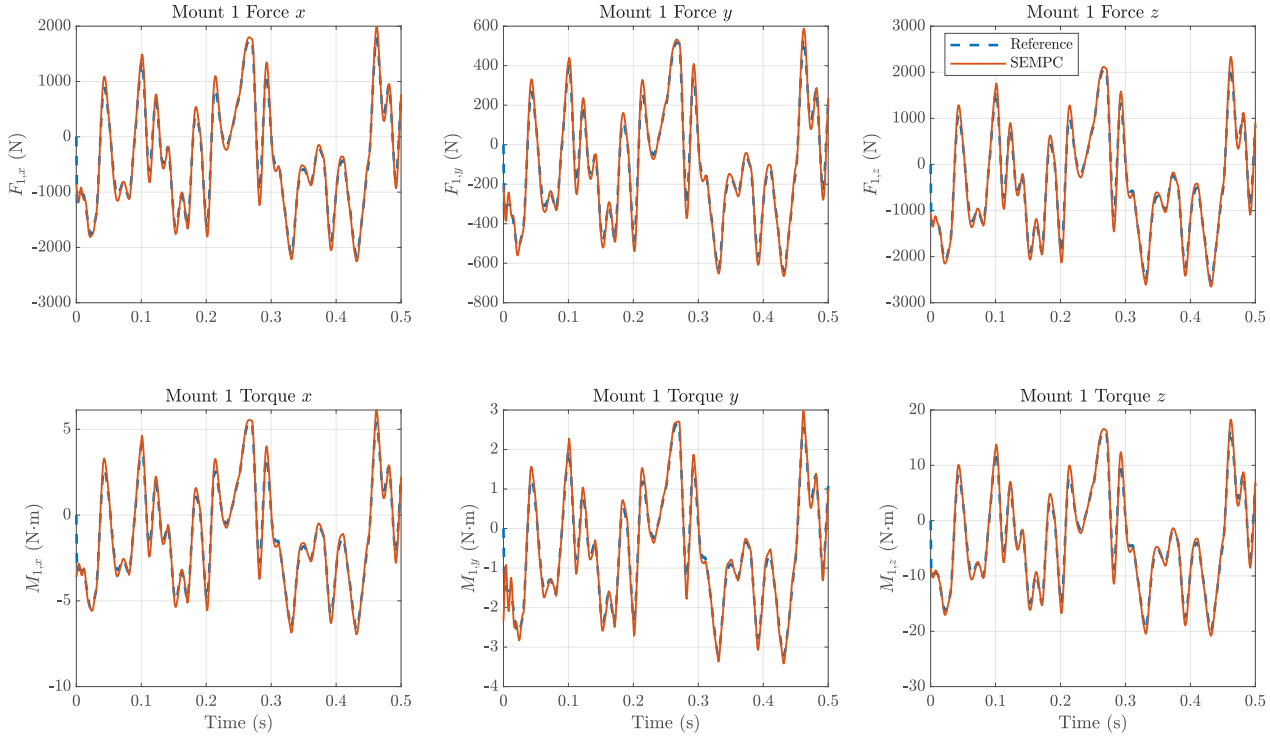


Figure 5. Study 2: Force (top) and torque (bottom) at the first attachment point recovered by the SEMPC load-splitter map (9) (solid orange) versus the reference model (dashed blue). The SEMPC estimates agree closely with the reference without introducing additional states.

DOI **10.3384/ECP219 189**

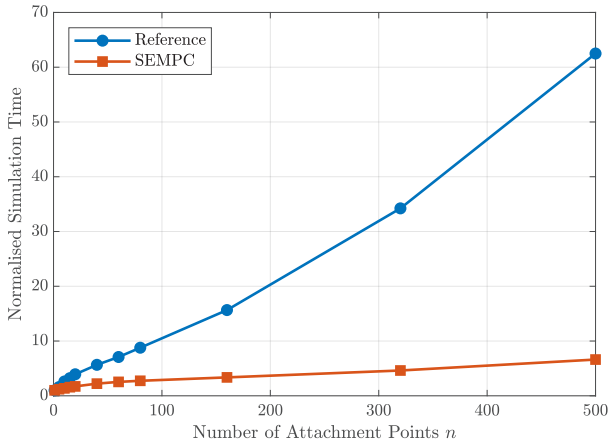


Figure 6. Study 3: Normalised wall-clock simulation time versus number of attachment points n. The reference model cost grows with n, while the SEMPC cost remains essentially constant due to the fixed 12-state runtime ODE.

5 Conclusion

The SEMPC consolidates a cluster of mounts into a single, well-conditioned 6-DOF element, offering both computational efficiency and physical interpretability. It can be seamlessly integrated into Modelica multibody models, supports per-mount load sensing, and—when combined with a second-order filter—enables stable real-time performance even at high stiffness levels.

Regarding its limitations, the first-order rotational approximation reduces accuracy under large relative rotations. Additionally, nonlinear or hysteretic mount behavior can be incorporated through incremental linearization, yielding time-varying stiffness matrices $\mathbf{K}_i(t)$ and $\mathbf{K}_{\theta,i}(t)$ while preserving the same assembly framework.

References

Fredette, Luke, Siddharth Rath, and Rajendra Singh (2019). "Nonlinear Fluid Damping Models for Hydraulic Bushing under Sinusoidal or Transient Excitation". In: *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* 233.3, pp. 595–604. DOI: 10.1177/0954407017751787.

Goldstein, Herbert, Jr. Charles P. Poole, and John L. Safko (2002). *Classical Mechanics*. 3rd ed. San Francisco, CA: Addison-Wesley. ISBN: 978-0-201-65702-9.

Huang, Shuguang and Joseph M. Schimmels (1998a). "Achieving an Arbitrary Spatial Stiffness with Springs Connected in Parallel". In: *Journal of Mechanical Design* 120.4, pp. 520–526. DOI: 10.1115/1.2829309.

Huang, Shuguang and Joseph M. Schimmels (1998b). "The Bounds and Realization of Spatial Stiffnesses Achieved with Simple Springs Connected in Parallel". In: *IEEE Transactions on Robotics and Automation* 14.3, pp. 466–475. DOI: 10.1109/70.678455.

Huang, Shuguang and Joseph M. Schimmels (2001). "A Classification of Spatial Stiffness Based on the Degree of Translational–Rotational Coupling". In: *Journal of Mechanical Design* 123.3, pp. 353–358. DOI: 10.1115/1.1374197.

Lee, Seong Beom, Jung-Woog Shin, and Alan S. Wineman (2003). "An Experimental Study of Nonlinear Viscoelastic Bushing Model for Axial Mode". In: *KSME International Journal* 17.9, pp. 1324–1331. DOI: 10.1007/BF02982473.

Ok, Jin-Kyu, Wan-Suk Yoo, and Jeong-Hyun Sohn (2008). "New Nonlinear Bushing Model for General Excitations Using Bouc–Wen Hysteretic Model". In: *International Journal of Automotive Technology* 9.2, pp. 183–190. DOI: 10.1007/s12239-008-0023-8.

Otter, Martin, Hilding Elmqvist, and Sven Erik Mattsson (2003). "The New Modelica MultiBody Library". In: *Proceedings of the 3rd International Modelica Conference*. Ed. by Peter Fritzson. Linköping, Sweden: Modelica Association, pp. 311–330. URL: https://modelica.org/events/Conference2003/papers/h37_Otter_multibody.pdf.

Ueda, Masahiro, Satoshi Ito, and Daichi Suzuki (2016). "Application Study of Nonlinear Viscoelastic Constitutive Model for Dynamic Behavior of Suspension Arm Bushing". In: *SAE International Journal of Passenger Cars–Mechanical Systems* 9.1, pp. 336–345. DOI: 10.4271/2016-01-1375.

DOI **10.3384/ECP219 189**

A 2-DOF MultiBody Track Joint for Race Car Optimization

Srikanth Sivaramakrishnan[1] Behzad Danaei[1] Adwait Verulkar[1] Ryan Trinter[1]

[1]General Motors, USA, {srikanth.sivaramakrishnan, behzad.danaei, adwait.verulkar, ryan.trinter}@gm.com

Abstract

`TrackJoint` is a Modelica MultiBody component that turns a three-dimensional road surface into a joint with two degrees of freedom. The two joint coordinates are the distance travelled along a reference line and the sideways offset from that line. Together they give the position, orientation and angular velocity of a single `frame_b` connector. All road geometry reaches the joint as a small set of periodic piecewise-quintic splines, so one component can represent any circuit whose splines have been fitted beforehand. The road constraint is closed with d'Alembert's principle. The two generalized forces that belong to the joint coordinates are set to zero, so no Lagrange multipliers are needed. The joint is connected to a multibody race car in Dymola and the pair is exported as one FMI 3.0 FMU. The same FMU is then used for time-domain simulation and for CasADi/IPOPT minimum lap-time optimization, with no change to the model in between. This works because the joint keeps the arc-length coordinate as a real state and publishes its time derivative, which is the quantity that lets the problem be rewritten with distance We report results for minimum lap time simulations on the Singapore Formula 1 circuit.

Keywords: Modelica, MultiBody, Functional Mockup Unit, race car simulation,minimum lap time, track model, Monge patch, kinematic joint, d'Alembert principle, optimal control, optimization, CasADi, FMIOPT

1 Introduction

Race-car simulation in Modelica relies on a road model that generates accurate surface geometry in terms of position, orientation and angular velocity for each tire contact patch. Inside a generic MultiBody vehicle simulation this is naturally expressed by a Modelica *joint* that constrains the chassis of the vehicle to lie on the road surface while leaving the in-plane road coordinates free. For ideal flat surfaces this role is filled by `Modelica.Mechanics.MultiBody.Joints.Planar`. Although analytically defined ribbons have been used in earlier race-car work (Perantoni and Limebeer 2015; Lovato et al. 2021), they are typically hand-coded inside a surrogate optimization model rather than packaged as a reusable MultiBody component. The idea of a reusable path-following joint that constrains motion to a spline-parametrized curve has been formulated symbolically in a graph-theoretic multibody setting by Hall et al. (2014). This method is used to reduce complex automotive suspension linkages into a single degree-of-freedom joint for computational efficiency. A more closely related precedent is the `Claytex.Mechanics.MultiBody.Joints.Rail` joint of the commercial Claytex library (Briant 2021), which constrains a `frame_b` to follow a one-dimensional B-spline rail parametrized by arc length s and resolves the wrench through the spline-derivative projection that we also adopt below. The Claytex component is, however, restricted to a curve: it leaves a single translational degree of freedom and exposes the orientation either as a heading frame slaved to the rail tangent or as three independent angle splines.

This paper presents `TrackJoint`, a Modelica MultiBody joint that takes the Claytex `Rail` idea as its starting point and extends it from a 1-DOF rail to a 2-DOF Monge-patch racetrack surface. The body is now free to move both along a planar reference line (s) and laterally across it (w); the orientation triad is constructed analytically from the surface partials rather than from a separate angle table; and the wrench is closed by two d'Alembert relations rather than one. The underlying surface model is the Monge patch (Limebeer and van de Merwe 2025; do Carmo 1976)

$$\boldsymbol{r}_b(s,w) = \boldsymbol{r}(s) + w\,\boldsymbol{n}_g(s) + z(s,w)\,\boldsymbol{e}_z, \tag{1}$$

where $\boldsymbol{r}_b(s,w)$ is the world-frame position of the track point identified by (s,w), $\boldsymbol{r}(s) = (r_x(s), r_y(s), 0)$ is the planar reference line, $\boldsymbol{n}_g(s) = (-\partial_s r_y(s), \partial_s r_x(s), 0)$ its ground-plane left normal and

$$z(s,w) = r_z(s) + z_1(s)\,w + z_2(s)\,w^2 + z_3(s)\,w^3 \tag{2}$$

the lateral cubic in w proposed by Limebeer and van de Merwe (2025) for elevation. The six channel splines $\{r_x, r_y, r_z, z_1, z_2, z_3\}$ are obtained from a separate fitting pipeline that processes a measured racetrack point cloud and exports them as periodic piecewise-quintic (C^4 at the knots) B-splines anchored on the minimum-curvature line of the circuit (Verulkar et al. 2026; Heilmeier et al. 2020; Braghin et al. 2008). Here $(\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z)$ denotes the fixed world Cartesian basis. The present paper takes those splines as given and concentrates on the Modelica joint that consumes them.

The contribution of this paper is a Modelica implementation of a track joint that exposes the road surface as a single reusable MultiBody component suitable for both time-domain simulation and optimization. For efficiency the

DOI **10.3384/ECP219 196**

per-step evaluation is implemented with Horner expansion of the Monge cubic, constant-time segment lookup on the uniform break vector and a shared spline evaluator, so that the same component is cheap to evaluate at every collocation point of a CasADi/IPOPT (Andersson et al. 2019; Wächter and Biegler 2006) minimum lap-time problem. The model also ensures sufficient accuracy to the LiDAR measurements of the track and can be used in time-domain simulation of a multibody race car driven around a closed circuit.

The remainder of the paper is organized as follows. Section 2 recalls the Monge-patch surface formulation and fixes the notation used by the joint. Section 3 derives the joint equations from (1)–(2). Section 4 discusses the Modelica implementation and the FMI export. Section 5 reports the two downstream use cases.

2 Surface model

The road is modeled as a smooth orientable two-dimensional manifold $\mathscr{M} \subset \mathbb{R}^3$ parametrized by (1). We reproduce here only the relations needed by the Modelica implementation. A full geometric discussion is given in (Limebeer and van de Merwe 2025) and the companion fitting paper (Verulkar et al. 2026).

Differentiating the surface position once gives the in-surface tangent vectors

$$\partial_s \boldsymbol{r}_b = \big(\partial_s r_x - w\,\partial_{ss} r_y\big)\boldsymbol{e}_x + \big(\partial_s r_y + w\,\partial_{ss} r_x\big)\boldsymbol{e}_y + \partial_s z\,\boldsymbol{e}_z, \tag{3}$$

$$\partial_w \boldsymbol{r}_b = -\partial_s r_y\,\boldsymbol{e}_x + \partial_s r_x\,\boldsymbol{e}_y + \partial_w z\,\boldsymbol{e}_z, \tag{4}$$

in which the lateral derivatives $\partial_s z$ and $\partial_w z$ follow from the cubic (2),

$$\partial_s z = \partial_s r_z + \partial_s z_1\, w + \partial_s z_2\, w^2 + \partial_s z_3\, w^3, \tag{5}$$

$$\partial_w z = z_1 + 2\,z_2\, w + 3\,z_3\, w^2. \tag{6}$$

The surface unit normal is

$$\boldsymbol{n}_z = \frac{\partial_s \boldsymbol{r}_b \times \partial_w \boldsymbol{r}_b}{\|\partial_s \boldsymbol{r}_b \times \partial_w \boldsymbol{r}_b\|}, \tag{7}$$

and a right-handed surface triad is $(\boldsymbol{n}_x, \boldsymbol{n}_y, \boldsymbol{n}_z) = \big(\partial_s \boldsymbol{r}_b/\|\partial_s \boldsymbol{r}_b\|,\ \boldsymbol{n}_z \times \partial_s \boldsymbol{r}_b/\|\partial_s \boldsymbol{r}_b\|,\ \boldsymbol{n}_z\big)$, identical to that adopted in (Limebeer and van de Merwe 2025).

The constraint that keeps the attached body on $\mathscr{M}$ leaves two degrees of freedom, namely the parameters s and w. The Modelica joint carries (s, w) and their first time derivatives $(\dot{s}, \dot{w})$ as states, and works everything else out from the splines $\{r_x, r_y, r_z, z_1, z_2, z_3\}$ at the current s.

2.1 Regularity of the parametrization

The joint only works where the patch (1) is locally invertible. That puts a condition on the reference line. Let $\kappa(s)$ be the signed planar curvature of $\boldsymbol{r}(s)$. The cross product of the in-surface tangents (3)–(4) then has magnitude

$$\|\partial_s \boldsymbol{r}_b \times \partial_w \boldsymbol{r}_b\| \;=\; \big|1 - \kappa(s)\, w\big| \sqrt{1 + \|\nabla z\|^2} \;+\; \mathscr{O}(\partial_s z), \tag{8}$$

so the patch breaks down on the fold

$$w_{\text{fold}}(s) = \frac{1}{\kappa(s)}, \tag{9}$$

which is the centre of curvature of the reference line. Suppose the reference line is the geometric centre-line. In any corner whose radius of curvature drops below the local half-width $b(s)$ of the road we get $|w_{\text{fold}}(s)| < b(s)$, so the fold sits *inside* the drivable strip. There the map $(s, w) \mapsto \boldsymbol{r}_b$ stops being one-to-one, and the joint hands back two different points for the same (s, w) along with a Jacobian that goes to zero. This happens in the slow corners, which are the ones that decide lap time. Anchoring the splines on the minimum-curvature line lowers the curvature peaks and moves the fold outside the strip. The joint therefore needs its input data to satisfy

$$\min_s \frac{1}{|\kappa(s)|\, b(s)} \;>\; 1, \tag{10}$$

which we check for every circuit used in this paper in Section 5.3.

2.2 Continuity required of the splines

The joint needs high derivatives of its inputs. The rotation rates in Section 3.2 use the third s-derivative of the planar channels r_x and r_y. The Modelica tool then differentiates the angular-velocity equation once more while it reduces the DAE index, which takes the requirement up to a fourth derivative. A piecewise quintic with C^4 continuity at the interior knots and at the wrap is the lowest degree that keeps all of these continuous across a knot. A cubic spline leaves $\partial_{ss} \boldsymbol{r}_b$ jumping at every knot, which makes the integrator chatter at the segment boundaries. That is why the fitting pipeline (Verulkar et al. 2026) exports quintics even though cubics would be enough for the surface on its own.

3 Joint formulation

Figure 1 illustrates the overall working principle of the joint. The body `frame_b` is pinned to the road surface $\mathscr{M}$ and retains exactly the two in-plane degrees of freedom (s, w). From these coordinates the joint reconstructs the full pose and angular velocity of the connector and projects the constraint forces onto the surface, as detailed in the subsections below.

3.1 States and frame outputs

The joint has two free coordinates $\boldsymbol{q} = (s, w)$ with derivatives $\dot{\boldsymbol{q}} = (\dot{s}, \dot{w})$. Their values at any instant determine the position, orientation and angular velocity of the `frame_b` connector. The position is given directly by (1) as $\boldsymbol{r}_b(s, w)$, while orientation and angular velocity follow

$$\mathbf{R}_b = [\,\boldsymbol{n}_x \;\; \boldsymbol{n}_y \;\; \boldsymbol{n}_z\,]^{\top}, \tag{11}$$

$$\boldsymbol{\omega}_b = \mathbf{R}_b\big(\partial_s \boldsymbol{\phi}\, \dot{s} + \partial_w \boldsymbol{\phi}\, \dot{w}\big), \tag{12}$$

where $\partial_s \boldsymbol{\phi}, \partial_w \boldsymbol{\phi} \in \mathbb{R}^3$ are the parametric rotation-rate vectors of the surface triad with respect to s and w, and $\mathbf{R}_b$ is

DOI 10.3384/ECP219 196

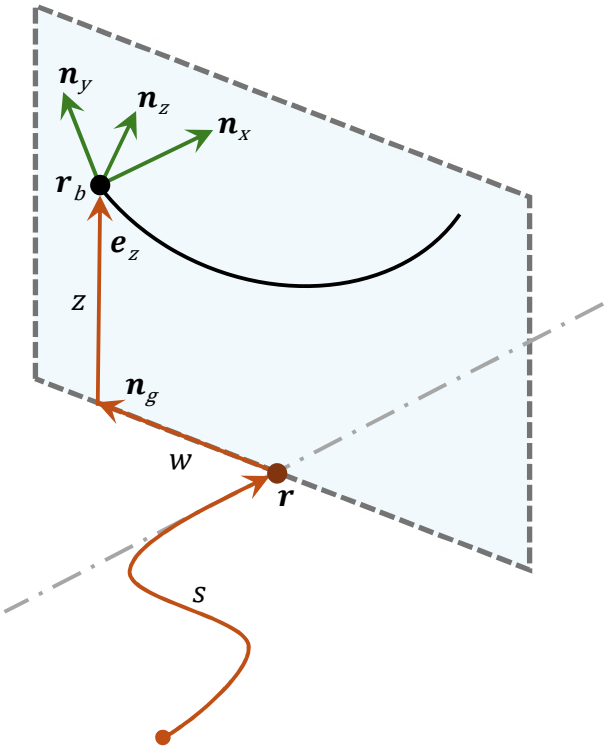


Figure 1. Schematic of the `TrackJoint`. The body `frame_b` is constrained to the Monge-patch surface $\mathscr{M}$ and is free to move along the reference line (s) and laterally across it (w). At the current (s,w) the joint evaluates the six channel splines $\{r_x, r_y, r_z, z_1, z_2, z_3\}$ to build the surface position $\boldsymbol{r}_b(s,w)$ and the right-handed surface triad $(\boldsymbol{n}_x, \boldsymbol{n}_y, \boldsymbol{n}_z)$, whose tangent $\boldsymbol{n}_x$ follows the reference line and whose normal $\boldsymbol{n}_z$ is the surface normal. The reaction wrench is resolved onto the two free directions through the d'Alembert projection.

the rotation matrix associated with that frame; and $\boldsymbol{\omega}_b$ is its world-frame angular velocity. Equation (12) expresses the world-frame angular velocity as a linear combination of two parametric angular velocities, exactly as in the standard MultiBody treatment of any two-parameter rigid kinematic chain.

3.2 Parametric rotation rates

The rotation-rate vectors are obtained in closed form from the surface partials. Let $\boldsymbol{\xi} \in \{s, w\}$ denote either parameter. Since the triad $\mathbf{R}_b = [\boldsymbol{n}_x\ \boldsymbol{n}_y\ \boldsymbol{n}_z]^\top$ is orthonormal for every (s,w), its parametric derivative is skew-symmetric, so there exists a unique vector $\partial_\xi \boldsymbol{\phi}$ with

$$\partial_\xi \boldsymbol{n}_i = \partial_\xi \boldsymbol{\phi} \times \boldsymbol{n}_i, \qquad i \in \{x, y, z\}, \tag{13}$$

and $\partial_\xi \boldsymbol{\phi}$ may be recovered from any two columns by $\partial_\xi \boldsymbol{\phi} = \frac{1}{2}\sum_i \boldsymbol{n}_i \times \partial_\xi \boldsymbol{n}_i$. Building it from the two pieces that the surface partials already give is cheaper and better conditioned, and is implemented in this code.

Write $\boldsymbol{c} = \partial_s \boldsymbol{r}_b \times \partial_w \boldsymbol{r}_b$ for the un-normalized binormal, so that $\boldsymbol{n}_z = \boldsymbol{c}/\|\boldsymbol{c}\|$. Differentiating this quotient gives

$$\partial_\xi \boldsymbol{n}_z = \frac{\partial_\xi \boldsymbol{c}}{\|\boldsymbol{c}\|} - \frac{(\boldsymbol{n}_z \cdot \partial_\xi \boldsymbol{c})}{\|\boldsymbol{c}\|}\boldsymbol{n}_z = \frac{1}{\|\boldsymbol{c}\|}\left(\mathbf{I} - \boldsymbol{n}_z \boldsymbol{n}_z^\top\right)\partial_\xi \boldsymbol{c}, \tag{14}$$

the component of $\partial_\xi \boldsymbol{c}$ tangent to the surface. Compare (14) with (13) for $i = z$ and use $\boldsymbol{n}_z \times (\boldsymbol{n}_z \times \boldsymbol{v}) = -(\mathbf{I} - \boldsymbol{n}_z \boldsymbol{n}_z^\top)\boldsymbol{v}$. This identifies the two in-plane components of $\partial_\xi \boldsymbol{\phi}$ as $\boldsymbol{n}_z \times \partial_\xi \boldsymbol{c}/\|\boldsymbol{c}\|$. That term turns the triad about axes lying in the surface, so it is the roll and pitch part. It says nothing about the component along $\boldsymbol{n}_z$, because turning about $\boldsymbol{n}_z$ does not move $\boldsymbol{n}_z$.

That remaining component is fixed by the tangent column. From $\boldsymbol{n}_x = \partial_s \boldsymbol{r}_b/\|\partial_s \boldsymbol{r}_b\|$ and (13) for $i = x$,

$$\partial_\xi \boldsymbol{\phi} \cdot \boldsymbol{n}_z = \left(\boldsymbol{n}_x \times \partial_\xi \boldsymbol{n}_x\right) \cdot \boldsymbol{n}_z = \frac{(\partial_\xi \partial_s \boldsymbol{r}_b) \cdot \boldsymbol{n}_y}{\|\partial_s \boldsymbol{r}_b\|}, \tag{15}$$

where the normalization factor drops out because only the part of $\partial_\xi \partial_s \boldsymbol{r}_b$ orthogonal to $\boldsymbol{n}_x$ survives the projection onto $\boldsymbol{n}_y$. This is the turn about the surface normal, so it is the yaw part. Adding the two contributions and setting ξ to s and then to w gives

$$\partial_s \boldsymbol{\phi} = \frac{\boldsymbol{n}_z \times \partial_s \boldsymbol{c}}{\|\boldsymbol{c}\|} + \frac{(\partial_{ss} \boldsymbol{r}_b) \cdot \boldsymbol{n}_y}{\|\partial_s \boldsymbol{r}_b\|}\boldsymbol{n}_z, \tag{16}$$

$$\partial_w \boldsymbol{\phi} = \frac{\boldsymbol{n}_z \times \partial_w \boldsymbol{c}}{\|\boldsymbol{c}\|} + \frac{(\partial_{sw} \boldsymbol{r}_b) \cdot \boldsymbol{n}_y}{\|\partial_s \boldsymbol{r}_b\|}\boldsymbol{n}_z. \tag{17}$$

The required derivatives of $\boldsymbol{c}$ follow from the product rule,

$$\partial_s \boldsymbol{c} = (\partial_{ss} \boldsymbol{r}_b) \times \partial_w \boldsymbol{r}_b + \partial_s \boldsymbol{r}_b \times (\partial_{sw} \boldsymbol{r}_b), \tag{18}$$

$$\partial_w \boldsymbol{c} = (\partial_{sw} \boldsymbol{r}_b) \times \partial_w \boldsymbol{r}_b + \partial_s \boldsymbol{r}_b \times (\partial_{ww} \boldsymbol{r}_b), \tag{19}$$

which combined with (3)–(4) only require the spline values up to the third s-derivative ($\partial_{sss} r_x$, $\partial_{sss} r_y$), the second s-derivative of the height channels ($\partial_{ss} r_z, \partial_{ss} z_k$) and the lateral derivatives of $z(s,w)$ up to second order ($k \in \{1,2,3\}$ for z_k).

3.3 Time-to-space transformation for minimum lap-time problems

A minimum lap-time problem is naturally posed with the arc-length coordinate s rather than physical time t as the independent variable, so that the integration domain $s \in [0, L]$ is fixed and the lap time $T = \int_0^L \mathrm{d}t$ becomes the cost (L is the track length). The change of variable rests on the single scalar relation

$$\frac{\mathrm{d}t}{\mathrm{d}s} = \frac{1}{\dot{s}}, \qquad \dot{s} = \frac{v_t \cos\chi - v_n \sin\chi}{1 - \kappa(s)\, w}, \tag{20}$$

where (v_t, v_n) are the chassis velocity components in the track triad, χ the chassis yaw relative to the reference-line tangent, and $\kappa(s)$ the planar curvature of the reference line. Every other state derivative $\dot{\boldsymbol{x}}$ in the vehicle model is then rescaled as $\mathrm{d}\boldsymbol{x}/\mathrm{d}s = \dot{\boldsymbol{x}}/\dot{s}$ before being handed to the

DOI 10.3384/ECP219 196

collocation transcription. The factor $\dot{s}$ is therefore the single quantity through which the road geometry enters the time-to-space transformation.

The present joint exports $\dot{s}$ as the state variable `vsTrack = der(sTrack)`, declared with `StateSelect.always` so that any Modelica tool retains it as a continuous output of the FMU into which the host vehicle model and the joint are exported together. The FMIOPT transcription reads this variable directly from the exported FMI 3.0 interface and uses it to rescale every other state derivative without any further model modification. It supplies the one quantity the time-to-space transformation needs, which makes it a prerequisite for the minimum lap-time application reported in Section 5.

3.4 D'Alembert constraint

Rather than introducing two Lagrange multipliers for the orthogonal directions to $\mathscr{M}$, we close the kinematic chain by setting the two generalized forces conjugate to (s,w) to zero. Resolving the body-frame wrench $(\boldsymbol{f}_b, \boldsymbol{\tau}_b)$ on the `frame_b` connector to the world orientation, $\boldsymbol{f}_b^w = \mathbf{R}_b^\top \boldsymbol{f}_b, \boldsymbol{\tau}_b^w = \mathbf{R}_b^\top \boldsymbol{\tau}_b$, the principle of virtual work gives

$$0 = \boldsymbol{f}_b^w \cdot \partial_s \boldsymbol{r}_b + \boldsymbol{\tau}_b^w \cdot \partial_s \boldsymbol{\phi}, \tag{21}$$

$$0 = \boldsymbol{f}_b^w \cdot \partial_w \boldsymbol{r}_b + \boldsymbol{\tau}_b^w \cdot \partial_w \boldsymbol{\phi}. \tag{22}$$

Equations (21)–(22) provide the two algebraic equations needed to close the index-1 DAE in $(\dot{s}, \dot{w})$ once the vehicle dynamics couple the wrench to the states. They are linear in the six wrench components $(\boldsymbol{f}_b^w, \boldsymbol{\tau}_b^w)$ and can be written as $\mathbf{A}\,[\boldsymbol{f}_b^w; \boldsymbol{\tau}_b^w] = \mathbf{0}$ with the 2×6 matrix

$$\mathbf{A} = \begin{bmatrix} (\partial_s \boldsymbol{r}_b)^\top & (\partial_s \boldsymbol{\phi})^\top \\ (\partial_w \boldsymbol{r}_b)^\top & (\partial_w \boldsymbol{\phi})^\top \end{bmatrix}. \tag{23}$$

The DAE is solvable when $\operatorname{rank}\mathbf{A} = 2$, and that depends only on the surface geometry.

Proposition 1. $\operatorname{rank}\mathbf{A} = 2$ *if and only if* $\partial_s \boldsymbol{r}_b$ *and* $\partial_w \boldsymbol{r}_b$ *are linearly independent, i.e. if and only if* $\|\partial_s \boldsymbol{r}_b \times \partial_w \boldsymbol{r}_b\| \neq 0$.

Proof. The first three columns of $\mathbf{A}$ are $[\partial_s \boldsymbol{r}_b, \partial_w \boldsymbol{r}_b]^\top$. If these two vectors are independent that block already has rank two, so $\operatorname{rank}\mathbf{A} = 2$. Conversely, suppose $\partial_w \boldsymbol{r}_b = \lambda\, \partial_s \boldsymbol{r}_b$ for some $\lambda \in \mathbb{R}$. Then $\boldsymbol{c} = \mathbf{0}$, the triad is undefined, and by (16)–(17) the rate vectors degenerate to $\partial_w \boldsymbol{\phi} = \lambda\, \partial_s \boldsymbol{\phi}$ as well, so the second row of $\mathbf{A}$ is λ times the first and $\operatorname{rank}\mathbf{A} \leq 1$. □

By (8) the condition in Proposition 1 fails exactly on the fold (9). So the Frenet fold reported in (Verulkar et al. 2026) is more than an awkward piece of geometry. It is the configuration in which the joint's constraint Jacobian loses rank and the DAE can no longer be solved. Condition (10) on the input splines is what keeps $\mathbf{A}$ at full rank across the whole drivable strip.

3.5 Compact form of the implementation

Listing 1 reproduces the equation block of the Modelica component. It evaluates the spline channels at the current s, expands the Monge cubic in Horner form, assembles the surface partials (3)–(4) and their parametric derivatives, builds the triad and the parametric angular velocity vectors (16)–(17), and asserts the constraints (21)–(22).

Listing 1. Equation block of `TrackJoint`. The reference line and height channels are evaluated by a piecewise spline function bound to the current segment index `ix`. The Monge cubic and its partial derivatives are computed in Horner form. The surface triad and the parametric rotation-rate vectors follow directly, and the d'Alembert relations close the 2-DOF DAE without Lagrange multipliers.

```
// Height z(s,w) in Horner form and
    partial derivatives
z = rz + wTrack*(z1 + wTrack*(z2 + wTrack
    *z3));
dzds = drzds + wTrack*(dz1ds + wTrack*(
    dz2ds + wTrack*dz3ds));
dzdw = z1 + wTrack*(2*z2 + wTrack*(3*z3))
    ;
d2zds2 = d2rzds2 + wTrack*(d2z1ds2 +
    wTrack*(d2z2ds2 + wTrack*d2z3ds2));
d2zdsdw = dz1ds + wTrack*(2*dz2ds +
    wTrack*(3*dz3ds));
d2zdw2 = 2*z2 + wTrack*(6*z3);

// Surface position and partial
    derivatives
rTrack = {rx - wTrack*dryds, ry + wTrack*
    drxds, z};
drds = {drxds - wTrack*d2ryds2, dryds +
    wTrack*d2rxds2, dzds};
drdw = {-dryds, drxds, dzdw};
d2rds2 = {d2rxds2 - wTrack*d3ryds3,
    d2ryds2 + wTrack*d3rxds3, d2zds2};
d2rdsdw = {-d2ryds2, d2rxds2, d2zdsdw};
d2rdw2  = {0, 0, d2zdw2};

// Track triad: tangent, normal, binormal
nxTrack = normalize(drds);
nzTrack = normalize(cross(drds, drdw));
nyTrack = cross(nzTrack, nxTrack);

// Parametric angular-rate vectors (
    compact triad transport)
cTrack = cross(drds, drdw);
dcTrackds = cross(d2rds2,  drdw) + cross(
    drds, d2rdsdw);
dcTrackdw = cross(d2rdsdw, drdw) + cross(
    drds, d2rdw2);
dphids = cross(nzTrack, dcTrackds)/length
    (cTrack) + ((d2rds2 *nyTrack)/length(
    drds))*nzTrack;
dphidw = cross(nzTrack, dcTrackdw)/length
    (cTrack) + ((d2rdsdw*nyTrack)/length(
    drds))*nzTrack;

// Frame_b angular velocity from
    parametric rates
derPhiTrack = dphids*vsTrack + dphidw*
    vwTrack;
```


DOI 10.3384/ECP219 196

```
// D'Alembert: virtual work in s and w
    directions
0 = fFrameB*drds + tFrameB*dphids;
0 = fFrameB*drdw + tFrameB*dphidw;

// Frame_b: position, rotation matrix,
    angular velocity
frame_b.r_0 = rTrack;
frame_b.R.T = transpose([nxTrack, nyTrack
    , nzTrack]);
frame_b.R.w = frame_b.R.T*derPhiTrack;
```

Several design choices in the listing are worth noting.

- **Ground-plane lateral direction.** The lateral offset direction is the ground-plane normal $\{-\partial_s r_y, \partial_s r_x, 0\}$ to the reference-line tangent, so the orientation of `frame_b` is determined entirely by the spline first and second s-derivatives. No additional yaw spline is needed and there is no quaternion or Euler-angle integration in the joint.

- **Horner evaluation.** The Monge cubic and its s- and w-partials are evaluated in Horner form. Horner's method is the backward-stable way to evaluate a polynomial, so it keeps the relative error small in the high-order derivatives that feed the surface triad and the acceleration-level constraint. It is not the cheapest form here, though. The Modelica tool expands integer powers and then shares common subexpressions across the local coordinate that all channels have in common, so an explicitly expanded polynomial compiles to faster code. Section 5.3 measures both effects. We keep Horner form for the conditioning of the derivative chain, not for speed.

- **D'Alembert constraint.** The two generalized forces in the (s,w) directions are set to zero by projecting the resolved body-frame wrench $(\boldsymbol{f}^b, \boldsymbol{\tau}^b)$ onto the virtual displacements $\partial_s \boldsymbol{r}_b$ and $\partial_w \boldsymbol{r}_b$. This is equivalent to the principle of virtual work and closes the 2-DOF DAE in the form expected by a Modelica DAE solver without introducing auxiliary multiplier variables.

- **C^4 continuity requirement.** The DAE solver differentiates the angular-velocity equation to form the acceleration-level constraint, which requires the second s-derivatives of every spline channel and the first mixed sw-derivatives. These exist because the export step of the companion fitting pipeline (Verulkar et al. 2026) enforces C^4 continuity at every interior knot and at the periodic wrap.

4 Implementation and FMI export

4.1 Parameter record and spline storage

All road geometry enters `TrackJoint` through a single parameter record. Each of the six channels $(r_x, r_y, r_z, z_1, z_2, z_3)$ is stored as a coefficient matrix of size $N_{\text{seg}} \times (d+1)$ with $d = 5$, plus a single break vector $\{s_i\}_{i=0}^{N_{\text{seg}}}$ shared by all channels. The break vector is uniform in s in the present implementation, which lets the break segment index ix lookup at a query s be done in constant time using a custom Modelica function. For nonuniform breaks, the same lookup can be replaced by a binary search without any change to the rest of the joint equations. Once ix is known, every spline evaluation is

$$y^{(q)}(s) = \sum_{j=q}^{d} \frac{j!}{(j-q)!}\, a_{\text{ix},j}\, (s - s_{\text{ix}})^{j-q}, \qquad (24)$$

performed in Horner form through a custom Modelica function, with the requested derivative order $q \in \{0,1,2,3\}$ passed as an argument. The function is declared locally with the common segment index ix, s, the polynomial degree and the break vector partially applied, so each per-channel evaluation call only selects its coefficient matrix and its derivative order.

4.2 State selection and initialization

The joint declares the four state variables $(s, \dot{s}, w, \dot{w})$ with `StateSelect.always` so that a Modelica compiler chooses them as continuous states regardless of the surrounding vehicle topology, eliminating state-aliasing failures observed when the chassis floor was otherwise free in $\boldsymbol{e}_z$. Initial values are exposed as parameters and propagated to the `start` attribute of the corresponding states. A boolean override is provided to relax the state selection in cases where the user prefers the tool to choose.

4.3 MultiBody integration

The joint exposes a single MultiBody `Frame_b` connector (from `Modelica.Mechanics.MultiBody.Interfaces`). A `Connections.root(frame_b.R)` statement establishes the joint as the root of the orientation graph of the kinematic subtree it heads, mirroring the pattern used by the standard MultiBody joints. The body-frame wrench arriving on the connector is rotated to world orientation by the inverse of the triad,

```
fFrameB = resolve1(frame_b.R, frame_b.f);
tFrameB = resolve1(frame_b.R, frame_b.t);
```

and used in the d'Alembert relations (21)–(22).

In a complete vehicle assembly the joint never carries the chassis on its own. The `frame_b` of `TrackJoint` is followed by a standard `Modelica.Mechanics.MultiBody.Joints.Prismatic` heave joint oriented along the surface *binormal* $\boldsymbol{n}_z$, and by three revolute joints that realize roll, pitch and yaw of the chassis *relative to the track triad* rather than relative to the world frame. Note that $\boldsymbol{n}_z$ is the vertical, out-of-plane direction of the patch. It is not the in-surface lateral normal $\boldsymbol{n}_y$, along which the joint coordinate w is already measured. The chassis therefore retains its full six degrees of freedom, but each one is naturally aligned with the road: the

DOI 10.3384/ECP219 196

heave coordinate is the vertical ride height above the patch along $\boldsymbol{n}_z$, and the three angles are the chassis attitude as a driver would describe it. This decomposition is what enables the initialization strategy discussed in Section 5.

4.4 FMI 3.0 export

In use, the joint is connected inside Dymola to a Modelica MultiBody vehicle model and the closed system is exported as a single FMI 3.0 model exchange FMU. The joint contributes the four continuous states $(s, \dot{s}, w, \dot{w})$ to the exported state vector; the heave prismatic along the surface binormal $\boldsymbol{n}_z$ (vertical, out-of-plane) and the three track-relative roll/pitch/yaw revolute joints stacked on top of `frame_b` contribute a further four position states and their derivatives, so that the chassis exposes its six degrees of freedom relative to the road surface. The remaining states, inputs and outputs of the FMU are those of the host vehicle model. The spline coefficient matrices of the joint appear as `tunable` parameters of the combined FMU so that a single FMU binary can be re-targeted to a new circuit at run-time without re-compilation. This is the form consumed by the FMIOPT direct-collocation transcription described in Section 5.

5 Results

We illustrate the use of `TrackJoint` in two settings.

5.1 Time-domain simulation

A Modelica experiment is constructed as shown in Figure 2 in order to validate the track surface produced by `TrackJoint`. In this experiment, a point-mass is driven around the Singapore Grand Prix Circuit along the minimum curvature line from the companion pipeline (Verulkar et al. 2026) with the joint as the only constraint. The resultant road surface is compared against the meshed terrain from the LiDAR point cloud of the Singapore Grand Prix Circuit using rFPro Terrain Server.

Figures 3 and 4 show the simulation results from Dymola 2026x Refresh 1, which compare the reconstructed track surface geometry produced by the joint with the 3D mesh road geometry from the LiDAR point cloud. The reconstructed surface at $\boldsymbol{r}_b(s, w)$ closely tracks the 3D mesh road geometry in absolute height, grade and bank angles across the lap, which confirms that the joint faithfully reproduces the measured road surface in time-domain simulation.

5.2 Minimum lap-time direct collocation

The multibody vehicle-plus-track-joint FMU described in Section 4.4 is then handed to the FMIOPT toolchain, which wraps it inside a CasADi direct-collocation NLP (Andersson et al. 2019) solved by IPOPT (Wächter and Biegler 2006) with the MUMPS linear solver. Because the joint exposes the parametric Jacobian factor through $\partial_s \boldsymbol{r}_b$, the standard time-to-space transformation (20) is built directly from the splines without any further model modification. On every circuit on which the splines are anchored on the minimum-curvature line of (Verulkar et al. 2026), the denominator of (20) stays bounded away from zero on the full track strip and the NLP converges to the IPOPT tolerances; on circuits on which the splines are anchored on the geometric centerline, the same NLP exhibits the convergence failure documented in the companion paper. This is what Proposition 1 means in practice. The denominator of (20) is the same factor $1 - \kappa w$ that sets the rank of the constraint Jacobian (23), so a reference line that breaks (10) hurts the transcription and the joint at the same time. Section 5.3 gives the margin for all 23 circuits and the solve statistics for Singapore.

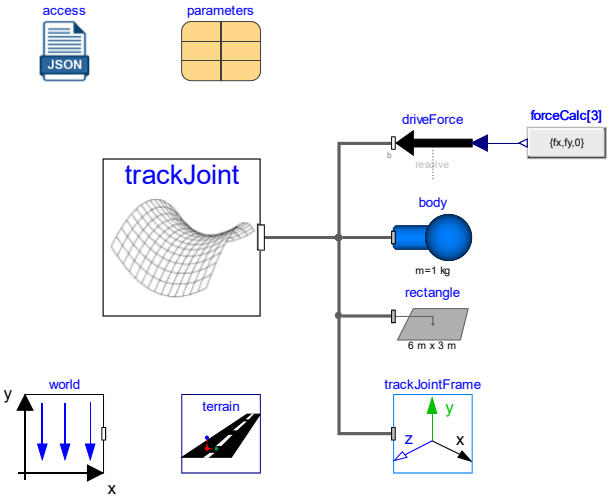


Figure 2. Modelica validation experiment of a point mass driven around the track using `TrackJoint`.

A second, equally important role of `TrackJoint` in the direct-collocation pipeline is the *initialization* of the NLP. Because the chassis degrees of freedom are expressed relative to the track triad through the heave prismatic and the three track-relative revolute joints stacked on top of `frame_b`, a feasible initial guess for an entire lap is produced by setting every chassis coordinate (heave, roll, pitch, yaw, lateral offset w) and every velocity except $\dot{s}$ to zero, and prescribing $s(\tau)$ as a linear ramp from 0 to L over the normalized time horizon $\tau \in [0, 1]$. This guess is trivial to generate, requires no preliminary simulation and no knowledge of the circuit beyond its length, yet it places the vehicle exactly on the road surface with its body axes aligned to the track triad at every collocation point. For Singapore the guess is 324 copies of the same zero vector for every chassis coordinate together with a linear ramp $s(\tau) = \tau L$, and from that IPOPT reaches `Optimal Solution Found` in 264 iterations (Table 2). It works because the track-relative decomposition makes the guess satisfy the road constraint by construction. With all chassis coordinates at zero the vehicle sits on $\boldsymbol{r}_b(s, 0)$ with $\mathbf{R}_b$ equal to the surface triad, so the two d'Alembert relations already hold at the first iterate and only the vehicle dynamics are left to resolve. In our experience this is what al-

DOI 10.3384/ECP219 196

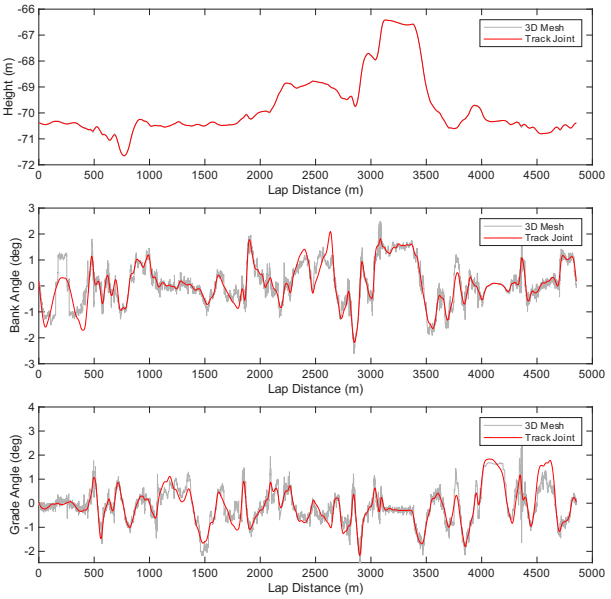


Figure 3. Validation of the `TrackJoint` surface output against the measured track geometry of the Singapore Grand Prix Circuit from LiDAR point cloud.

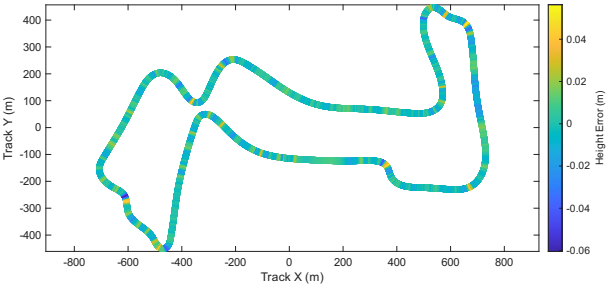


Figure 4. Path of the point mass driven through the `TrackJoint` and the error in absolute height against the 3D mesh track surface of the Singapore Grand Prix Circuit constructed from LiDAR point cloud.

lows IPOPT to take its first steps without running into the orientation-singular regions that plague world-frame chassis parametrizations, and it is the practical reason why the same NLP converges on every circuit fitted by the companion pipeline.

Figures 5 and 6 show the result of a minimum lap-time optimization problem of a full-detail multibody vehicle model driven around the Singapore Grand Prix Circuit through `TrackJoint`. Figure 6 plots the optimized racing line on the track strip generated by the joint splines, and Figure 5 the corresponding optimized velocity profile along the lap. Both quantities are produced in a single converged solve of the FMIOPT direct-collocation NLP, illustrating that the `TrackJoint` component yields a well-conditioned NLP suitable for full-circuit minimum lap-time optimization of a detailed multibody vehicle.

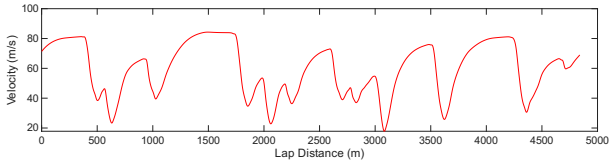


Figure 5. Optimized velocity profile of a multibody vehicle model driven around the Singapore Grand Prix Circuit using `TrackJoint` in a FMIOPT minimum lap-time problem.

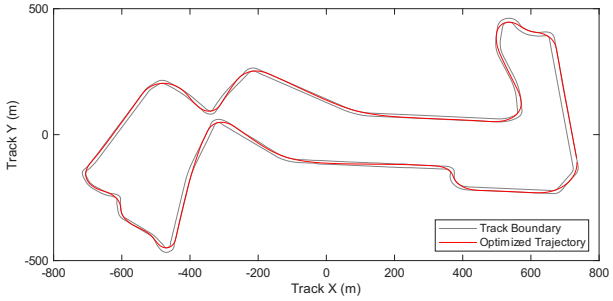


Figure 6. Optimized racing line of a multibody vehicle model driven around the Singapore Grand Prix Circuit using `TrackJoint` in a FMIOPT minimum lap-time problem.

5.3 Quantitative benchmarks

All measurements below were taken on one desktop workstation with Dymola 2026x Refresh 1. Timings are medians of five repetitions.

Circuit coverage. The joint has been run on the 23 Formula 1 circuits of the 2026 calendar. Every circuit is fitted with a uniform 5 m knot spacing, which gives 657–1387 quintic segments for track lengths of 3283–6934 m. A complete spline parameter record is about 1.2 MB of JSON. Table 1 lists the extremes. The injectivity margin (10) is above one on *all* 23 circuits. The tightest is 1.03 at Monaco, where the reference line comes down to a 9.3 m radius. The set covers both very tight corners (Monaco, Singapore) and steep camber (Zandvoort, up to 16.2° of bank).

Per-step evaluation cost. To measure the cost of the joint on its own, a unit point mass was driven one full lap through it with a fixed-step fourth-order Runge–Kutta integrator over 180,000 steps, so that every variant makes exactly 720,000 model evaluations. The released Horner version costs 3.54 s of integration CPU time, which is 4.9 μs per model evaluation for the whole 2-DOF joint including the triad and both rotation-rate vectors. Writing every polynomial out in expanded power form, and changing nothing else, gives bit-identical trajectories for 2.63 s, or 26% less. The expanded form wins because Dymola turns integer powers into repeated multiplications and then shares $x^2, \ldots, x^5$ across all eight channels and all four derivative orders. The Horner recursion instead gives each of the 26 evaluations its own chain of dependent multiply–adds. We keep Horner form for the accu-

DOI 10.3384/ECP219 196

Table 1. Geometry of the fitted circuits: track length L, number of quintic segments, minimum reference-line radius, maximum bank angle and injectivity margin $\min_s 1/(|\kappa|b)$. Extremes of the 23-circuit set.

Circuit	L [m]	N_{seg}	$R_{\min}$ [m]	bank [°]	margin
Monaco	3283	657	9.3	5.2	1.03
Zandvoort	4185	837	31.2	16.2	2.32
Singapore	4858	972	23.0	2.1	1.56
Spa	6934	1387	22.5	4.6	1.42
Silverstone	5798	1160	41.2	1.9	2.38
all 23	3283	657	9.3	1.5	**1.03**
(range)	6934	1387	41.2	16.2	2.89

Table 2. Singapore minimum lap-time NLP with the two FMU builds, which differ only in the spline evaluation function. Identical track, setup, guess, mesh and solver options.

	Horner	expanded
IPOPT status	optimal	optimal
Iterations	264	249
IPOPT time [s]	464.4	442.5
Final NLP error	8.0×10^{-4}	3.8×10^{-4}
Lap time [s]	90.6462	90.6429

racy reason below, not for speed.

Evaluation accuracy. We evaluated both forms in double precision at every segment of the Singapore fit and compared them against a 60-digit reference evaluation of the same polynomial. Over 972 segments, all six channels and derivative orders 0–3, the largest relative error is 2.6×10^{-12} for Horner and 5.0×10^{-12} for the expanded form. So Horner is at best a factor 1.9 more accurate, and on some channels the order reverses. Both sit far below any tolerance that matters physically at this knot spacing. The difference only starts to count for much coarser knots.

FMU size and NLP performance. The vehicle-plus-joint model exports to an FMI 3.0 model-exchange FMU with a 35.5 MB shared library (10.9 MB archived). On the Singapore circuit the transcription uses 323 intervals (324 collocation points) and the solve is warm-started only by the trivial guess described above. Table 2 reports the back-to-back result for the two FMUs, which differ *only* in the body of the spline evaluation function; both consumed byte-identical problem data. Both reach `Optimal Solution Found`, and the lap times agree to 3.3 ms out of 90.6 s, a relative difference of 4×10^{-5}. The iteration counts differ by 6%. We read that as the sensitivity of a nonconvex interior-point path to perturbations of order 10^{-12}, not as a real advantage either way. On this evidence the choice of evaluation form makes no measurable difference to convergence.

The FMU, not the linear algebra, dominates the cost of a solve. Of the 464 s total, 167 s go into 266 objective-gradient evaluations (627 ms each), 120 s into 389 constraint evaluations (310 ms each) and 23 s into 266 constraint-Jacobian evaluations (85 ms each). Making the road model cheaper to evaluate therefore pays off directly, which is why the 26% figure above is worth having.

Table 3. Claytex `Rail` versus `TrackJoint`.

	`Rail`	`TrackJoint`
Free DOF	1 (s)	2 (s, w)
Geometry	space curve	Monge-patch surface
Orientation	rail tangent or 3 angle tables	analytic triad from surface partials
Input data	xyz or $xyz\phi$ sample table	6 quintic splines + shared knots
Constraint	1 projection	2 d'Alembert relations
Camber/crown	Constant (centerline)	$z(s, w)$ polynomial

Comparison with the Claytex `Rail` joint. Table 3 sets the present component beside the 1-DOF rail joint (Briant 2021). The differences are about what the two components can represent. `Rail` holds a body on a curve and has to be told its orientation, either by tying it to the tangent or through three extra angle tables. `TrackJoint` holds a body on a surface and works the whole orientation triad out from the surface partials. Timing the two against each other would not mean much, since they solve different problems. The point is that the second degree of freedom, the analytic triad and the second d'Alembert relation all come from six scalar splines and no extra tabulated data.

6 Limitations

There are limits to what the component does today, and to what the measurements above can show.

Dependence on the fitting pipeline. The joint uses splines it does not produce, and it cannot tell at run time that a reference line was badly anchored. If (10) is broken, the constraint Jacobian quietly loses rank in the offending corner. A cheap guard would be to export $\min_s 1/(|\kappa|b)$ with the splines and assert on it at initialization. The present version does not do this.

Validation scope. We validated the surface against LiDAR terrain for one circuit and one point-mass run, and the optimization results use one vehicle, one setup and one initial guess per circuit. We have not run a spline-resolution sensitivity study or a multi-seed convergence study, so our comments about robustness are experience rather than measurement.

Availability. `TrackJoint` is proprietary. We give the formulation in full here so that it can be reimplemented, but the component, the vehicle models and the fitted circuit data are not public.

Implementation restrictions. The break vector is uniform, so non-uniform knots would need a search in place of the constant-time index lookup. The joint assumes a closed circuit and reduces s modulo the lap length. It exposes one `frame_b`, so four-wheel contact-patch queries need four instances sharing one parameter record.

7 Conclusions

Packaging a racetrack surface as an ordinary MultiBody joint works well in Modelica. `TrackJoint` exposes

DOI **10.3384/ECP219 196**

only a `frame_b` connector and two joint coordinates, so it plugs into the standard library like any elementary joint, and the whole road model comes down to six scalar splines and a shared knot vector.

Three things came out of this work. First, the regularity of the patch is not a modelling detail. It is the condition under which the joint can be solved at all: Proposition 1 shows the constraint Jacobian loses rank exactly on the Frenet fold, so $\min_s 1/(|\kappa|b) > 1$ on the input splines is what makes the DAE well posed. All 23 circuits fitted here meet it, the tightest being 1.03 at Monaco. Second, writing the chassis coordinates relative to the track triad makes the minimum lap-time problem easy to start. Setting every chassis coordinate to zero and ramping s puts the vehicle on the road with its axes aligned at every collocation point, with no preliminary simulation. Third, about two thirds of a 464 s Singapore solve happens inside the FMU. The 26% gap we measured between two mathematically identical polynomial forms is a reminder that what counts is the generated code, not the operation count on paper.

The next steps come straight out of Section 6: assert the injectivity margin at initialization, find out what the cubic lateral term buys on a circuit that needs it, swap the uniform-knot storage for an adaptive one, and build a contact-patch variant in which four instances share one parameter record.

Acknowledgements

The component builds on the geometric formulation of Limebeer and van de Merwe (2025), with input splines from the companion fitting work (Verulkar et al. 2026). We gratefully acknowledge the Modelica Standard Library MultiBody package, CasADi (Andersson et al. 2019) and IPOPT (Wächter and Biegler 2006).

References

Andersson, J. A. E., J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl (2019). "CasADi – a software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11.1, pp. 1–36. DOI: 10.1007/s12532-018-0139-4.

Braghin, F., F. Cheli, S. Melzi, and E. Sabbioni (2008). "Race driver model". In: *Computers & Structures* 86.13–14, pp. 1503–1516. DOI: 10.1016/j.compstruc.2007.04.028.

Briant, David (2021). *Universal Rail Joint*. https://www.claytex.com/tech-blog/universal-rail-joint/. Accessed 31 May 2026. Claytex.

do Carmo, M. P. (1976). *Differential Geometry of Curves and Surfaces*. Englewood Cliffs, NJ: Prentice Hall.

Hall, Andrew, Chad Schmitke, and John McPhee (2014-08). "Symbolic Formulation of a Path-Following Joint for Multibody Dynamics". In: *Proceedings of the ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE)*. Volume 3: 16th International Conference on Advanced Vehicle Technologies, V003T01A025. DOI: 10.1115/DETC2014-35082.

Heilmeier, Alexander, Alexander Wischnewski, Leonhard Hermansdorfer, Johannes Betz, Markus Lienkamp, and Boris Lohmann (2020). "Minimum curvature trajectory planning and control for an autonomous race car". In: *Vehicle System Dynamics* 58.10, pp. 1497–1527. DOI: 10.1080/00423114.2019.1631455.

Limebeer, D. J. N. and C. V. van de Merwe (2025). "Another look at 3D road surface modelling". In: *Vehicle System Dynamics*. Published online 7 July 2025. DOI: 10.1080/00423114.2025.2525240.

Lovato, S., M. Massaro, and D. J. N. Limebeer (2021). "Curved-ribbon-based track modelling for minimum lap-time optimisation". In: *Meccanica* 56.8, pp. 2139–2152. DOI: 10.1007/s11012-021-01387-3.

Perantoni, G. and D. J. N. Limebeer (2015). "Optimal control of a formula one car on a three-dimensional track—Part 1: Track modelling and identification". In: *ASME Journal of Dynamic Systems, Measurement, and Control* 137.5, p. 051018. DOI: 10.1115/1.4028253.

Verulkar, A., B. Danaei, S. Sivaramakrishnan, and R. Trinter (2026). "A Singularity Avoiding Track Model for Race Car Optimization". In: *Proceedings of the North American Modelica and FMI Conference*. Atlanta, GA, USA.

Wächter, A. and L. T. Biegler (2006). "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Mathematical Programming* 106.1, pp. 25–57. DOI: 10.1007/s10107-004-0559-y.

DOI **10.3384/ECP219 196**

A Singularity Avoiding Track Model for Race Car Optimization

Adwait Verulkar[1] Behzad Danaei[1] Srikanth Sivaramakrishnan[1] Ryan Trinter[1]

[1]General Motors, USA, {adwait.verulkar,behzad.danaei,
srikanth.sivaramakrishnan,ryan.trinter}@gm.com

Abstract

A track fitting pipeline is presented that converts a measured racetrack point cloud into a closed, C^4 differentiable, three-dimensional surface model suitable for high-fidelity Modelica race-car optimization. The surface is a Monge patch, parametrised by a curvilinear arc-length coordinate measured along the minimum-curvature line (MCL) and a signed lateral offset, with a height polynomial of user-selectable degree capturing the cross-sectional elevation profile. The choice of MCL as the reference line is deliberate since anchoring the same surface on the geometric centre-line can cause adjacent normals to cross inside the drivable region on tight corners. The resulting non-injective parametrisation assigns multiple curvilinear addresses to the same three-dimensional point and returns invalid road geometry. The MCL is obtained from a quadratic program that minimises integrated curvature subject to track-boundary, periodicity, and jerk-regularisation constraints. By reducing total curvature while remaining within the track bounds, the MCL moves the region of parametrisation breakdown outside the drivable strip on all circuits studied, keeps the normal map injective across the full track width, and yields a more linear time-to-arc-length transformation for minimum lap-time problems. The fitted surface and the associated track widths are exported as periodic piecewise-quintic C^4 B-splines, solved as a nonlinear program. The exported splines are consumed directly by `TrackJoint`, a Modelica MultiBody component that enforces the two degree-of-freedom kinematic road constraint via d'Alembert's principle. We also describe the parts of that Modelica implementation that set the export format: how the splines are stored on a shared knot vector as tunable parameters, how a segment is found in constant time and evaluated in Horner form, why index reduction forces C^4 continuity, how the chassis is stacked relative to the track frame, and how the FMI 3.0 export is used by the CasADi/IPOPT toolchain. A companion paper (Sivaramakrishnan et al. 2026) derives the joint in full.

Keywords: Modelica, Functional Mock-up Interface (FMI), race car simulation, track modelling, Monge patch, minimum curvature line, optimal control

1 Introduction

The track model is a crucial component of minimum lap-time simulations of high-performance vehicles. The contact-patch normal load, the gravitational projections in the body-fixed frame and the kinematic constraints that keep all four wheels on the road are all expressed in terms of the road surface and its first two derivatives (Limebeer and van de Merwe 2025; Limebeer, Bastin, et al. 2023). Inaccuracies in the track model therefore propagate directly into the predicted lap time, tyre forces, suspension loads and aerodynamic behaviour, degrading the fidelity of the entire simulation.

The community has converged on a curvilinear (s,n) description for roads since the work of Cossalter et al. (1999); the recent survey by Limebeer and van de Merwe (2025) summarises the line of development. Three-dimensional refinements followed: the Frenet–Serret ribbon of Perantoni and Limebeer (2015) parametrises the road by an arc-length spine plus an orthonormal moving frame and works well on typical racing layouts but assumes lateral flatness; Lovato et al. (2021) and Limebeer, Bastin, et al. (2023) remove that assumption by superimposing a Monge-patch elevation correction on top of a Frenet ribbon; and most recently Limebeer and van de Merwe (2025) dispense with the ribbon altogether and obtain the surface as a free Monge-patch parametrisation $\boldsymbol{r}_b(s,w) = s\,\boldsymbol{e}_s^g + w\,\boldsymbol{e}_w^g + z(s,w)\,\boldsymbol{e}_z^g$ identified through an optimal control problem on the lateral polynomial coefficients. The Monge patch removes the need for Euler angles and yields the first and second fundamental forms in closed form from the partials of $z(s,w)$, but it leaves open the choice of the reference line that anchors the (s,w) frame.

The Frenet singularity

In every published instance of the curvilinear description, including (Limebeer and van de Merwe 2025), the reference line is taken as the geometric *centre-line* of the track, i.e. the average of the left and right boundary polylines. While this choice is intuitive and requires no optimisation, it has a structural defect that is fatal for a production track model, also noted by Fork (2024). Recall that the curvilinear description maps the planar parameters (s,w) to a 3D point through

$$\boldsymbol{r}_b(s,w) = \boldsymbol{r}(s) + w\,\boldsymbol{n}_g(s) + z(s,w)\,\boldsymbol{e}_z, \tag{1}$$

where $\boldsymbol{r}(s)$ is the reference polyline traced at arc length s, $\boldsymbol{t}(s) = \boldsymbol{r}'(s)$ is its unit tangent and $\boldsymbol{n}_g(s) = R_{\pi/2}\boldsymbol{t}(s)$ its unit left normal.

DOI 10.3384/ECP219 205

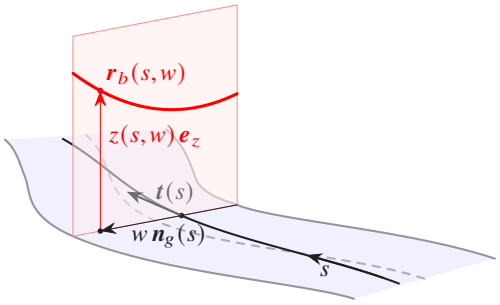


Figure 1. Three-dimensional schematic of the Monge-patch track model $\boldsymbol{r}_b(s,w) = \boldsymbol{r}(s) + w\,\boldsymbol{n}_g(s) + z(s,w)\,\boldsymbol{e}_z$. The drivable corridor is delimited by the left/right track boundaries (grey); the geometric centre-line (dashed) runs midway between them, whereas the minimum-curvature reference line $\boldsymbol{r}(s)$ (black) is free to move laterally within the bounds. The reference-line frame $(\boldsymbol{n}_g(s), \boldsymbol{e}_z)$ is drawn at a station along the line. At the marked station s the vertical cross-section plane is spanned by the lateral normal $\boldsymbol{n}_g(s)$ (the w axis) and $\boldsymbol{e}_z$; the road altitude along that section is the lateral polynomial $z(s,w) = \sum_{k=0}^{N} z_k(s)\,w^k$ (red) shows that the elevation of every surface point (s,w) above the datum plane equals the value of that polynomial.

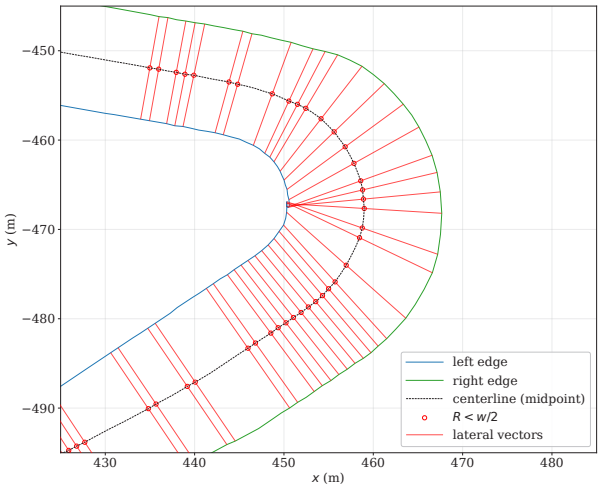


Figure 2. Geometric origin of the centre-line defect. Where the local radius of curvature $1/\kappa_c(s)$ of the reference line falls below the half-width $b(s)/2$ of the road, neighbouring lateral vectors $\boldsymbol{n}_g(s)$ intersect inside the drivable region and the Frenet map $(s,w) \mapsto \boldsymbol{r}_b(s,w)$ ceases to be injective.

The Frenet–Serret relations on the planar reference give $\boldsymbol{t}'(s) = \kappa_c(s)\,\boldsymbol{n}_g(s)$ and $\boldsymbol{n}_g'(s) = -\kappa_c(s)\,\boldsymbol{t}(s)$, with $\kappa_c(s)$ the signed curvature of the reference line. These follow directly from the tangent-angle parametrisation: writing $\boldsymbol{t}(s) = (\cos\theta(s), \sin\theta(s))$ with $\theta(s)$ the heading of the reference tangent and $\kappa_c(s) = \theta'(s)$, the left normal is $\boldsymbol{n}_g(s) = R_{\pi/2}\boldsymbol{t}(s) = (-\sin\theta, \cos\theta)$, so that

$$\boldsymbol{t}'(s) = \theta'(s)\,(-\sin\theta, \cos\theta) = \kappa_c(s)\,\boldsymbol{n}_g(s), \tag{2}$$

$$\boldsymbol{n}_g'(s) = \theta'(s)\,(-\cos\theta, -\sin\theta) = -\kappa_c(s)\,\boldsymbol{t}(s). \tag{3}$$

Figure 1 illustrates this Monge-patch construction, contrasting the geometric centre-line with the minimum-curvature reference line introduced below. Differentiating (1) yields the in-surface tangent vectors

$$\partial_s \boldsymbol{r}_b = \big(1 - \kappa_c(s)\,w\big)\boldsymbol{t}(s) + \partial_s z\,\boldsymbol{e}_z, \tag{4}$$

$$\partial_w \boldsymbol{r}_b = \boldsymbol{n}_g(s) + \partial_w z\,\boldsymbol{e}_z, \tag{5}$$

so that the first fundamental form of the Monge patch,

$$I_P = [\partial_s \boldsymbol{r}_b,\, \partial_w \boldsymbol{r}_b]^{\top}\,[\partial_s \boldsymbol{r}_b,\, \partial_w \boldsymbol{r}_b], \tag{6}$$

has Jacobian determinant

$$\begin{aligned}\sqrt{\det I_P} &= \big\|\partial_s \boldsymbol{r}_b \times \partial_w \boldsymbol{r}_b\big\| \\ &= \sqrt{\big(1 - \kappa_c(s)\,w\big)^2\big(1 + (\partial_w z)^2\big) + (\partial_s z)^2} \\ &\approx \big|1 - \kappa_c(s)\,w\big|\,\sqrt{1 + \|\nabla z\|^2}.\end{aligned} \tag{7}$$

where the longitudinal slope $\partial_s z$ is negligible compared with the lateral slope on racetrack surfaces, so that the vanishing of the Jacobian is controlled entirely by the factor $1 - \kappa_c(s)\,w$. The map (1) is a local diffeomorphism only where $\sqrt{\det I_P} > 0$, i.e. where $1 - \kappa_c(s)\,w \neq 0$. The critical value

$$w_{\text{fold}}(s) = \frac{1}{\kappa_c(s)} \tag{8}$$

is exactly the centre of curvature of $\boldsymbol{r}(s)$ at station s.

On some racetracks the centre-line, taken as a planar polyline between two surveyed boundaries, has stretches where its local radius of curvature $1/|\kappa_c(s)|$ is smaller than the local half-width $b(s)/2$ of the road, so that $|w_{\text{fold}}(s)| < b(s)/2$ and the fold sits *inside* the drivable region. At every such station the lateral coordinate frame defined by $\boldsymbol{n}_g(s)$ is no longer injective on the full width: two stations $s_1 \neq s_2$ produce laterals that intersect inside the road, as shown in Figure 2, and the Jacobian (7) vanishes at $w = w_{\text{fold}}$ and changes sign beyond it. A vehicle dynamics model that queries the joint at such a point receives multi-valued geometry and a vanishing Jacobian, and the resulting tyre-load equations are ill-posed. We have observed this failure mode repeatedly on racing circuits that contain a hairpin or a tight slow-speed sequence, such as the Singapore circuit of the Formula 1 series that is used as the running example throughout this paper.

The same defect is particularly damaging for minimum lap-time formulations because they are usually solved in the spatial domain by doing an independent-variable transformation. Following the standard change of independent variable from time t to arc length s along the reference line, the dynamics of both the vehicle and the track model are scaled by the factor $1/\dot{s}$, with

$$\dot{s} = \frac{v_x \cos\chi - v_y \sin\chi}{1 - \kappa_c(s)\,w}, \tag{9}$$

where v_x, v_y are the longitudinal and lateral velocities in the vehicle-fixed body frame and χ the heading relative

DOI **10.3384/ECP219 205**

to the reference tangent. The denominator $1-\kappa_c(s)\,w$ is the same Jacobian factor as in (7): as the trajectory approaches the fold $w = w_{\text{fold}}(s)$, $\dot{s} \to \infty$ and $1/\dot{s} \to 0$, collapsing the time transformation and producing a stiff, ill-conditioned NLP whose Hessian becomes singular at exactly the stations the optimiser is most interested in. The minimum-curvature line reduces the curvature peaks responsible for this fold and, on the processed tracks, keeps $|1-\kappa(s)\,w|$ bounded away from zero on the full track strip. This yields a well-conditioned spatial-domain transcription. Moreover, because the minimum-curvature line typically lies near a track boundary in tight corners, the NLP bounds constrain the vehicle path to have no excursion into the singular region.

A secondary effect compounds the singularity in the same corners: when w is measured from the centreline, cambered slow corners place the largest gradients of $z(s,w)$ at large $|w|$ on the side opposite the apex, and the degree-N lateral polynomial expansion of $z(s,w)$ must absorb that asymmetry with larger high-order coefficients, producing spurious wiggles in the partials $\partial_w z$ and $\partial_{ww} z$ that ultimately drive the tyre normal load.

Proposed solution

The primary contribution of this paper is a pipeline that fits a singularity-avoiding parameterisation of any track surface discretised by a point cloud, the only requirement being that the point cloud has unique and sorted indices along the longitudinal and lateral directions. The key component is the *minimum-curvature* reference line, computed via a quadratic program based on Heilmeier et al. (2020). Once the reference line is obtained, the pipeline fits the Monge patch in lateral polynomials of degree N and exports an arc-length-parameterised, C^4-continuous, periodic piecewise-quintic representation consumed by Modelica-based race-car optimisation problems.

Racing-line generation has its own large literature that the proposed pipeline draws on. Direct minimum-time optimal control (Perantoni and Limebeer 2015) provides the true optimum but is expensive and requires a vehicle model. Sequential or geometric approaches such as the minimum-curvature quadratic program of Braghin et al. (2008) and the iterated, constraint-aware formulation of Heilmeier et al. (2020) are within a fraction of a percent of the time-optimal solution while being orders of magnitude faster and vehicle-agnostic; the two-step decomposition of Kapania et al. (2016) alternates path and velocity profile. The novel contribution of this paper is to repurpose the minimum-curvature line of Heilmeier et al. (2020) and Braghin et al. (2008) not as a driving target but as the curvilinear reference that anchors the Monge-patch surface parametrisation of Limebeer and van de Merwe (2025), thereby certifying injectivity of the Frenet map on the full track strip.

The rest of the paper is organised as follows. Section 2 develops the fitting pipeline, from the minimum-curvature quadratic program to the C^4 piecewise-quintic export. Section 3 shows how the Modelica `TrackJoint` component uses those splines, and why the differential-algebraic equation it forms needs each property of the export format. Section 4 reports the effect on the conditioning of a full-circuit minimum lap-time problem.

2 Methodology

The end-to-end pipeline is organised in seven stages:

1. Parse the structured point cloud $\{(j,k,x_{jk},y_{jk},z_{jk},\boldsymbol{n}_{jk})\}$ exported by the surveyor, where j indexes the longitudinal station, k the lateral sample at that station, (x_{jk},y_{jk},z_{jk}) is the measured 3D point and $\boldsymbol{n}_{jk}$ the associated unit surface normal;

2. Extract the median-lateral reference polyline and the left/right boundaries from the point-cloud lateral indexing;

3. Solve the minimum-curvature quadratic program on the boundary polylines, iterating on the local linearisation, with periodic α-smoothing to suppress active-constraint chatter;

4. Build the Frenet frame on the resulting line and recover the lateral widths $b_{\text{left}}(s)$, $b_{\text{right}}(s)$ by ray-polyline intersection with the boundaries;

5. Re-sample the point cloud onto a longitudinal–lateral grid that is orthonormal with respect to the minimum-curvature line, i.e. aligned with the new (s,w) frame;

6. Fit, at each station, the Monge-patch elevation $z(s,w) = \sum_{k=0}^{N} z_k(s)\,w^k$ in the lateral coordinate;

7. Fit each longitudinal channel $(r_x, r_y, z_0, \ldots, z_N, b_{\text{left}}, b_{\text{right}})$ by a periodic piecewise quintic with C^4 continuity at the knots that is consumed by the Modelica-based vehicle and track model.

2.1 Surface model

The track is a smooth, regular, orientable two-dimensional manifold $\mathcal{M} \subset \mathbb{R}^3$. We parametrise it by the Monge patch (1) of (Limebeer and van de Merwe 2025), with (s,w) in a rectangular domain $\mathcal{D} \subset \mathbb{R}^2$, $\boldsymbol{r}(s)$ the reference line at arc length s, $\boldsymbol{n}_g(s)$ its unit left normal and $\boldsymbol{e}_z$ the vertical unit vector. The elevation is taken in the algebraic-function class of (Limebeer and van de Merwe 2025),

$$z(s,w) = \sum_{k=0}^{N} z_k(s)\,w^k, \tag{10}$$

truncated at a user-selectable degree N. In practice we adopt the low order $N = 3$ proposed by Limebeer and van de Merwe (2025), which is sufficient to capture camber

DOI **10.3384/ECP219 205**

and crown on every circuit we have processed while keeping the lateral derivatives $\partial_w z$ and $\partial_{ww} z$ well behaved.

For any $N \geq 2$ these lateral derivatives exist, and provided each coefficient $z_k(s)$ is a C^4 piecewise polynomial in s they are also continuously differentiable in s up to second order and beyond. Together with $\partial_s \boldsymbol{r}_b$ and $\partial_w \boldsymbol{r}_b$, which are linearly independent everywhere by construction, this provides every quantity needed for the first and second fundamental forms I_P and Π_P:

$$I_P = \begin{bmatrix} (1-\kappa_c w)^2 + (\partial_s z)^2 & \partial_s z\, \partial_w z \\ \partial_s z\, \partial_w z & 1 + (\partial_w z)^2 \end{bmatrix}, \tag{11}$$

$$\Pi_P = \begin{bmatrix} \boldsymbol{e}_m^{P\top} \partial_{ss} \boldsymbol{r}_b & \boldsymbol{e}_m^{P\top} \partial_{sw} \boldsymbol{r}_b \\ \boldsymbol{e}_m^{P\top} \partial_{sw} \boldsymbol{r}_b & \boldsymbol{e}_m^{P\top} \partial_{ww} \boldsymbol{r}_b \end{bmatrix}, \tag{12}$$

with $\boldsymbol{e}_m^P = (\partial_s \boldsymbol{r}_b \times \partial_w \boldsymbol{r}_b)/\|\partial_s \boldsymbol{r}_b \times \partial_w \boldsymbol{r}_b\|$ the unit normal. Together, I_P, Π_P and $\boldsymbol{e}_m^P$ provide the gravity projections, the kinematic constraints involving the body-fixed angular velocities ω_x^b, ω_y^b, and the closed-form normal acceleration $\ddot{\boldsymbol{r}}_b^\top \boldsymbol{e}_m^P$ of (Limebeer and van de Merwe 2025). With positive camber defined by positive lateral slope in the w direction, the local camber angle is $\phi(s,w) = \arctan(\partial_w z(s,w)) = \arctan\big(\sum_{k=1}^{N} k\, z_k(s)\, w^{k-1}\big)$.

2.2 Minimum curvature reference line

Here we replace the centre line by the minimum curvature line. The minimum curvature objective minimises the integral of $\kappa^2(s)$ over the closed loop subject to the bounds of the track. We also verify non-intersection of the recovered laterals after the line and widths are recomputed in the minimum-curvature frame. A further geometric benefit is that the line typically hugs the inside boundary through slow corners, so the admissible (s,w) excursion is effectively one-sided and the relevant $|w|$ shrinks, improving the conditioning of both the lateral polynomial fit and the spatial-domain time transformation $1/\dot{s}$ of (9).

2.2.1 Quadratic program

Following Braghin et al. (2008) and the iterative constraint-aware refinement of Heilmeier et al. (2020), candidate points along the line are written as

$$\boldsymbol{r}_i = \boldsymbol{p}_i + \alpha_i \boldsymbol{n}_i, \qquad \alpha_i \in \left[-b_{\mathrm{r},i} + \tfrac{b_v}{2},\ b_{\mathrm{l},i} - \tfrac{b_v}{2}\right], \tag{13}$$

where $\boldsymbol{p}_i$ is a reference point (initially the centre-line), $\boldsymbol{n}_i$ is its unit left normal, $b_{\mathrm{l},i}, b_{\mathrm{r},i}$ the local left/right widths recovered by ray-polyline intersection against the boundary polygons, and b_v the vehicle width. The objective is a weighted sum of squared discrete curvatures with curvature samples written as second-order finite differences. We retain the cross term in the discrete curvature expansion and omit explicit curvature constraints because the closed-loop containment check described below guarantees strict containment.

The objective is augmented with two regularisers. A length-cost $w_L \sum_i \|\boldsymbol{r}_{i+1} - \boldsymbol{r}_i\|^2$ discourages local stretching of the discretisation when the boundaries are noisy.

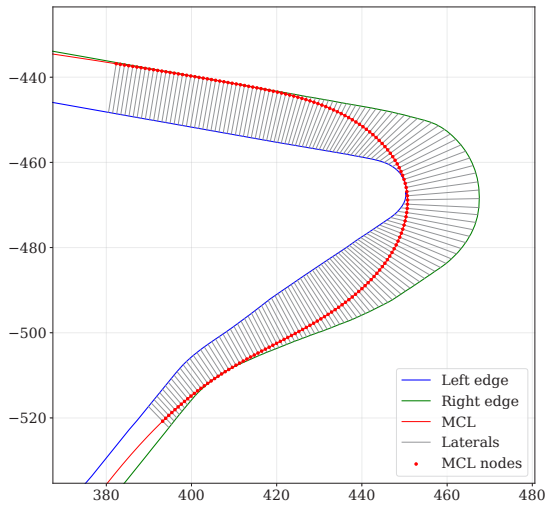


Figure 3. Result of the proposed reference-line construction on the same hairpin region as Fig. 2. The minimum-curvature line (MCL, red) stays strictly inside the track and the recovered laterals (grey) no longer intersect within the drivable region, restoring injectivity of the Frenet map.

A jerk-cost $w_J \sum_i \|\boldsymbol{r}_{i+2} - 3\boldsymbol{r}_{i+1} + 3\boldsymbol{r}_i - \boldsymbol{r}_{i-1}\|^2$ removes residual two-sample chatter at stations where the bound constraints are exactly active. To reduce sensitivity to non-uniform sampling, the second- and third-difference rows are weighted by $1/\Delta s_i^{3/2}$ and $1/\Delta s_i^{5/2}$, respectively. The resulting QP has the standard form $\min \frac{1}{2}\boldsymbol{\alpha}^\top H \boldsymbol{\alpha} + f^\top \boldsymbol{\alpha}$ subject to the margin-shrunk box constraints $-\boldsymbol{b}_{\mathrm{r}} \leq \boldsymbol{\alpha} \leq \boldsymbol{b}_{\mathrm{l}}$. Concretely, the three penalty groups are assembled into a single weighted least-squares residual $A\boldsymbol{\alpha} - \boldsymbol{c}$. We introduce the periodic circulant difference operators D_1, D_2, D_3 with stencils $(-1,1)$, $(1,-2,1)$ and $(-1,3,-3,1)$, the diagonal weight matrices $W_\kappa = \mathrm{diag}(\Delta s_i^{-3/2})$, $W_L = \sqrt{w_L}\,\mathrm{diag}(\Delta s_i^{-1/2})$ and $W_J = \sqrt{w_J}\,\mathrm{diag}(\Delta s_i^{-5/2})$, and the normal-projection diagonals $N_x = \mathrm{diag}(n_{i,x})$, $N_y = \mathrm{diag}(n_{i,y})$. With $\boldsymbol{p}_x, \boldsymbol{p}_y$ the reference-point coordinate vectors, the residual is the block-sparse stack $A^\top = [\ N_x D_2 W_\kappa, N_y D_2 W_\kappa, N_x D_1 W_L, N_y D_1 W_L, N_x D_3 W_J, N_y D_3 W_J\ ]$ and $\boldsymbol{c}^\top = -[\ \boldsymbol{p}_x^\top D_2 W_\kappa, \boldsymbol{p}_y^\top D_2 W_\kappa, \boldsymbol{p}_x^\top D_1 W_L, \boldsymbol{p}_y^\top D_1 W_L, \boldsymbol{p}_x^\top D_3 W_J, \boldsymbol{p}_y^\top D_3 W_J\ ]$, where the x and y component rows are stacked per penalty group. The Hessian and the linear term of the QP then follow as the Gauss–Newton products $H = A^\top A$ and $f = -A^\top \boldsymbol{c}$, so that H is symmetric positive semidefinite and banded by the four-point periodic stencil, and $\boldsymbol{c}$ collects the curvature, length and jerk of the current frozen reference $\{\boldsymbol{p}_i, \boldsymbol{n}_i\}$. The QP is solved with OSQP (Stellato et al. 2020).

Figure 3 shows the same hairpin region as Fig. 2, now anchored on the minimum-curvature line produced by the pipeline of Section 2.2. It can be seen that the recovered laterals no longer cross inside the drivable region.

DOI 10.3384/ECP219 205

2.2.2 Iteration, smoothing and containment

A single QP solve does *not* return the true minimum-curvature line, because the parameterisation $\boldsymbol{r}_i = \boldsymbol{p}_i + \alpha_i \boldsymbol{n}_i$ in (13) is linear in α_i only for a *frozen* reference $\{\boldsymbol{p}_i, \boldsymbol{n}_i\}$, whereas the underlying minimum-curvature problem is nonlinear in the position of the line itself. Three coupled approximations are introduced when the reference is fixed:

1. **Geometric linearisation of curvature.** The discrete curvature operator is built as a second finite difference of α_i projected onto the reference normals $\boldsymbol{n}_i$. This is a Gauss–Newton-style linearisation of $\kappa\, ds$ about the current line; the true κ depends nonlinearly on the tangent rotation between adjacent samples, which itself changes once $\boldsymbol{\alpha}$ moves the line. A single solve therefore only yields a curvature-reducing *step*.

2. **Stale arc-length and normals.** The row weights $1/\Delta s_i^{3/2}$ and $1/\Delta s_i^{5/2}$, the normals $\boldsymbol{n}_i$ and the second-difference stencil are all computed on the *previous* reference. After the line shifts, Δs_i and the local normal direction change, so the discrete operator no longer represents $\int \kappa^2\, ds$ on the updated geometry until the reference is refreshed and the arc-length resampling is repeated.

3. **Stale lateral bounds.** The half-widths $b_{\mathrm{l},i}, b_{\mathrm{r},i}$ are obtained by ray-casting from the reference $\boldsymbol{p}_i$ along $\boldsymbol{n}_i$ to the boundary polygons. Once the line has moved laterally by α_i, the ray footpoint and direction at the new station are different and the old bounds are merely an approximation; in particular, on tight corners the new line can land outside the polygon that the old bounds claimed to enclose it in (see the containment step below).

We therefore wrap the QP in a fixed-point iteration: $\boldsymbol{p}_i^{(k+1)} \leftarrow \boldsymbol{p}_i^{(k)} + \alpha_i^{(k)} \boldsymbol{n}_i^{(k)}$ followed by an arc-length resampling and recomputation of normals, Δs_i and ray-cast widths before the next solve. In practice, 3–5 iterations suffice.

This outer loop is structurally an SQP step. It's effectively a convex QP built on a linearisation of a nonlinear problem, iterated to a fixed point. We avoid a general-purpose SQP/NLP solver such as IPOPT (Wächter and Biegler 2006) because the dominant nonlinearities such as ray-polyline intersections for width computation, polygon containment of the MCL and arc-length resampling, are non-smooth operations in terms of $\boldsymbol{\alpha}$. With these quantities frozen, each subproblem is a convex QP with banded Hessian $H = A^\top A$, making OSQP a natural choice.

Because widths are computed at the previous reference, the new line can in pathological cases land slightly outside the boundary polygons. A boolean inside/outside test using `contains_points` flags such points after every solve. If any point is flagged, the iteration is rejected and the line reverts to the most recent iteration whose points were all strictly inside. If the bounds become active after the first QP solve, no previous iterate is available to fall back on, so we keep the solution and rely on the smoothing step below to remove bound-induced noise in the line and laterals. After each solve, the discrete $\boldsymbol{\alpha}$ is filtered with a short periodic centred moving average that suppresses the two-sample zig-zag arising at stations where the bound constraints are active.

2.2.3 Frenet frame and track widths

The closed minimum-curvature line is fitted with a periodic quintic B-spline $(x(s), y(s))$ in arc length. Tangents and unit left normals follow analytically:

$$\boldsymbol{t}(s) = \frac{(x'(s), y'(s))}{\|(x'(s), y'(s))\|}, \quad \boldsymbol{n}_g(s) = R_{\pi/2}\boldsymbol{t}(s). \tag{14}$$

Once the reference line is found using the logic of the previous section, the accurate lateral half-widths $b_{\mathrm{left}}(s), b_{\mathrm{right}}(s)$ are computed by ray-polyline intersection of the local lateral against both boundaries in both signs, taking the smallest positive parameter.

2.3 Re-sampling and Monge-patch fit

A regular $(n_{\mathrm{long}} \times n_{\mathrm{lat}})$ grid is generated on the minimum-curvature frame, so that the sample stations are orthonormal with respect to the reference line and lateral stations sweep $w_k \in [-b_{\mathrm{right}}(s_i), b_{\mathrm{left}}(s_i)]$, as shown in Figure 4. The elevation at every grid point (s_i, w_k) is obtained from an internal tool that retains a high-density representation of the surveyed surface and queries it at the grid stations to provide elevation values at arbitrary (x, y) locations beyond those of the original samples. The tool uses Delaunay triangulation of the surveyed points as the primary interpolant, falling back to a radial-basis-function (thin-plate-spline kernel, 50 nearest neighbours) interpolant for query points that lie outside the convex hull of the triangulation. At every longitudinal station the elevation samples $z(s_i, w_k)$ are fitted by least squares with the degree-N lateral polynomial of (10) as shown in Figure 5.

2.4 Export: C^4 piecewise quintic

The Modelica model consumes a periodic piecewise-quintic description with shared knots: each of the $N+5$ channels $(r_x, r_y, z_0, \ldots, z_N, b_{\mathrm{left}}, b_{\mathrm{right}})$ is a polynomial of degree 5 on each segment with C^4 continuity at interior knots and strict periodicity of the function value and the first four derivatives at the wrap. Quintics are needed for two reasons: $z(s, w)$ derivatives must be continuous through the expressions for I_P and Π_P (Section 2.1), and the track FMU evaluates these in real time inside a Modelica DAE solver that is sensitive to discontinuities in slope.

The fit is formulated as a smoothness-regularised periodic nonlinear program in CasADi (Andersson et al. 2019) and solved by IPOPT (Wächter and Biegler 2006) with the HSL MA57 linear solver (Duff 2004). For a fixed number of segments N_{seg} and a polynomial degree $d = 5$, the variables are the local-basis coefficients

DOI 10.3384/ECP219 205

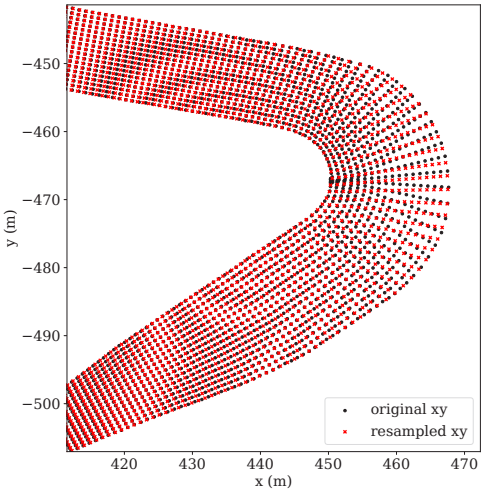


Figure 4. Regular ($n_{\text{long}} \times n_{\text{lat}}$) resampling of the surveyed point cloud on the minimum-curvature frame.

$\{a_{i,j}\}_{i=0,\ldots,N_{\text{seg}}-1;\, j=0,\ldots,d}$:

$$y_i(s) = \sum_{j=0}^{d} a_{i,j}\,(s - s_i)^j, \quad s \in [s_i, s_{i+1}]. \tag{15}$$

The objective combines the per-sample least-squares mismatch and the integrals, evaluated by 3-point Gauss–Legendre quadrature, of the squared first to fourth derivatives:

$$\frac{1}{m}\sum_{p=1}^{m}\big(y_{i(p)}(s_p) - y_p\big)^2 + \sum_{q=1}^{4} w_q \int_0^L \big(y^{(q)}(s)\big)^2\,\mathrm{d}s. \tag{16}$$

Hard C^4 continuity is imposed at every interior knot and at the wrap. For the $N+1$ elevation channels we additionally apply a zero-phase FFT low-pass filter with cutoff wavelength equal to the segment length to prevent the quintic from being asked to represent longitudinal waves in vertical height whose wavelength is comparable to the spacing of its knots, which would otherwise appear as under-resolved spline oscillations.

3 Modelica Implementation

This section describes the receiving end of the pipeline, the `TrackJoint` component. The companion paper (Sivaramakrishnan et al. 2026) gives the full derivation of the joint, its validation against the LiDAR terrain mesh and its use in time-domain simulation.

`TrackJoint` is a holonomic two-degree-of-freedom kinematic joint that realises the Monge patch (1) as a `Modelica.Mechanics.MultiBody` (Otter et al. 2003) component. Its joint coordinates are the same two curvilinear parameters used throughout this paper: the arc-length coordinate s (`sTrack`) measured along the minimum-curvature line, and the signed lateral offset w (`wTrack`). From these two coordinates and the exported channels the joint builds the surface position, the orientation triad and the angular velocity of a single `frame_b` connector, and closes the road constraint through d'Alembert's principle.

3.1 Parameter record and spline storage

Of the $N+5$ exported channels the joint consumes only the $N+3$ geometry channels $\{r_x, r_y, r_z, z_1, \ldots, z_N\}$, where $r_z \equiv z_0$ of (10); at the adopted order $N = 3$ this is the six-channel set $\{r_x, r_y, r_z, z_1, z_2, z_3\}$. The two width channels $b_{\text{left}}, b_{\text{right}}$ are not part of the kinematics and are consumed instead as the path constraint $-b_{\text{right}}(s) \le w \le b_{\text{left}}(s)$ on the joint coordinate `wTrack` in the optimal-control problem.

All geometry enters the joint through a single parameter record. Each channel is stored as the matrix of local monomial coefficients $a_{i,j}$ of Section 2.4, of size $N_{\text{seg}} \times (d+1)$ with $d = 5$, together with one break vector shared by every channel:

Listing 1. Spline storage in the `TrackJoint` parameter record: one coefficient matrix per exported channel and a single shared break vector.

```
parameter Real sBreak[nSeg + 1];
parameter Real rxCoeff[nSeg, d + 1];
parameter Real ryCoeff[nSeg, d + 1];
parameter Real rzCoeff[nSeg, d + 1];
parameter Real z1Coeff[nSeg, d + 1];
parameter Real z2Coeff[nSeg, d + 1];
parameter Real z3Coeff[nSeg, d + 1];
```

Sharing one break vector across channels is why the export step of Section 2.4 solves all channels on a common knot sequence. The segment index is then found once per integration step and reused by every channel. The export also makes the value and the first four derivatives periodic at the wrap, so the joint needs no special case at the start/finish line and simply reduces the arc-length coordinate modulo the track length L. The coefficient matrices are declared `tunable`, so one compiled FMU can be pointed at a different circuit at run time without recompiling.

3.2 Segment lookup and Horner evaluation

Because the exported breaks are uniform, the segment index at a query s is obtained in constant time,

$$\mathrm{ix} = \min\left(\left\lfloor \frac{s \bmod L}{\Delta s} \right\rfloor,\; N_{\text{seg}} - 1\right), \tag{17}$$

avoiding the branch-heavy interval search of a general B-spline evaluator. Every channel derivative is then the local monomial sum

$$y^{(q)}(s) = \sum_{j=q}^{d} \frac{j!}{(j-q)!}\, a_{\mathrm{ix},j}\,(s - s_{\mathrm{ix}})^{j-q}, \tag{18}$$

evaluated in Horner form by one Modelica function that takes the derivative order q as an argument. Both (17) and (18) are branch-free in the continuous variables, so

DOI 10.3384/ECP219 205

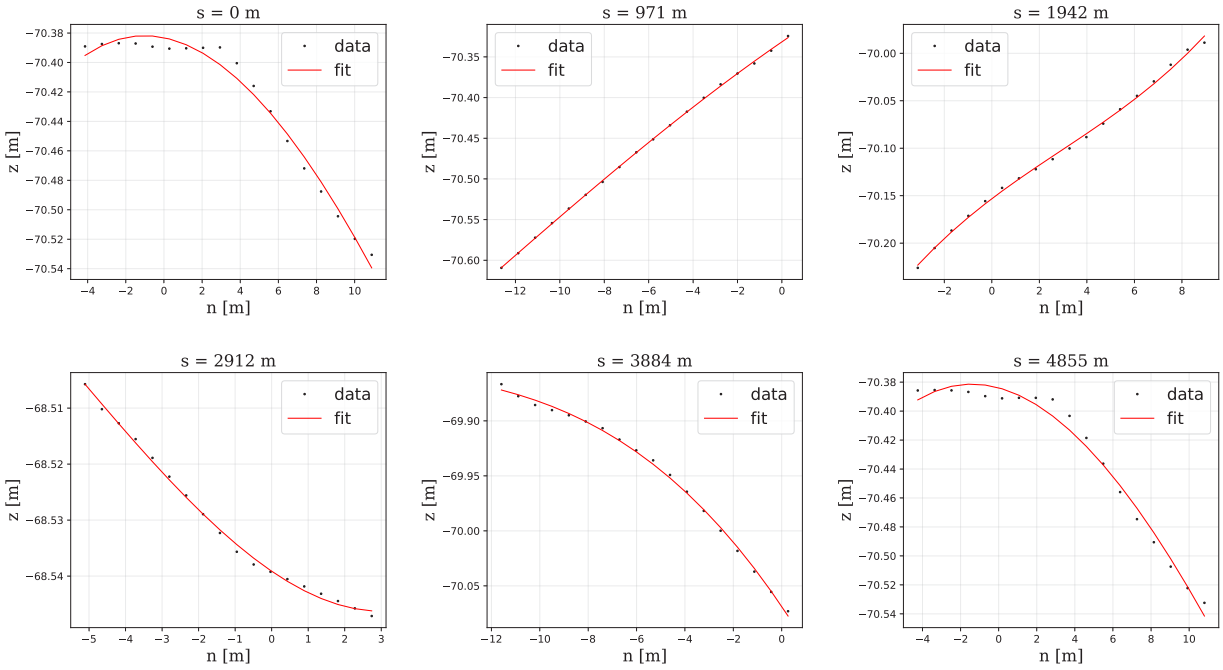


Figure 5. Lateral degree-N polynomial fit of the elevation $z(s_i, w_k)$ at each longitudinal station.

the whole joint traces to a smooth expression graph when CasADi differentiates the FMU at each collocation point.

This is also where the degree and continuity chosen in Section 2.4 come from. The joint needs $\partial_{sss} r_x$ and $\partial_{sss} r_y$ to form the parametric rotation rates, so q reaches 3. The Modelica tool then differentiates the angular-velocity equation once more during index reduction, which takes the requirement to $q = 4$. A piecewise quintic with C^4 continuity at every interior knot and at the wrap is the lowest degree that keeps everything the DAE solver differentiates continuous across a knot. A cubic spline leaves $\partial_{ss} \boldsymbol{r}_b$ jumping at each knot, which makes the integrator chatter at the segment boundaries.

3.3 Surface geometry and the d'Alembert constraint

Listing 2 reproduces the equation block that turns the evaluated channels into surface geometry. The height polynomial (10) and its partials are expanded in Horner form, the in-surface tangents $\partial_s \boldsymbol{r}_b$ and $\partial_w \boldsymbol{r}_b$ follow directly, and the triad $(\boldsymbol{n}_x, \boldsymbol{n}_y, \boldsymbol{n}_z)$ is built analytically from them, so that no Euler angles or quaternions are integrated inside the joint.

Listing 2. Core surface block of the Modelica `TrackJoint` that consumes the exported channels $\{r_x, r_y, r_z, z_1, z_2, z_3\}$: Horner height polynomial, surface position and partials, track-frame triad and d'Alembert constraint. The full component is given in (Sivaramakrishnan et al. 2026).

```
// Height z(s,w) in Horner form and partial
    derivatives
z = rz + wTrack*(z1 + wTrack*(z2 + wTrack*
    z3));
dzds = drzds + wTrack*(dz1ds + wTrack*(
    dz2ds + wTrack*dz3ds));
dzdw = z1 + wTrack*(2*z2 + wTrack*(3*z3));

// Surface position: lateral offset along
    ground-plane normal
rTrack = {rx - wTrack*dryds, ry + wTrack*
    drxds, z};

// First partial derivatives of surface
    position
drds = {drxds - wTrack*d2ryds2, dryds +
    wTrack*d2rxds2, dzds};
drdw = {-dryds, drxds, dzdw};

// Track frame triad: tangent, surface
    normal, binormal
nxTrack = normalize(drds);
nzTrack = normalize(cross(drds, drdw));
nyTrack = cross(nzTrack, nxTrack);

// D'Alembert: virtual work in s and w
    directions must vanish
0 = fFrameB*drds + tFrameB*dphids;
0 = fFrameB*drdw + tFrameB*dphidw;

// Frame_b assignment: position and
    rotation matrix
frame_b.r_0 = rTrack;
frame_b.R.T = transpose([nxTrack, nyTrack,
    nzTrack]);
```

The two constraint equations in the listing are the joint's link back to the geometry of Section 2.1. Instead of introducing Lagrange multipliers for the directions orthogonal to $\mathcal{M}$, the body-frame wrench $(\boldsymbol{f}_b, \boldsymbol{\tau}_b)$ arriv-

DOI **10.3384/ECP219 205**

ing on `frame_b` is resolved to world orientation with `resolve1` and the two generalised forces conjugate to (s, w) are set to zero,

$$0 = \boldsymbol{f}_b^w \cdot \partial_s \boldsymbol{r}_b + \boldsymbol{\tau}_b^w \cdot \partial_s \boldsymbol{\phi}, \tag{19}$$
$$0 = \boldsymbol{f}_b^w \cdot \partial_w \boldsymbol{r}_b + \boldsymbol{\tau}_b^w \cdot \partial_w \boldsymbol{\phi}, \tag{20}$$

where $\partial_s \boldsymbol{\phi}, \partial_w \boldsymbol{\phi}$ are the parametric rotation-rate vectors of the triad. These two relations close the index-1 DAE in $(\dot{s}, \dot{w})$ and are linear in the wrench components. Their 2×6 Jacobian has rank two wherever the parametrisation is regular, and the Frenet fold (8) is the configuration in which it loses rank. So the minimum-curvature reference line of Section 2.2 does more than improve the conditioning of the NLP. It is what keeps the Modelica constraint solvable at all.

3.4 State selection and MultiBody assembly

The joint declares $(s, \dot{s}, w, \dot{w})$ as `StateSelect.always` variables so that the tool keeps them as continuous states whatever the surrounding vehicle topology. Without this, state aliasing against a chassis that is otherwise free in $\boldsymbol{e}_z$ makes the dynamic state selection fail. A `Connections.root(frame_b.R)` statement makes the joint the root of the orientation graph of the subtree it heads, following the pattern of the standard MultiBody joints.

In a vehicle assembly the joint never carries the chassis alone. Stacked on `frame_b` are a standard `Modelica.Mechanics.MultiBody.Joints.Prismatic` heave joint along the surface binormal $\boldsymbol{n}_z$ and three revolute joints realising roll, pitch and yaw *relative to the track triad*. The chassis thus keeps all six degrees of freedom, but each one is expressed relative to the road rather than to the world frame.

3.5 FMI 3.0 export and use in FMIOPT

The joint and the host MultiBody vehicle model are compiled in Dymola and exported together as a single FMI 3.0 (Modelica Association 2023) model-exchange FMU. The joint contributes the four continuous states $(s, \dot{s}, w, \dot{w})$, and the stacked heave and attitude joints a further four position states and their derivatives. Two features of this FMU matter for the optimal-control application. First, $\dot{s}$ is exported as the model variable `vsTrack`, which is the factor (9) through which the road geometry enters the time-to-space transformation. FMIOPT reads it from the FMI interface and rescales every other state derivative, with no rewriting of the model. Second, the chassis coordinates are track-relative, so a feasible initial guess for a full lap comes from setting every chassis coordinate and every velocity except $\dot{s}$ to zero and prescribing s as a linear ramp from 0 to L. That puts the vehicle on the road surface with its axes aligned to the track triad at every collocation point, without any preliminary simulation.

4 Results

The geometric quality of the surface fit was characterised in the previous sections through the per-station residuals of the lateral polynomial expansion (Section 2.3) and the C^4 piecewise-quintic export (Section 2.4). We therefore focus here on the downstream consequence of that geometry: whether the resulting track model yields a well-conditioned minimum lap-time problem. The representative case reported below is the Formula 1 Singapore circuit processed by the pipeline.

The `TrackJoint` component (Section 3) is connected to the multi-body F1 vehicle model in Dymola, and the coupled system is exported as a single FMI 3.0 FMU. FMIOPT then wraps this FMU inside a CasADi (Andersson et al. 2019) direct-collocation NLP and hands the resulting problem to IPOPT (Wächter and Biegler 2006) configured with the MUMPS linear solver. The time-to-space transformation of (9) is supplied by `TrackJoint` itself. The arc-length coordinate s (`sTrack`) is a continuous state of the joint and its time derivative $\dot{s}$ (`vsTrack`) is used in the formulation of the joint. FMIOPT uses $\dot{s}$ to recast the dynamics from the time domain into the spatial domain along the reference arc length s, so the conditioning of the Jacobian factor $1 - \kappa(s)\, w$ directly governs the convergence behaviour of the resulting NLP.

The Singapore circuit contains corners whose radius is smaller than the local road width, and the same geometry has appeared on several other racing circuits processed by the pipeline. Because the centre-line parameterisation is not even well defined on these circuits, the previous version of the track parameterization had to introduce a geometric distortion of the reference by adding a small lateral offset to the centre-line at the offending stations, pushing the fold $w_{\text{fold}}(s) = 1/\kappa_c(s)$ outside the track width. With the minimum-curvature line no such distortion is required as, by construction, $\min_s 1/|\kappa(s)| > b_{\max}$.

This geometric difference appears directly in the lap-time solve. As shown in Fig. 6, the centre-line joint in the ribbon-plus-Monge-patch form of Limebeer, Bastin, et al. (2023) does not converge: the dual infeasibility oscillates by several orders of magnitude from one IPOPT iteration to the next, the primal residual stalls and IPOPT adds regularisation in later iterations. This is consistent with the numerical signature of an ill-conditioned KKT system in the same tight-corner region where the centre-line parameterisation is geometrically near its Frenet fold.

With the minimum-curvature joint, all other settings unchanged, the Frenet fold is moved outside the drivable strip. Both primal and dual residuals decrease monotonically to the IPOPT tolerances, and the NLP converges in a comparable number of iterations to problems solved on well-conditioned tracks. This makes several lap-time cases that were previously intractable solvable without modifying the vehicle model or solver settings.

The optimised solution is shown in Figs. 7 and 8. The racing line of Fig. 7 traces the lateral offset $w(s)$ of the

DOI **10.3384/ECP219 205**

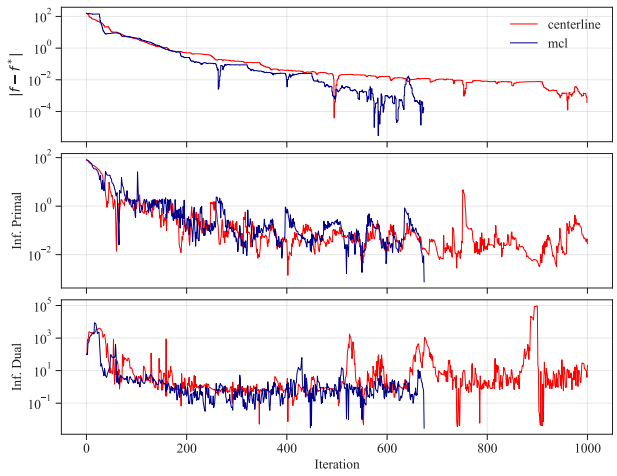


Figure 6. IPOPT convergence trace of the minimum lap-time NLP on a circuit containing a hairpin tighter than the road is wide. The non-converging case uses the centre-line-anchored track joint of Limebeer, Bastin, et al. (2023): the dual infeasibility jumps erratically across iterations and the problem fails to converge.

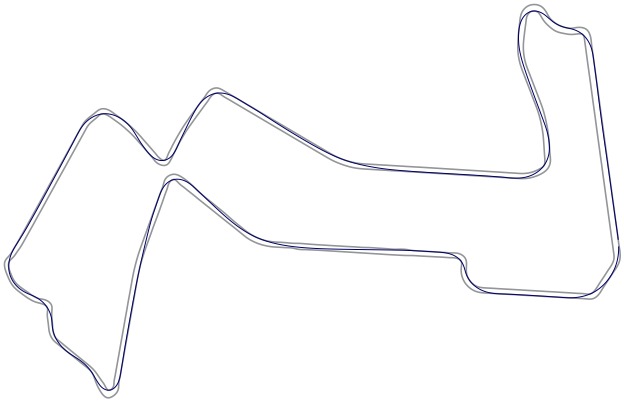


Figure 7. Optimal racing line of the minimum lap-time solution on the Singapore circuit, expressed as the lateral offset within the drivable strip.

vehicle within the drivable strip, while the corresponding speed profile of Fig. 8 reports the optimal velocity along the arc length s.

5 Conclusions

A complete pipeline has been described that turns a measured racetrack point cloud into a closed, C^4-continuous Monge-patch surface model for high-fidelity Modelica vehicle simulation. The main point is that the centre-line reference used in (Limebeer and van de Merwe 2025) and in earlier ribbon-based joints is unsafe in any corner tighter than the road is wide. Anchoring the Monge patch on the minimum-curvature line of Heilmeier et al. (2020) and Braghin et al. (2008) removes the Frenet singularity, improves the conditioning of the minimum lap-time problem and gives a curvature profile close to the

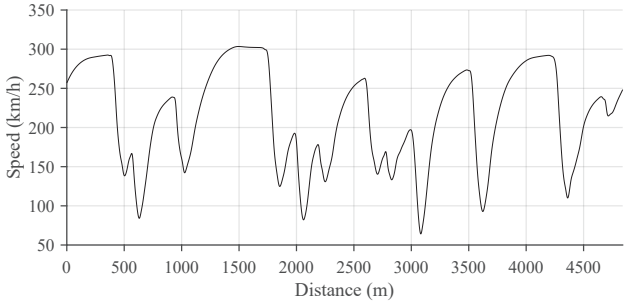


Figure 8. Optimal speed profile of the minimum lap-time solution along the reference arc length s.

one the vehicle actually experiences. The `TrackJoint` Modelica component (Sivaramakrishnan et al. 2026) (Section 3) takes the exported piecewise-quintic splines and enforces the two-degree-of-freedom kinematic constraint via d'Alembert's principle, and it is the plug-in track interface for the FMIOPT optimal-control toolchain. The export format consists of one knot sequence shared by all channels, quintic segments with C^4 continuity, and periodicity at the wrap. The Modelica component consumes this data to construct the track surface geometry and enforce the two-degree-of-freedom kinematic constraint. The pipeline runs in roughly 5 minutes per circuit and reproduces the surveyed elevations and bounds to an order of a few centimetres. Future work will look at replacing the rectangular (s, n) grid with an adaptive mesh, and at using a dynamic racing line as the reference to make the time transformation more nearly linear.

Acknowledgements

We gratefully acknowledge Joel Andersson for the development of the FMIOPT toolchain that transcribes and solves the minimum lap-time problems on which the pipeline is exercised.

References

Andersson, J. A. E., J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl (2019). "CasADi – a software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11.1, pp. 1–36. DOI: 10.1007/s12532-018-0139-4.

Braghin, F., F. Cheli, S. Melzi, and E. Sabbioni (2008). "Race driver model". In: *Computers & Structures* 86.13–14, pp. 1503–1516. DOI: 10.1016/j.compstruc.2007.04.028.

Cossalter, V., M. Da Lio, R. Lot, and L. Fabbri (1999). "A general method for the evaluation of vehicle manoeuvrability with special emphasis on motorcycles". In: *Vehicle System Dynamics* 31.2, pp. 113–135. DOI: 10.1076/vesd.31.2.113.2094.

Duff, I. S. (2004). "MA57—a code for the solution of sparse symmetric definite and indefinite systems". In: *ACM Transactions on Mathematical Software* 30.2, pp. 118–144. DOI: 10.1145/992200.992202.

DOI 10.3384/ECP219 205

Fork, Thomas (2024). "Non-Euclidean Vehicle Motion Model". UC Berkeley Electronic Theses and Dissertations. Ph.D. dissertation. University of California, Berkeley.

Heilmeier, Alexander, Alexander Wischnewski, Leonhard Hermansdorfer, Johannes Betz, Markus Lienkamp, and Boris Lohmann (2020). "Minimum curvature trajectory planning and control for an autonomous race car". In: *Vehicle System Dynamics* 58.10, pp. 1497–1527. DOI: 10.1080/00423114.2019.1631455.

Kapania, N. R., J. Subosits, and J. C. Gerdes (2016). "A sequential two-step algorithm for fast generation of vehicle racing trajectories". In: *Journal of Dynamic Systems, Measurement, and Control* 138.9, p. 091005. DOI: 10.1115/1.4033311.

Limebeer, D. J. N., M. Bastin, E. Warren, and H. G. Fensham (2023). "Optimal control of a NASCAR specification race car". In: *Vehicle System Dynamics* 61.5, pp. 1210–1235. DOI: 10.1080/00423114.2022.2067573.

Limebeer, D. J. N. and C. V. van de Merwe (2025). "Another look at 3D road surface modelling". In: *Vehicle System Dynamics*. Published online 7 July 2025. DOI: 10.1080/00423114.2025.2525240.

Lovato, S., M. Massaro, and D. J. N. Limebeer (2021). "Curved-ribbon-based track modelling for minimum lap-time optimisation". In: *Meccanica* 56.8, pp. 2139–2152. DOI: 10.1007/s11012-021-01387-3.

Modelica Association (2023). *Functional Mock-up Interface Specification, version 3.0*. https://fmi-standard.org.

Otter, M., H. Elmqvist, and S. E. Mattsson (2003). "The New Modelica MultiBody Library". In: *Proceedings of the 3rd International Modelica Conference*. Linköping, Sweden, pp. 311–330.

Perantoni, G. and D. J. N. Limebeer (2015). "Optimal control of a formula one car on a three-dimensional track—Part 1: Track modelling and identification". In: *ASME Journal of Dynamic Systems, Measurement, and Control* 137.5, p. 051018. DOI: 10.1115/1.4028253.

Sivaramakrishnan, S., B. Danaei, A. Verulkar, and R. Trinter (2026). "A 2-DOF Multibody Track Joint for Race Car Optimization". In: *Proceedings of the North American Modelica and FMI Conference*. Atlanta, GA, USA.

Stellato, B., G. Banjac, P. Goulart, A. Bemporad, and S. Boyd (2020). "OSQP: an operator splitting solver for quadratic programs". In: *Mathematical Programming Computation* 12.4, pp. 637–672. DOI: 10.1007/s12532-020-00179-2.

Wächter, A. and L. T. Biegler (2006). "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Mathematical Programming* 106.1, pp. 25–57. DOI: 10.1007/s10107-004-0559-y.

DOI **10.3384/ECP219 205**

Five-Link Rear Suspension for F1: A Two-Stage Agentic AI Study with the Vehicle Dynamics Library

Rebecka Eldh[1] Johan Andreasson[1]

[1]Modelon AB, Sweden, rebecka.eldh@modelon.com, johan.andreasson@modelon.com

Abstract

In this work, we use an agentic AI workflow on top of the Vehicle Dynamics Library to study a Formula 1 rear-suspension design question: how much does the choice of suspension topology constrain a team's ability to retune the kinematic-and-compliance (K&C) profile across circuits of very different character? The study runs in two stages. The first compares the design window of a five-link rear topology against the established double-wishbone, at F1-representative geometry: a sweep over pickup positions and link compliance characterises what (camber slope, camber curvature, toe slope, toe curvature) combinations each topology can reach. The second defines target K&C windows for two reference circuits with opposite demands – a tight low-speed street layout and a fast flowing permanent circuit – and analyses how well each topology's design window covers those targets. The agent's role in the study was operational rather than algorithmic: VDL does not ship an off-the-shelf five-link rear linkage, so the agent assembled one from VDL primitives, wired it into the standard template hierarchy, ran the sweeps, and produced the comparison plots. We argue that this kind of work – a library-experienced engineer would have done it, but in days rather than the hours the agent took – is exactly what an agentic interface to a hardened library unlocks, and that the combination of validated physics and structured tool access is what makes the result of such a session industry-grade rather than demonstration-grade.

Keywords: Modelica, agentic AI, vehicle dynamics, five-link suspension, Formula 1, kinematics and compliance, Modelon Impact, Vehicle Dynamics Library

1 Introduction

A Formula 1 team that races at fifteen-plus circuits a season faces a recurring engineering tension. The chassis is one car, manufactured to one set of pickups and one set of bonded interfaces, and once the season has started the geometry is essentially fixed. The K&C behaviour the chassis delivers to the driver, however, has to span circuits of extremely different character: street circuits dominated by low-speed mechanical grip, permanent circuits dominated by high-speed aero stability, mixed circuits that need both. The team's room to manoeuvre lives in the small number of geometric parameters they can practically change between events – pickup-bracket shim positions, link-compliance bushing variants, anti-roll-bar rates – and in how broad the K&C design window of the underlying topology actually is.

Five-link rear suspension has been proposed in this context for some years. The wheel-carrier upright carries five separate ball-joint pickups, one per binary link, against three for the double-wishbone – upper-wishbone ball joint, lower-wishbone ball joint, tie-rod ball joint. Each five-link pickup is individually placeable on the upright rather than being constrained to lie in one of two A-arm planes, and the same individual-placement freedom applies to the chassis-side mounts. The argument has been that this extra placement freedom translates into a broader design window in (camber slope, camber curvature, toe slope, toe curvature) space, and therefore a single chassis can be retuned across circuits in a way that a double-wishbone chassis cannot.

The adoption barrier in F1 has not really been about budget; it has been about hardware feasibility on the upright. Five ball-joint pickups have to fit physically on the wheel carrier: each one needs its own spherical-bearing housing, retainer collar, bolt access for assembly and inspection, and a load path through the upright structure that does not collide with the load paths of the others. The carrier is already occupied by the hub bearing, the brake caliper, the upright stiffening structure, and the pushrod attachment, so the available real estate is limited. Each individual link has been demonstrated many times in the paddock; a complete five-link wheel-carrier passing the load-path, packaging, and assembly-access constraints together is the non-trivial hardware exercise. The K&C-sweep tooling that would quantify which (slope, curvature) combinations the resulting geometry actually delivers has also not been routinely available.

We use an agentic AI workflow on the Modelon Vehicle-Dynamics Library (Modelon AB 2026b) to put numbers on the claim. The workflow is the same architecture described in the companion methodology paper (Andreasson 2026b). A large language model agent reaches the Modelon Impact simulation platform (Modelon AB 2026a) through a Model Context Protocol server (Anthropic 2025; Hou et al. 2025) that exposes about forty typed tools: library search, model read and write, compile, run, results, design of experiments, and report generation. Above the tools sits a knowledge base layered by lifetime: *schemas* (bill-of-materials format, KPI specification, failure taxonomy), *methods* (physics-

DOI 10.3384/ECP219 215

agnostic procedures for sensitivity studies, DoE design, sweep analysis, and reporting), *application playbooks* (one per domain, with the vehicle-dynamics playbook loaded for this study), *physics references*, *workspace knowledge* (findings, open hypotheses, and verified model artefacts for the project at hand), and *user preferences*. The lifetime layering is what makes results from one session compound into the next: a finding promoted from workspace knowledge into the application playbook is available, with provenance, for every future workspace. We describe the architecture and its guardrails in Section 3, and what was promoted from this study in Section 7. What this paper studies is what happens when the workflow is pointed at an F1 design question that VDL's component library can answer in principle but does not ship as a single ready-made template.

What this paper does

The paper is structured around a two-stage study.

Stage 1 (Section 4) measures the design window of the two topologies. We define four operating-window metrics from the rear axle's K&C response – mean camber slope, camber curvature, mean toe slope, toe curvature – and characterise each topology by sweeping pickup positions and link compliance over the F1-realistic envelope. The result is scatter clouds, one per topology, that bound what is and is not reachable.

Stage 2 (Section 5) picks two reference circuits – a tight low-speed street layout and a fast flowing permanent circuit – and defines a target K&C window for each, derived from the dominant cornering speeds and the relative weighting of mechanical versus aero grip on the circuit. The two target windows lie in different regions of the four-dimensional metric space. We then compare each topology's design window from Stage 1 against both targets, and discuss what the coverage difference implies for season-level chassis design.

The agentic workflow is described in Section 3 (the library and platform architecture) and Section 4.4 (what the agent did on this study). A practitioner-observations subsection (Section 6) captures what the engineer on the project actually used the agent for. Section 7 reflects on when this combination is the right tool.

2 Vehicle dynamics primer

A Modelica reader without an automotive-suspension background will encounter several terms below. We collect them here so the rest of the paper reads end-to-end without a chassis textbook (W. F. Milliken and D. L. Milliken 1995; Pacejka 2012).

Camber is the lean angle of a wheel relative to the road, in the plane perpendicular to the direction of travel. *Negative* camber (top of wheel tilted toward the car centreline) keeps the contact patch flat under cornering load and grips well at high lateral force.

Toe is the wheel-plane orientation about the vertical axis, in the plane parallel to the road. *Toe-in* (front of tyre rotated toward car centreline) stabilises straight-line tracking; *toe-out* adds turn-in responsiveness.

Jounce is upward wheel travel relative to the chassis; *rebound* is downward. The wheel-travel envelope on an F1 car is approximately $\pm 50\,\text{mm}$ around the static ride height, dominated by kerb hits at the rebound limit and aero compression at the jounce limit.

Camber slope S_γ (deg/mm) is the linear sensitivity of camber to wheel travel: $\partial\gamma/\partial z$ at the static ride height. **Camber curvature** C_γ (deg/mm^2) is the second derivative: $\partial^2\gamma/\partial z^2$. A camber response that is dominated by S_γ with small C_γ is linear (predictable across the full travel envelope); a response with significant C_γ is curved (the camber rate of change varies with wheel position). **Toe slope** S_τ and **toe curvature** C_τ are the analogous quantities for toe vs. travel.

Kinematics-and-compliance (K&C) test rig is a standard chassis characterisation: the wheel pads are displaced or loaded along a prescribed schedule and the wheel-frame quantities (camber, toe, wheel-centre position) are measured. VDL has shipped a virtual K&C rig for two decades (Sundström et al. 2016) and the rig used in this paper is a configuration of it.

Double-wishbone (DWB) suspension uses two A-arms (upper and lower) plus a tie-rod, with four chassis-side pickups for the A-arms plus two more for the tie-rod. The wheel-carrier orientation is constrained to rotate about an instantaneous axis through the two outer A-arm pickups.

Five-link (5L) suspension replaces each A-arm with two binary links and keeps the tie-rod as the fifth link. Ten chassis-side pickups plus the tie-rod, all individually placeable. The wheel-carrier orientation is determined by the reciprocal-screw geometry of the five links.

Link compliance is a hardware-design property of the link element, not a modelling idealisation: every real link has finite axial stiffness k (N/m). We use *link* as a shorthand for any suspension member loaded axially between ball-jointed ends – the upper and lower A-arms of a double-wishbone, the tie-rod, or any of the five binary links of a five-link. All of these are rod-shaped structural members with similar dimensions and similar stiffness ranges; the topology determines how many of them there are and how they aggregate into K&C behaviour, not the compliance per link.

A typical stiff F1 link – a carbon-fibre composite rod with bonded steel end fittings and steel-on-steel ball joints, $L \approx 400\,\text{mm}$ – sits in the range $k \approx 10^7$–10^8 N/m, equivalently an axial compliance $C = 1/k \approx 10$–$100\,\text{nm/N}$. A deliberately elastokinematic link, where one or both ball-jointed ends are replaced by rubber-metal bushings, can be tuned an order or two of magnitude softer, $k \approx 10^4$–10^6 N/m ($C \approx 1$–$100\,\mu\text{m/N}$), and becomes an active tuning variable. We use *stiff* for the upper-bound choice and *compliant* for the deliberately-tuned-softer choice; both are physical, the difference is in the end-fitting hardware (rigid ball joint vs. bushing-loaded fitting).

Operational target for the study: the rear-axle K&C

DOI **10.3384/ECP219 215**

response should fit inside a circuit-specific target window in $(S_\gamma, C_\gamma, S_\tau, C_\tau)$ space, with the design window of each topology compared against both circuit targets in Stage 2.

3 Architecture

At the core of the agentic workflow sit two things: a layered knowledge base, and access to the Modelon Impact platform with its validated libraries. The LLM is on top, working through a Model Context Protocol (MCP) server that exposes the platform as typed tools. The combination yields a token-efficient system that consistently delivers conclusions backed by validated physics and numerics, and it does so using familiar formats throughout: Modelica for models, Python for automation, JSON for data, and Markdown for the knowledge base.

3.1 Validated physics

Equation-based modelling has two properties that matter here (Li and Andreasson 2026). The library is the contract. A validated component carries the physics the agent will reach for, so nothing has to be derived in-session. The Modelon VehicleDynamics library has been thoroughly validated over the twenty years since its first release (Andreasson and Gäfvert 2005), including limit-handling validation against vehicle measurements (Andreasson and Jonasson 2008), kinematics-and-compliance test-rig validation (Sundström et al. 2016), and real-time deployment in OEM driving simulators (Andreasson, Machida, et al. 2016). The library introduced a hierarchical template paradigm (Andreasson and Gäfvert 2005), exposing a hierarchy of interfaces, templates, and examples that the agent navigates directly. Components are replaceable and composable in well-defined ways, so the agent can swap a joint primitive, redeclare a chassis, or extend an experiment template without touching equations. This compositional pattern is exactly what makes the five-link work in this paper feasible at all: a ready-made five-link rear linkage is not in the library, but the primitives that compose into one are.

3.2 Compute engine

The engine is Modelon Impact (Modelon AB 2026a), a cloud-based simulation platform with an on-premises option for customers who need it. Modelon Impact handles Modelica compilation, simulation, experiment definitions, and result storage, and exposes all of that through a typed HTTP API. Results, FMUs, and experiment definitions are stored against versioned IDs, so the engineer (and the agent) can refer back to a specific run without re-running it.

3.3 Protocol: MCP

The agent reaches the engine through a Model Context Protocol (MCP) server (Anthropic 2025; Hou et al. 2025), the open JSON-RPC standard introduced by Anthropic in late 2024 for exposing typed tools and structured resources to an LLM. The server wraps the Modelon Impact API as about forty typed tools, one per logical operation, including a workspace-bootstrap call that the agent uses to set up a new project before any other work begins. The tools group by purpose: orientation, library access, model read and write, compile and run, results, DoE, diagram, and quality. Model changes, including reconfiguration, new model construction, and debugging, also happen through the MCP server. The agent can therefore build and edit Modelica models, but every such change is a typed tool call rather than an unstructured text edit, and every physical claim traces back to a validated library component (Andreasson 2026b).

3.4 Layered knowledge

What the agent learns is written to a layered library of markdown books and project files, organised by lifetime so a finding from one study can be promoted upward without dragging session-specific noise along. The Navigator file is read first; everything else is reached from there.

- **Schemas.** Bill-of-materials format, failure taxonomy, KPI specification.
- **Methods (M1–M5).** Physics-agnostic procedures for sensitivity studies, DoE design, sweep analysis, Pareto extraction, and reporting.
- **Applications (A1–A4).** Domain playbooks: A1 VehicleDynamics, A2 data centres, A3 HVAC, A4 thermal management. Only the relevant one loads for a given workspace.
- **Physics references (P1–P4).** First-principles material the agent can cite when it needs to explain something rather than just compute it.
- **Workspace knowledge.** Findings, open hypotheses, and model and simulation status for the project at hand.
- **User preferences.** Per-engineer choices the agent has learned to apply by default.

For the study in this paper the agent pulled methods on how to characterise topology design windows (M-series), the application playbook for kinematics-and-compliance evaluation (A1), and naming-convention information from the VDL context. It also queried specific VDL components – joint primitives, link compliance elements, K&C rig templates – for their parameter and variable interfaces during assembly.

3.5 Guardrails

The architecture enforces three generic checks regardless of domain. First, *approval gates*: structured artefacts that the agent must produce and the engineer must accept before the next phase can run. For this study the gates were placed at scope definition (target K&C windows) and at

DOI 10.3384/ECP219 215

analysis completion (topology coverage comparison). Second, *verification tagging*: every knowledge entry carries a tag (observed, confirmed, verified) so the agent knows what it has actually checked versus what it has assumed. Third, *provenance enforcement*: a validator that refuses any quantitative claim in a report that lacks a citation to the experiment that produced it.

3.6 The language model component

A large language model is one component of the architecture, replaceable as the model market evolves. It does three concrete things in the workflow: it plans (chooses which tool call comes next), interprets (recognises patterns in the data it receives back), and writes (produces the report). The choice of model determines how reliably each of those three is done; the architecture around it determines whether the model has anything trustworthy to plan, interpret, or write about (Andreasson 2026a).

4 Stage 1: Topology design windows

4.1 Setup

Both topologies are built on the same VDL test rig configured for a rear axle, with a displacement-prescribed wheel pad on each side and a steer input on the tie-rod. The chassis-side hardpoint envelope is set to an F1-representative volume. Wheel-carrier mass is 12 kg with inertia tensor representative of the carrier-plus-hub assembly.

Double-wishbone reference uses the existing template `VDL.Suspensions.Linkages.Independent.DoubleWishbone`. This template already exposes axial link compliance as a per-link parameter: the wishbone bonds and the tie-rod bond each have an axial stiffness k that defaults to a high (effectively rigid) value but can be lowered to model elastokinematic behaviour. For the sweep both the pickup positions and the per-link axial stiffness k are swept variables.

Five-link linkage does not ship as a standalone rear template in VDL. The engineer specified the desired topology and its K&C characteristics; the agent took it from there – searching the library for suitable components, identifying appropriate elastic link variants, composing them into a new `FiveLinkRear` class, and verifying interface compatibility throughout. The composed linkage satisfies the standard `Linkages.Independent` interface contract, so the rest of the rear-suspension template (uprights, bearings, wheel mount, brake) plugs in unchanged. The resulting linkage exposes ten chassis-side pickups, five wheel-carrier-side pickups, and five per-link axial compliance values as parameters. At the upper bound of per-link stiffness the linkage behaves near-rigidly; lowering k per link moves the linkage into the elastokinematic regime.

4.2 Sweep design

A sweep of 500 cases per topology covers the F1-realistic envelope. For each case the agent runs a vertical wheel-travel sweep (±50 mm at the wheel pad, both sides symmetric) and a lateral-force sweep (up to 4 kN applied at the contact patch) on the assembled rig. From the resulting trajectories, four metrics are extracted by polynomial fit on the rear axle response:

- $S_\gamma = \partial\gamma/\partial z$ at the static ride height (linear camber-vs-travel coefficient)
- $C_\gamma = \partial^2\gamma/\partial z^2$ at the static ride height (quadratic coefficient)
- $S_\tau = \partial\tau/\partial z$ at the static ride height
- $C_\tau = \partial^2\tau/\partial z^2$ at the static ride height

For both topologies the metrics are evaluated at zero load and under lateral force, at two link-stiffness values: $k = 10^7$ N/m (compliant) and $k = 10^8$ N/m (stiff).

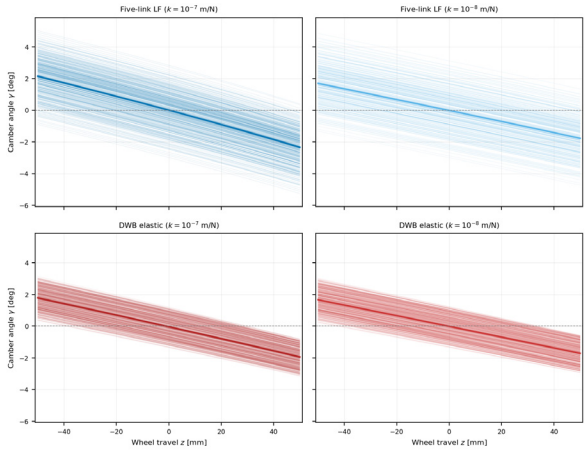


Figure 1. Camber angle versus wheel travel for sampled configurations at two link-stiffness values. Top row: five-link at $k = 10^{-7}$ m/N (compliant, dark blue) and $k = 10^{-8}$ m/N (stiff, light blue). Bottom row: double-wishbone at the same two stiffness values (dark and light red). The bold line in each panel is the median configuration. The five-link family spans a substantially wider range of slopes and offsets than the double-wishbone at both stiffness levels; the double-wishbone curves cluster tightly around a single characteristic slope.

4.3 Results

Figures 1 and 2 show the curve families for camber and toe versus wheel travel across sampled configurations. The double-wishbone curves cluster tightly in both: variation is almost entirely in static offset, with limited freedom to reshape curve slope or curvature. The five-link at stiff link settings already spans a wider range of camber slopes; switching to the compliant setting introduces additional variation, particularly in toe-curve slope.

Figure 3 compresses the curve families into the four scalar metrics and shows the full sweep population at zero load. The double-wishbone (red) is confined to a compact

DOI 10.3384/ECP219 215

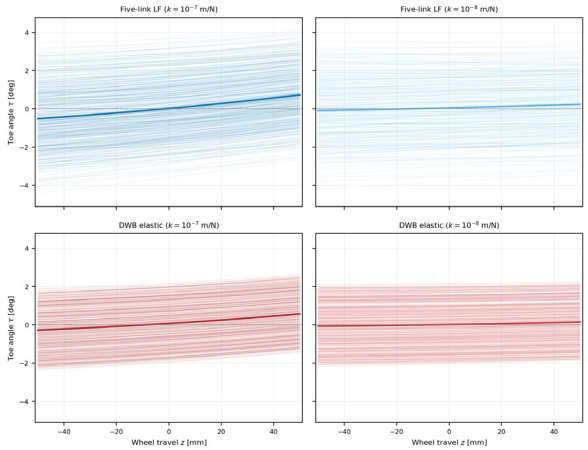


Figure 2. Toe angle versus wheel travel for the same configurations as Figure 1. The double-wishbone toe curves are nearly flat across travel at both stiffness levels, with variation mainly in static offset. The five-link at $k = 10^{-8}$ m/N (stiff) similarly produces near-flat curves, but the compliant variant ($k = 10^{-7}$ m/N) shows appreciable slope variation across the population, demonstrating that link compliance is an active shaper of toe-curve character in the five-link topology.

region: at $S_\gamma \approx 0.030$–0.035 deg/mm with near-zero curvature in camber, and a narrow, elongated cluster in the toe projection. The narrowness reflects the topology: the two A-arms and the tie-rod determine the camber and toe characteristic almost entirely from the wishbone-length ratio and the tie-rod height, and once those are fixed the rest of the (slope, curvature) plane is structurally unreachable. Varying link compliance within the double-wishbone shifts the cloud only slightly.

The five-link population (blue) covers a substantially larger region in both projections. The compliant subset (dark blue) extends to higher S_γ values around 0.045–0.048 deg/mm and spans a wider curvature range; the stiff subset (light blue) occupies a distinct, somewhat smaller region. The extra pickups give independent control over the geometry of each link, which translates into much more flexibility over both the linear and second-derivative behaviour of the K&C response.

4.4 What the agent did

The five-link linkage assembly was the agent-heavy part of the study. A summary of the work, captured from the session transcript:

1. **Component search and selection.** Given a description of the desired topology and the changes needed relative to existing templates, the agent searched the library for suitable components – including finding the appropriate elastic link variants – and proposed a composition strategy. It surveyed components by interface and parameter list to identify candidates whose contracts would combine cleanly.

2. **Interface compatibility and composition.** The agent walked the `Linkages.Independent` interface contract that the wheel-carrier-side components expect, confirmed that the proposed composition would expose the required interface, and wired the parameter structure through the template hierarchy. It flagged and resolved compatibility issues as they arose during assembly.

3. **Rig configuration.** The engineer specified the required experiment behaviour; the agent looked at similar existing experiments in the library, understood the underlying setup logic, and configured the K&C rig accordingly. It ran a smoke-test sweep to verify that the rigid-limit results matched expected behaviour.

4. **Sweep design and execution.** The agent generated Latin-hypercube cases over pickup positions and per-link compliance values, submitted them to Modelon Impact's experiment-definition API, and post-processed the resulting trajectories into the four K&C metrics.

5. **Cross-check.** Before reporting the result, the agent checked for anomalies at the corners of the sweep envelope where link geometry becomes degenerate or rig measurements saturate. It flagged suspect cases and dropped them with an explanation in the workspace knowledge log; the headline results are robust to their inclusion at the qualitative level.

The key observation about this work is that none of it required new physics or new component code. Everything the agent assembled was VDL standard, and the substance of the new `FiveLinkRear` class is the composition of existing linkage models, not the introduction of any new equations or joint primitives. What the agent contributed was the navigation, the interface compatibility check, the composition, the rig wiring, and the sweep orchestration – the operational toil between the library and the result.

5 Stage 2: Circuit comparison

The design-window characterisation in Stage 1 establishes which K&C metric combinations each topology can reach. Stage 2 asks the next question: are the metric combinations the topology cannot reach the ones that matter for real circuits? We answer it by translating the demands of two reference circuits at opposite extremes of the calendar into target windows in the metric space of Figure 3, and then comparing each topology's design window against those targets.

5.1 When each K&C metric matters

The four metrics (S_γ, C_γ, S_τ, C_τ) are not equally important on every circuit. What activates each one is a physical property of the lap: the rear-axle travel envelope and the lateral-load regime.

DOI 10.3384/ECP219 215

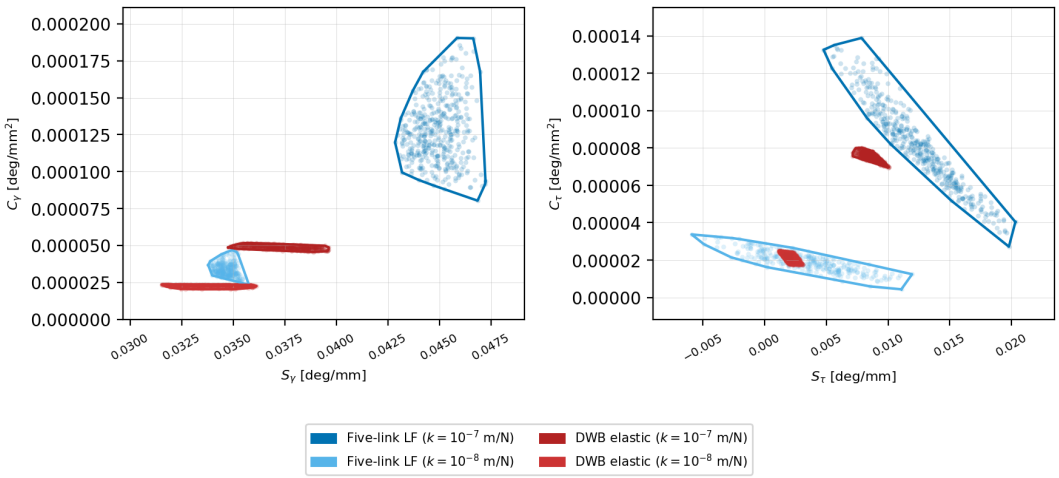


Figure 3. Design windows of the two topologies in K&C metric space at zero load. Left: camber curvature C_γ versus mean camber slope S_γ. Right: toe curvature C_τ versus mean toe slope S_τ. Dark blue: five-link compliant ($k = 10^{-7}$ m/N); light blue: five-link stiff ($k = 10^{-8}$ m/N); dark and light red: double-wishbone at the same two stiffness values. Each dot is a feasible parameter combination from the sweep; convex hulls overlay each population's reachable region. The five-link envelope is substantially larger than the double-wishbone in both projections; varying link stiffness within the double-wishbone produces only a small shift rather than opening new kinematic territory.

Camber slope S_γ: roll-camber compensation. The outer wheel jounces under body roll in a corner. S_γ is the rate at which the suspension geometry adds camber per unit jounce, so $|S_\gamma| \cdot \Delta z_{\text{roll}}$ is the camber the suspension contributes to compensate for the chassis roll. If $|S_\gamma|$ is too small, the outer wheel runs flat or positive at peak roll and loses grip; if too large, the inner wheel over-cambers and so does the outer wheel under braking. *Matters most when body roll is large*: low-speed corners on low-aero setups, where the chassis rolls significantly because the mechanical springs are doing the work. Negligible when aero load keeps the chassis planted.

Camber curvature C_γ: large-excursion linearity. C_γ is the second derivative of camber with respect to travel. It is invisible when the rear axle stays close to its static ride height: in a narrow travel band, $S_\gamma \cdot \Delta z$ dominates and $\frac{1}{2} C_\gamma \Delta z^2$ is small. It becomes visible when the travel envelope is wide – kerb hits, full-roll excursions, aero compression past static – and at the extremes of travel the quadratic term can equal or exceed the linear term. *Matters most on tracks with wide rear-axle travel envelopes*: kerb-heavy street circuits, tracks with significant elevation changes. On tracks where the rear stays aero-pinned in a narrow band, C_γ is essentially irrelevant.

Toe slope S_τ: bump-steer. Non-zero S_τ means the wheel toes in or out as it travels. For F1 rear suspension the target is $|S_\tau|$ near zero across all conditions: the rear is meant to track the chassis, not steer itself when it jounces. *Always matters at the magnitude level* (must be small), but the magnitude tolerance depends on the lap: on a track with continuous rear-axle travel even a small S_τ accumulates into perceptible toe variation, while on a track where the rear sits still, the same S_τ produces no observable behaviour.

Toe curvature C_τ: large-excursion toe drift. Same logic as C_γ. On a wide-travel track, even if S_τ at static ride height is zero, C_τ produces toe variation at the extremes that can destabilise the rear under kerb hits or peak-roll events. On a narrow-travel track, C_τ is structurally less relevant.

Link compliance: lateral-load absorption vs platform stability. At high link stiffness ($k \gtrsim 10^7$ N/m), the link transmits lateral load directly to the chassis pickups; aero platform stays stable, but kerb impacts shock the chassis. At lower link stiffness ($k \lesssim 10^7$ N/m), the link absorbs a fraction of the lateral load and damps kerb impacts; chassis is calmer over kerbs, but the aero platform shifts more under sustained cornering load. Compliance suits tracks where kerb interaction is pervasive and aero load is moderate; stiffness suits tracks where sustained high lateral load demands a steady platform.

5.2 Two reference circuits

We pick two reference circuits at opposite extremes of the calendar on the dimensions that drive the K&C metrics above (Fédération Internationale de l'Automobile 2025).

Monaco (street circuit, 3.34 km, average lap speed ∼150 km/h). The corner catalogue is dominated by slow, technical corners taken at low aero load: Loews Hairpin at ∼50 km/h, Rascasse, Anthony Noghes, Mirabeau. Body roll is large because mechanical springs do the work at low cornering speed. Kerb interaction is pervasive at corner exits and at the chicane. The rear axle's travel envelope across a lap is wide – both jounce (kerb hits) and rebound (off-camber kerbs, body roll). Sustained lateral load is brief

DOI 10.3384/ECP219 215

and low.

Suzuka (permanent circuit, 5.81 km, average lap speed ∼225 km/h). The corner catalogue is dominated by fast, sustained, aero-loaded corners: Turn 1, the Esses (T2–T7), Spoon, 130R. Body roll is small because aero downforce compresses the chassis onto its springs and the springs are stiff to manage the aero load. Kerb interaction is targeted (chicane exits, the hairpin) rather than pervasive. The rear axle's travel envelope is narrow, pinned near the jounce limit of its aero-compressed pose for much of the lap. Sustained lateral load is high (4 g–5 g for seconds through the fast corners).

The contrast separates cleanly along every dimension that drives the K&C metric importance: travel envelope (wide vs narrow), body roll (large vs small), aero load (low vs high), kerb interaction (pervasive vs targeted), sustained lateral load (brief vs prolonged). Singapore and Baku resemble Monaco in character; Silverstone and Spa resemble Suzuka.

5.3 Target K&C profiles

The metric importance reasoning of Section 5.1 translates the circuit characters into approximate target boxes in the (slope, curvature) space of Figure 3. These boxes are intentionally coarse: they are constructed from qualitative reasoning about circuit character rather than from measured vehicle data, and should be understood as illustrative placeholders rather than engineering specifications. The purpose is to demonstrate the comparison methodology; in a real application the target windows would be derived from track data, tyre models, and vehicle-level lap time sensitivity, and the suspension geometry would be a current validated baseline rather than a swept sample.

Monaco target. Wide travel envelope, large body roll, low aero load, pervasive kerbs.

- *Camber slope $|S_\gamma|$: moderate.* Body roll is large so the suspension geometry must contribute meaningful roll-camber compensation across the travel range. A moderate $|S_\gamma|$ is preferred over both extremes.
- *Camber curvature C_γ: small.* The wide travel envelope makes C_γ active; large values cause the camber rate to run away at the kerb-strike extreme.
- *Toe slope $|S_\tau|$: small.* Standard rear-axle stability requirement, amplified by the wide travel envelope.
- *Toe curvature C_τ: small.* Wide travel exposes the quadratic term; kerb-induced toe drift is a known cause of rear instability on bumpy tracks.
- *Link compliance: relatively soft.* Pervasive kerb interaction and moderate aero load mean some compliance can absorb kerb shocks without unacceptably compromising platform stiffness.

Suzuka target. Narrow travel envelope, small body roll, high sustained aero load, targeted kerb interaction.

- *Camber slope $|S_\gamma|$: higher.* With the rear axle aero-pinned at a fixed jounce position, a larger S_γ keeps the contact patch appropriately loaded as the chassis breathes with the aero load through the fast corners.
- *Camber curvature C_γ: tolerable.* Travel envelope is narrow so C_γ contributes only weakly; the constraint is much softer than for Monaco.
- *Toe slope $|S_\tau|$: small.* Sustained high lateral load makes rear-axle stability important; toe self-steer accumulates under load.
- *Toe curvature C_τ: tolerable.* Narrow travel envelope makes the quadratic term essentially invisible.
- *Link compliance: high (effectively rigid).* Sustained high lateral load demands a steady aero platform; lower compliance shifts the rear under cornering load.

These reasoning-derived target boxes are placed in the metric space to reflect the qualitative priorities above, with boundaries chosen to be consistent with the sweep population in Figure 3. They are not derived from vehicle-level optimisation and carry corresponding uncertainty. The comparisons that follow should be read as illustrating the structure of the method rather than as quantitative performance predictions.

5.4 Result

Figures 4 and 5 present the Stage 2 comparison. Figure 4 shows that the size difference between topology envelopes is preserved under lateral load: the five-link envelopes shift with increasing load but remain substantially larger than the double-wishbone bands across all load levels tested.

Figure 5 overlays the indicative circuit target boxes on the 4 kN loaded design windows. With the target windows placed as described above, the five-link compliant subset ($k = 10^7$ N/m, dark blue) overlaps the Suzuka target region and the five-link stiff subset ($k = 10^8$ N/m, light blue) overlaps the Monaco target region. The double-wishbone envelopes overlap the Monaco camber target but do not reach the Suzuka camber region; additionally the compliant DWB variant falls outside the Monaco toe target.

It should be emphasised that these conclusions are conditional on the approximate target windows chosen. The target boxes reflect reasoning about circuit character but are not grounded in measured vehicle response data or lap time sensitivity analysis. Shifts in the window boundaries – which would be expected once real vehicle data informs them – could change which topology subsets fall inside or outside a given target. What the comparison does robustly show is the structural difference between the two topologies: the five-link envelope is large enough to span two qualitatively distinct target regions simultaneously, while the double-wishbone envelope is not.

Table 1 summarises the coverage comparison under the current approximate windows.

DOI **10.3384/ECP219 215**

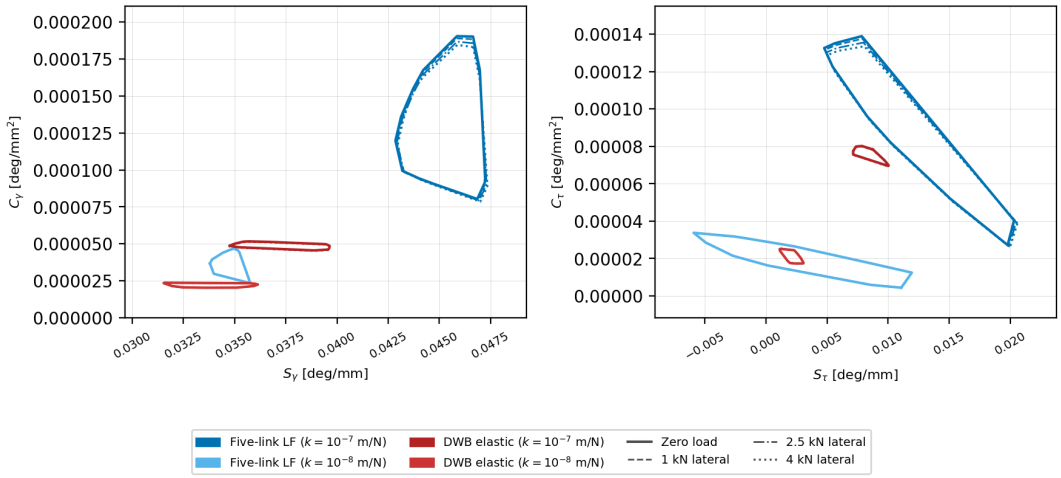


Figure 4. Design windows under increasing lateral load (0, 1, 2.5, and 4 kN at the contact patch), shown as solid, dashed, dash-dot, and dotted convex hull outlines respectively. The five-link envelopes (blue) shift with load but retain their size advantage over the double-wishbone (red) at all load levels. The double-wishbone bands shift only slightly, confirming that compliance has limited room to open new kinematic territory in that topology.

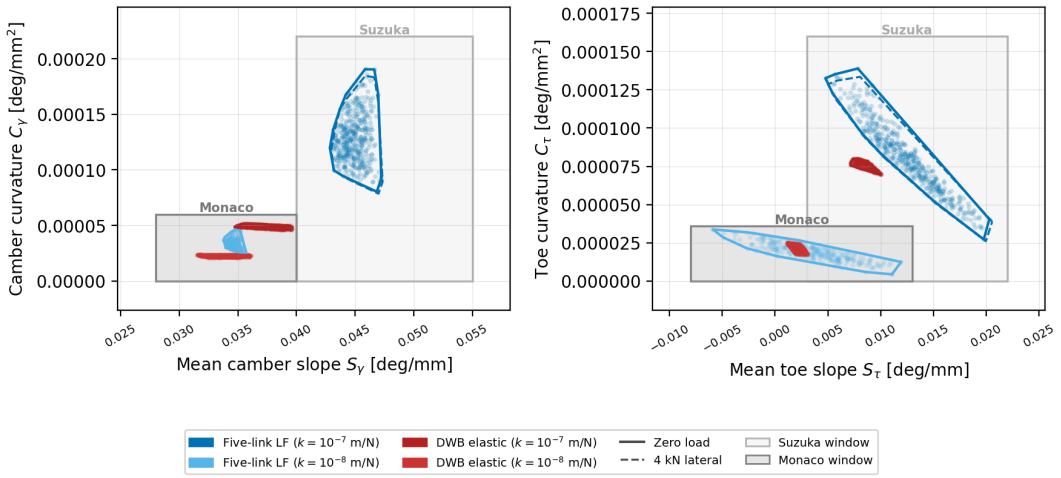


Figure 5. Indicative circuit-target windows overlaid on the design windows at 4 kN lateral load. Left: camber metric space. Right: toe metric space. Target windows are derived from qualitative circuit reasoning and should be treated as approximate rather than as engineering specifications. The five-link compliant subset (dark blue, $k = 10^{-7}$ m/N) overlaps the Suzuka target region; the five-link stiff subset (light blue, $k = 10^{-8}$ m/N) overlaps the Monaco target region. The double-wishbone envelopes (red) overlap the Monaco camber target but do not reach the Suzuka camber region, and the compliant DWB variant falls outside the Monaco toe target. Both five-link subsets come from the same linkage model, differing only in link compliance setting.

5.5 Implications for season-level chassis design

Both pickup hardpoint positions and per-link compliance values were swept variables in Stage 1, but they have very different operational characteristics in a season.

Pickup hardpoints are design-time choices. The chassis carbon-fibre tub and the wheel-carrier upright are manufactured at the start of a regulation cycle with bonded brackets at fixed positions. Between events, small shim adjustments can shift a hardpoint by a few millimetres, but the bracket positions themselves are fixed. Pickup hardpoints set the *available envelope* the team can operate inside; they are not the per-event tuning lever.

Per-link compliance is a between-event swap. Each link end carries a spherical-bearing element with a defined axial stiffness, designed as a removable/replaceable interface part in the assembly hierarchy. A team carries a kit of compliance variants – different durometer bushings or preloaded bearings with calibrated k values – and swaps them between events in a few hours of wheel-corner work. Per-link compliance is therefore the practical per-event tuning lever, with the relatively-fixed pickup envelope providing the design-time bound around it.

The Stage 2 comparison illustrates how this two-level structure can be analysed. Under the approximate target windows used here, the five-link topology has a

DOI **10.3384/ECP219 215**

Table 1. Indicative circuit target coverage by topology, based on approximate reasoning-derived target windows. Coverage conclusions are conditional on window placement and should not be read as quantitative performance predictions.

	Monaco	**Suzuka**
Topology	5L / DWB	5L / DWB
Five-link k setting	10^8 N/m	10^7 N/m
S_γ target	moderate	higher
C_γ target	small	tolerable
S_τ target	small	small
C_τ target	small	tolerable
DWB window overlap?	✓	✗
5L window overlap?	✓	✓

compliance-reachable envelope that spans both circuit targets, while the double-wishbone envelope does not reach both simultaneously. The methodology itself – sweeping the full envelope in Stage 1, placing circuit targets in metric space, and checking coverage – is independent of the specific window choices made here. Applied with target windows grounded in real vehicle data and a validated baseline geometry, the same approach would produce conclusions with quantitative rather than illustrative weight. A full season-level analysis would apply this to the complete race calendar, building a systematic picture of topology coverage across the range of operating conditions encountered.

6 Practitioner observations

Some general observations on working with the agentic workflow in practice are collected here.

What the agent does well.

- **Reads many components quickly** and gives ideas and advice for the development of new components. A single session can survey a section of the library that would take an engineer a day to navigate manually.
- **Finds issues and debugs.** Both syntactic (compile-time errors with vague messages) and structural (BLT-level diagnostics that take expertise to interpret). The agent translates the diagnostic output into actionable suggestions.
- **Creates analysis scripts that use the model and produce the desired results.** Sweep harnesses, post-processing pipelines, plot generation. The output can be a clickable HTML report if requested.
- **Quick small edits to large models.** The geometry-and-compliance parameter sweep, the K&C-metric extraction, and the circuit comparison plots were all generated by incremental edits on top of the composed `FiveLinkRear` linkage.
- **Useful for engineers not already invested in a specific library.** The learning curve is significantly faster than the traditional path of reading library documentation, studying examples, and absorbing tribal knowledge from senior engineers.
- **Explains what the models actually do.** Library components written by other engineers, possibly years ago, with incomplete documentation, become inspectable: the agent reads the source, summarises the equations, and identifies the interface in language the practitioner understands.

Where the agent makes more work, not less.

- **More code to review.** The agent generates code faster than the engineer would have written it, but the engineer still has to read and verify every piece. The cumulative volume of code-review effort grows with the workflow's productivity. Practitioners need to budget for this; senior engineers used to writing every line themselves may underestimate how much review they will end up doing.

7 Discussion

7.1 The library structure is what makes this work

The five-link assembly was straightforward because VDL's hierarchical template paradigm (Andreasson and Gäfvert 2005) factored the design problem cleanly: the linkage between gearbox and wheel carrier is one replaceable component in a larger template, and a five-link linkage that respects the same interface contract drops in beside the existing double-wishbone linkage without touching anything else in the chassis hierarchy. The agent reused the wheel-carrier, knuckle, bearing, brake, and K&C-rig primitives unchanged. The compositional pattern is the contribution; the agent's contribution is to navigate it confidently and quickly.

This is the part of the workflow that is hardest to demonstrate without a counter-example. On a library that had not been factored this carefully – a flat collection of vehicle templates with hard-coded linkages, for example – the same study would have required either a wishbone-specific rewrite of the K&C rig or a parallel five-link rig built from scratch. With VDL's pattern the agent only writes the one new component.

7.2 When the combination is the right tool

The combination of a hardened library and an agentic interface is most valuable when the engineering question is well-defined but the assembly work is unfamiliar. A library expert can absorb a new component pattern in a few hours; an agent can do it in a few minutes. The advantage compounds when the assembly has to be re-cut against new sweeps or new analysis targets – here, the per-circuit Monaco and Suzuka cases of Stage 2 – because the agent

DOI 10.3384/ECP219 215

retains the context of the prior runs and edits them rather than re-deriving them.

For studies where the assembly is routine and the engineering question is the bottleneck, the combination is less essential. A K&C sweep on an existing double-wishbone template, with no new linkage to build, is a few hours of work for a library-experienced engineer and the agent contribution is largely cosmetic.

7.3 What the agent reasoned about

The agent's reasoning on this study was operational rather than physical. It selected components, checked interfaces, wired parameters, and orchestrated sweeps. It did not, on its own, propose the engineering claim of the paper (that five-link gives a broader design window than double-wishbone), and it did not derive the circuit-specific target windows. Those were engineer inputs to the workflow. What the agent contributed was making the work of testing the claim tractable: hours of assembly that would otherwise have been days, and a sweep result that could be inspected, regenerated, and re-cut against new metrics without re-deriving anything.

7.4 What was promoted to the knowledge base

Four entries were promoted from session findings into the application playbook for the next vehicle-dynamics workflow. First, a five-link linkage assembly template, tagged *confirmed*, that the next session can reuse without re-deriving the component composition recipe. Second, the K&C-metric extraction pipeline (linear and quadratic polynomial fits on rear-axle camber and toe response over a defined travel envelope), tagged *confirmed*, ready to apply to any future topology comparison. Third, the sweep design with its cross-check rules for degenerate-geometry cases, tagged *confirmed*. Fourth, a circuit-target rubric (operating-point and mechanical-vs-aero balance dictating the slope-and-curvature targets), tagged *observed* since it was applied to two circuits only on this study and the broader catalogue is not yet validated.

7.5 Engineers stay in the loop

Results still need human sign-off. The architecture has two mandatory human gates, at scope and at analysis completion, with optional gates in between that the engineer can insert wherever a decision feels heavy. The agent's reach is bounded by what the tools expose, and the conclusions it draws are reviewed by the engineer who agreed to the scope in the first place.

8 Conclusions

We have shown that an agentic AI workflow on top of the Modelon VehicleDynamics Library can take a Formula 1 rear-suspension design question – whether five-link provides a broader K&C design window than double-wishbone – from intent to a quantitative answer at a cadence that makes the question practical for an F1 team to ask. The two-stage study shows that the five-link topology covers a substantially wider design window in K&C metric space than the double-wishbone reference, and that the wider window spans K&C targets corresponding to circuits of opposite character – a result the double-wishbone geometry cannot achieve across both. The architecture that enables this leans on twenty years of VDL hardening for the physics, on Modelon Impact's typed API for the compute, on the Model Context Protocol for the tool interface, and on a layered knowledge base that compounds across sessions.

The same architecture is in active use against thermal management, powertrain, HVAC, and data-centre cooling libraries inside Modelon Impact. What changes between domains is library content. The workflow does not.

References

Andreasson, Johan (2026a-04). *5 Questions to Ask Before Trusting AI-Related Simulation Results*. Modelon blog. URL: https://modelon.com/blog/5-questions-to-ask-before-trusting-ai-related-simulation-results/.

Andreasson, Johan (2026b-03). *From Intent to Action: Agentic AI for Vehicle Dynamics in Modelon Impact*. Modelon blog. URL: https://modelon.com/blog/from-intent-to-action-agentic-ai-for-vehicle-dynamics-in-modelon-impact/.

Andreasson, Johan and Magnus Gäfvert (2005). "The VehicleDynamics Library — Overview and Applications". In: *Proceedings of the 5th International Modelica Conference*. Session 1b3. Vienna, Austria. URL: https://modelica.org/events/modelica2006/Proceedings/sessions/Session1b3.pdf.

Andreasson, Johan and Mats Jonasson (2008). "Vehicle Model for Limit Handling: Implementation and Validation". In: *Proceedings of the 6th International Modelica Conference*. Session 3c2. Bielefeld, Germany. URL: https://modelica.org/events/conference2008/sessions/session3c2.pdf.

Andreasson, Johan, Naoya Machida, Masashi Tsushima, John Griffin, and Peter Sundström (2016). "Deployment of high-fidelity vehicle models for accurate real-time simulation". In: *Proceedings of the 1st Japanese Modelica Conference*. Linköping Electronic Conference Proceedings 124. Tokyo, Japan: Linköping University Electronic Press, pp. 78–86. URL: https://ep.liu.se/en/conference-article.aspx?series=ecp&issue=124&Article_No=11.

Anthropic (2025). *Model Context Protocol — Specification*. Online. Revision 2025-11-25, accessed 2026-05. URL: https://modelcontextprotocol.io/specification/2025-11-25.

Fédération Internationale de l'Automobile (2025). *Formula 1 Circuit Guide – Season 2025/2026*. Online. Accessed 2026-05. URL: https://www.fia.com/events/fia-formula-one-world-championship.

Hou, Xinyi, Yanjie Zhao, Shenao Wang, and Haoyu Wang (2025). *Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions*. arXiv: 2503.23278 [cs.CR]. URL: https://arxiv.org/abs/2503.23278.

Li, Lixiang and Johan Andreasson (2026-03). *From Intent to Insight: AI-Driven, Physics-Based Modeling with Modelica*. Modelon blog. URL: https://modelon.com/blog/from-intent-to-insight-ai-driven-physics-based-modeling-with-modelica/.

Milliken, William F. and Douglas L. Milliken (1995). *Race Car Vehicle Dynamics*. Warrendale, PA: SAE International. ISBN: 978-1-56091-526-3.

DOI **10.3384/ECP219 215**

Modelon AB (2026a). *Modelon Impact: cloud-based system simulation platform*. Accessed 2026-05. URL: https://modelon.com/modelon-impact/.

Modelon AB (2026b). *Vehicle Dynamics Library Users Guide, v4.9*. Accessed 2026-05. URL: https://help.modelon.com/latest/library_documentation/users_guide/vehicledynamics/.

Pacejka, Hans B. (2012). *Tyre and Vehicle Dynamics*. 3rd ed. Oxford, UK: Butterworth-Heinemann. ISBN: 978-0-08-097016-5.

Sundström, Peter et al. (2016). "Virtual Vehicle Kinematics and Compliance Test Rig". In: *Proceedings of the 1st Japanese Modelica Conference*. Linköping Electronic Conference Proceedings 124. Tokyo, Japan: Linköping University Electronic Press, pp. 29–34. URL: https://ep.liu.se/en/conference-article.aspx?series=ecp&issue=124&Article_No=4.

DOI **10.3384/ECP219 215**

A Modelica-based Virtual Testbed for Hybrid-Cooled Data Centers

Hongjun Li[1] Viswanathan Ganesh[1] Wangda Zuo[1,2]

[1]Department of Architectural Engineering, The Pennsylvania State University, University Park, PA, USA.
{hongjun.li, viswanathan.ganesh, wangda.zuo}@psu.edu
[2]National Laboratory of the Rockies, Golden, CO, USA.

Abstract

Hybrid cooling systems that integrate air and liquid cooling are increasingly adopted in data centers to manage high-density and spatially nonuniform thermal loads. In these systems, the air- and liquid-cooling subsystems are dynamically coupled and exhibit different responses to variations in IT load, making coordinated control challenging. However, models capable of capturing the interactions between air- and liquid-cooling systems remain limited in the literature. To address this gap, this paper develops a hierarchical Modelica-based model spanning the system and rack levels. The system-level model is validated using open-source data from the Energy System Integration Facility at the National Laboratory of the Rockies. To demonstrate the model's application, a setpoint optimization case study is conducted to coordinate the liquid- and air-side cooling setpoints while retaining local PI control. Compared with fixed-setpoint operation, the optimized strategy maintains thermal constraints and reduces cooling energy consumption by 8.3%.

Keywords: Modelica, Virtual Testbed, Data Center Cooling, Hybrid cooling, Optimization

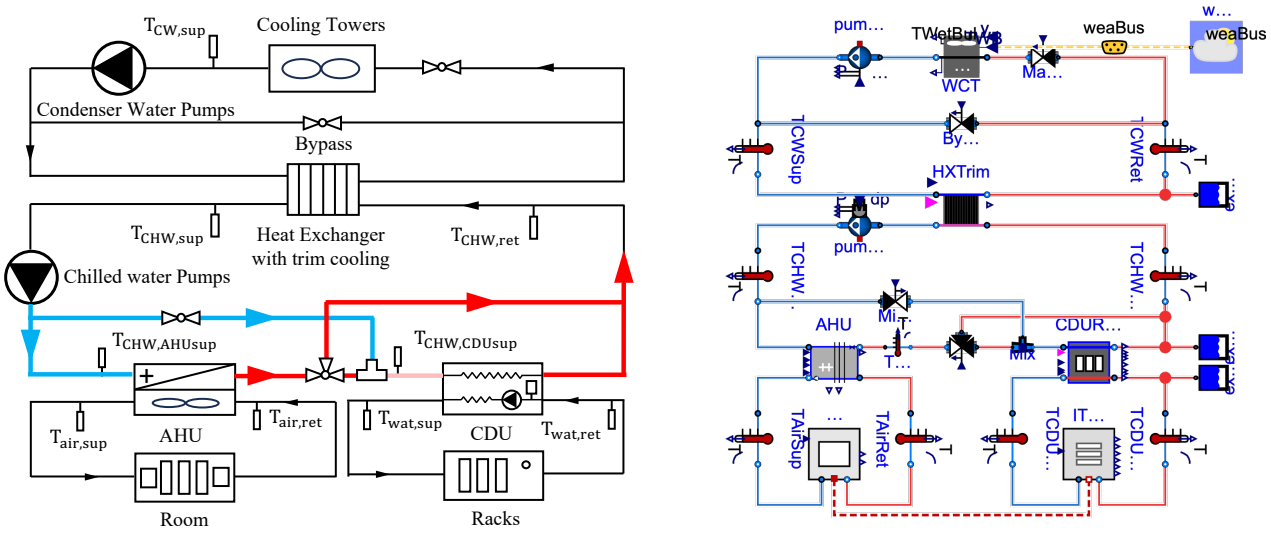
1 Introduction

The rapid growth of artificial intelligence (AI) and high-performance computing (HPC) is increasing both the magnitude and variability of data center heat loads (Chen et al. 2025). Conventional air cooling is increasingly constrained in high-density environments because it requires large airflow rates and is less effective at removing concentrated chip-level heat (Khalaj and Halgamuge 2017; Qu et al. 2024). In hybrid-cooled data centers, coolant distribution units (CDUs) typically remove approximately 60–90% of the IT heat load from high-power components such as CPUs and GPUs, while air-handling units (AHUs) manage the remaining 10–40% of heat (Van Zetten et al. 2026; Vertiv 2024). AHUs also provide humidity control, filtration, airflow management, and cooling for non-liquid-cooled equipment such as power distribution and networking devices. These two cooling paths are thermally coupled rather than independent (Gheni et al. 2026). Heat from the liquid-cooled loop can affect the air side through the server chassis, room air, and surrounding equipment surfaces. In addition, both loops are connected through the facility water system and central heat rejection plant, which further couples their operation and control.

Reliable thermal management in hybrid-cooled data centers requires coordinated control of both air- and liquid-side cooling systems. Previous studies have investigated data center cooling control in several contexts, including predictive control of the AHU (Lazic et al. 2018), multi-chiller plant control (Zhao et al. 2024), and optimization of hybrid liquid/free-air cooling with workload allocation (L. Li et al. 2014). Fu, Zuo, and Baker (2020) developed a real-time multi-market optimization framework for data centers, showing that coordinated data center operation can reduce energy costs while enabling participation in regulatory markets. These studies show that system-level coordination can improve cooling efficiency by accounting for interactions among cooling capacity, grid-service requirements, and workload-dependent heat generation. However, they primarily address air-side room control or central plant optimization. For hybrid-cooled data centers, the key control challenge is different: the air and liquid loops have different thermal time constants, share plant-level cooling resources, and are coupled through the rack, room air, and facility water system. Therefore, an air-side setpoint change can affect liquid-side operation, and a liquid-side setpoint change can alter air-side cooling demand and auxiliary power (Gheni et al. 2026). Under dynamic IT loads and varying weather conditions, these interactions can shift the optimal operating point and create tradeoffs between cooling energy and thermal reliability. Optimal coordinated control of air- and liquid-side setpoints remains insufficiently studied in this context, and a physically detailed hybrid-cooling testbed is needed to evaluate such strategies before deployment.

The complexity of the system and its control requirements motivate the development of a high-fidelity physical model capable of capturing the coupled thermal dynamics and equipment-level control behavior of hybrid-cooled data centers. Several modeling approaches have been proposed for examining data centers that combine air- and liquid-cooled IT equipment, but the cooling load is often assigned to the two systems using fixed percentages. Such simplified load-splitting assumptions limit the ability of these models to capture the dynamic response of IT loads and cooling systems (Stahlhut et al. 2025). More detailed data center cooling models have also been developed. For example, Gheni et al. (2026) proposed a data-center-level power and thermal model for the HAWK supercomputer.

DOI **10.3384/ECP219 226**



(a) Schematic of the hybrid cooling system **(b) Modelica implementation**

Figure 1. System-level schematic and corresponding Modelica implementation of the hybrid-cooled data center.

Although this model provides useful operational insights, its applicability is limited by its site-specific modeling approach, which is tailored to the HAWK system architecture and operating data. Therefore, applying it to different climate zones, equipment configurations, or control strategies would require substantial model reconfiguration and re-parameterization.

Modelica offers a promising platform for high-fidelity, equation-based modeling of such systems, particularly because it supports modular representation of coupled thermal-fluid components and control logic. Modelica has been widely used for dynamic building and HVAC simulation, while existing data center applications have primarily focused on air-cooled system configurations (Fu, Zuo, Wetter, et al. 2019; H. Li et al. 2026). In addition, Brewer et al. (2024) proposed an open-source digital twin framework for liquid-cooled supercomputers in which the liquid-cooling system is modeled in Modelica. Ganesh et al. (2026) presented a physics-informed reinforcement learning framework for economic-thermal co-optimization of liquid-cooled crypto-mining data centers using a Modelica-based model. However, the interaction between the liquid-cooling loop and the air-side cooling system is not fully represented. To the best of the authors' knowledge, no readily available Modelica-based testbed for hybrid-cooled data centers currently satisfies all of the following requirements: a) representation of a state-of-the-art hybrid cooling architecture, b) explicit modeling of the coupled thermal dynamics between air and liquid-cooling loops, and c) incorporation of equipment-level control logic.

This paper addresses these gaps by developing a Modelica-based virtual testbed for a hybrid-cooled data center and using it to evaluate coordinated supervisory set-point optimization. The main contributions are as follows: (1) a rack-to-system-level Modelica virtual testbed is developed for a hybrid-cooled data center based on the NLR data center, incorporating rack-room and plant-level coupling; (2) the testbed is validated using measured data; and (3) a coordinated air- and liquid-side setpoint optimization strategy is evaluated against fixed-setpoint operation in terms of thermal reliability and cooling energy.

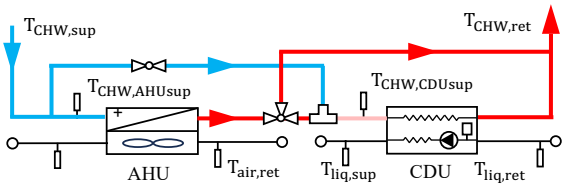
The remainder of this paper is organized as follows. Section 2 describes the Modelica-based virtual testbed. Section 3 presents the validation results using measured data. Section 4 provides a case study on coordination optimization and discusses the resulting performance. Finally, Section 5 concludes the paper and outlines future work.

2 Virtual Testbed Implementation

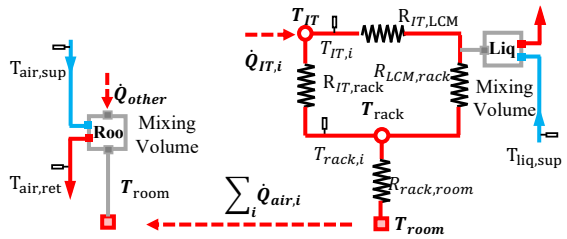
2.1 Overview of hybrid-cooled data centers

Figure 1 presents the framework of the Modelica-based virtual testbed developed using the open-source NLR data (Sickinger et al. 2018). From the data, the liquid-side load represents the dominant thermal disturbance and accounts for a major portion, around 80% of the IT heat load. The air- and liquid-side cooling loops are thermally coupled: changes in the liquid-cooled IT load can transfer heat to the air side through room air and server chassis (Gheni et al. 2026). As a result, increasing the liquid-side temperature setpoint can also affect air-side cooling energy. In addition, high-density data centers may experience large step changes in IT load, while weather conditions and IT load profiles are not synchronized. These characteristics make optimized setpoint control important for both energy efficiency and thermal reliability.

DOI 10.3384/ECP219 226



(a) Plant-level coupling through the shared chilled-water loop



(b) Rack–room thermal coupling

Figure 2. Thermal coupling in the hybrid-cooled data center model.

2.2 System description

A reliable virtual testbed is required before optimized setpoint control can be evaluated. Figure 1 shows the schematic of the hybrid cooling system considered in this study. The model is developed under the following assumptions. First, the measured air- and liquid-side heat rejection rates, $\dot{Q}_{\mathrm{air}}$ and $\dot{Q}_{\mathrm{liq}}$, are summed as the total IT heat load. The rack–room coupling model dynamically partitions this load between the two cooling paths, and the measured heat rejection rates are used for validation. Second, the data center is classified as H1 and W27 according to the ASHRAE Thermal Guidelines (ASHRAE TC9.9 2021), so the air and liquid supply temperatures are constrained below 22°C and 27°C, respectively. Third, a chiller is included to provide trim cooling when the cooling tower cannot fully satisfy the cooling load.

2.3 Model implementation

Based on the system architecture shown in Figure 1, a hierarchical Modelica virtual testbed was developed to represent the hybrid-cooled data center from the plant level down to the CDU and rack levels. The implementation follows the equation-based and object-oriented modeling paradigm of Modelica, where physical subsystems are assembled through acausal fluid and thermal connectors rather than prescribed signal-flow relationships. Standard hydraulic, thermal, heat-exchanger, pump, valve, and sensor components from the Modelica Buildings Library (MBL) (Wetter et al. 2014) are reused where applicable, while data-center-specific components that are not directly available in the library are implemented as custom reusable models.

At the system level, the chilled-water and condenser-water loops, cooling tower, trim-cooling heat exchanger, AHU, CDU bank, and associated pumps and valves are interconnected to reproduce the hybrid cooling topology shown in Figure 2. The AHU and CDU subsystems share the chilled-water loop, thereby introducing plant-level thermal and hydraulic coupling between the air- and liquid-cooling paths. The model also includes the anti-freeze control loop and a mixed-supply cascade configuration for operation across different outdoor conditions. Local control loops are implemented using Modelica signal connectors and PI-based controllers for supply-temperature regulation, pump operation, and valve modulation. The liquid-cooling subsystem is modeled hierarchically, as illustrated in Figure 3. The parallel-CDU model contains multiple identical CDU branches instantiated as an array, with each branch receiving supervisory primary- and secondary-flow commands and exchanging heat through a primary-to-secondary heat exchanger. Each CDU is further decomposed into a primary-side control valve, a secondary-side circulation pump, a coolant-handling manifold, and a heat exchanger. This structure allows the number of operating CDU/rack branches and branch-level flow allocation to be represented explicitly while preserving a common plant-side chilled-water connection.

Table 1. Principal model parameters.

Parameter	Symbol	Value
CDU/rack branches	N_{CDU}	7
Air mass flow rate	$\dot{m}_{\mathrm{air,nom}}$	18.6 kg/s
CHW mass flow rate	$\dot{m}_{\mathrm{CHW,nom}}$	20 kg/s
Rack coolant flow rate	$\dot{m}_{\mathrm{rack,nom}}$	15 kg/s
Rack coolant density	ρ_{rack}	1030 kg/m^3
Coolant residence time	τ_{rack}	20 s
IT thermal capacitance	C_{IT}	2.0×10^5 J/K
Rack thermal capacitance	C_{rack}	1.0×10^5 J/K
IT–LCM resistance	$R_{\mathrm{IT,LCM}}$	1.5×10^{-4} K/W
IT–rack resistance	$R_{\mathrm{IT,rack}}$	1.0×10^{-3} K/W
LCM–rack resistance	$R_{\mathrm{LCM,rack}}$	5.0×10^{-3} K/W
Rack–room resistance	$R_{\mathrm{rack,room}}$	8.0×10^{-5} K/W

At the rack level, a reduced-order thermal network is introduced to capture the dynamic partitioning of IT heat between the liquid- and air-cooling paths. Instead of prescribing a fixed liquid-cooling fraction, the total IT heat load is applied to an equivalent IT thermal node with capacitance C_{IT}. Heat is transferred from the IT node to the liquid-cooling module and to the rack enclosure through the equivalent resistances $R_{\mathrm{IT,LCM}}$ and $R_{\mathrm{IT,rack}}$, respectively. The rack enclosure is represented by a second thermal capacitance C_{rack} and is coupled to the room air through $R_{\mathrm{rack,room}}$. A weak thermal path between the liquid-cooling module and rack enclosure is represented by $R_{\mathrm{LCM,rack}}$. The liquid-side thermal dynamics are coupled directly to the rack coolant control volume through the fluid heat port, while the rack-to-room heat flow is transferred through a thermal connector to the room mixing volume. Consequently, the liquid- and air-side heat-rejection rates are model outputs determined by the instan-

DOI 10.3384/ECP219 226

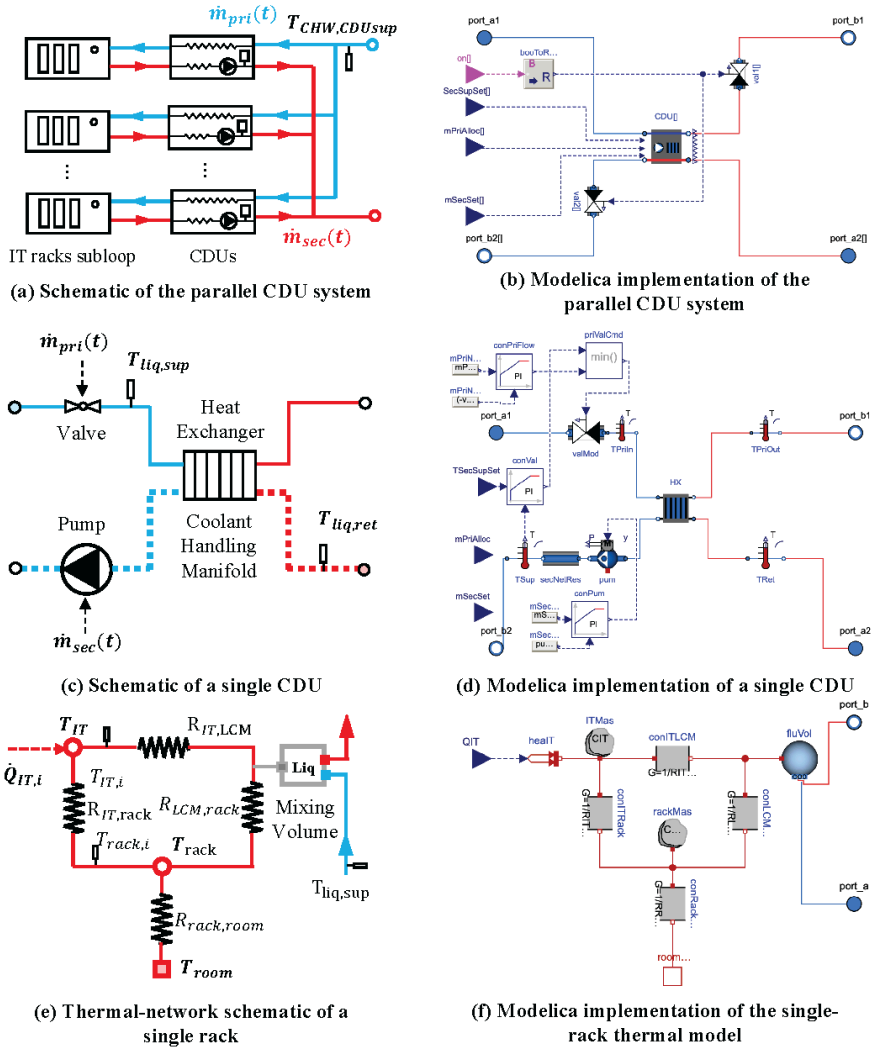


Figure 3. Schematic representations and corresponding Modelica implementations of the CDU and rack subsystems.

taneous thermal states and resistances rather than predefined fractions. The parallel-rack model aggregates multiple single-rack instances and connects each rack simultaneously to its CDU secondary loop and to the common room thermal node. This establishes rack–room thermal coupling at the equipment level, while the shared chilled-water loop establishes coupling between the AHU and CDU systems at the plant level. The principal design and reduced-order thermal parameters used in the model are summarized in Table 1.

3 Validation

The reduced-order thermal parameters were determined prior to validation. Parameters available from the original system model were retained, while selected equivalent thermal resistances governing the rack-level heat partition were tuned offline against measured heat-rejection and temperature data. Once determined, all parameters were fixed throughout the annual validation and control case study. The Modelica testbed is validated against the measured data using normalized mean bias error (NMBE) and coefficient of variation of the root mean square error (CVRMSE) (ASHRAE 2023).

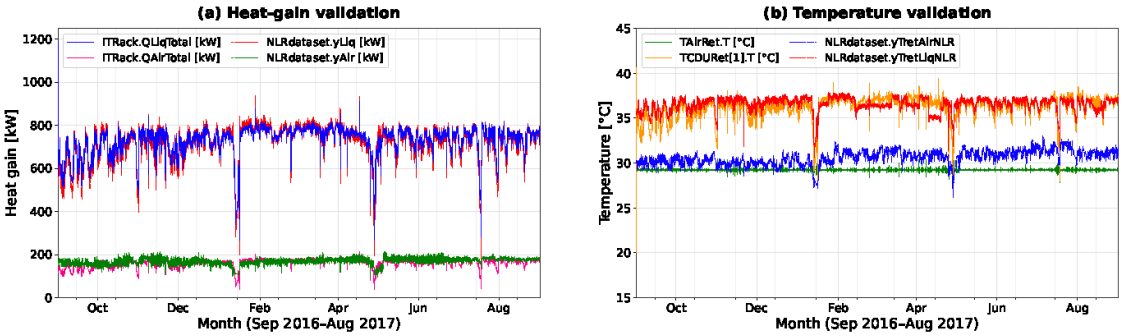
$$\mathrm{NMBE} = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)}{n\bar{y}} \times 100\%, \tag{1}$$

$$\mathrm{CVRMSE} = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}}{\bar{y}} \times 100\%, \tag{2}$$

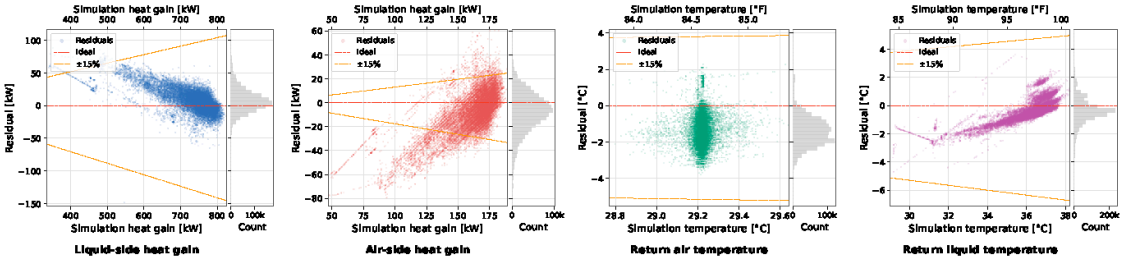
where y_i is the measured value, $\hat{y}_i$ is the simulated value, $\bar{y}$ is the mean of the measured values, and n is the number of data points.

The annual simulation was conducted using constant supply-temperature setpoints and the measured total IT

DOI 10.3384/ECP219 226



(a) Annual comparison of measured and simulated heat gains and return temperatures from September 2016 to August 2017.



(b) Residual distributions for liquid-side heat gain, air-side heat gain, return air temperature, and return liquid temperature predictions.

Figure 4. Model validation of the virtual testbed.

heat load derived from the open-source dataset from the Energy System Integration Facility at the National Laboratory of the Rockies. As shown in Figure 4a, the model reproduces the annual variations in both air- and liquid-side heat gain and captures the corresponding return-temperature trends. Figure 4b further compares the residual distributions for the four validation variables. The liquid-side heat gain and return liquid temperature show relatively small deviations, while larger variability is observed for the air-side heat gain and return air temperature. Quantitatively, Table 2 gives NMBE values of −1.12%, 4.76%, 4.62%, and 0.95%, with corresponding CVRMSE values of 2.68%, 11.31%, 5.29%, and 2.35%, respectively. Overall, the results indicate that the virtual testbed captures the annual heat-partition and thermal response of the hybrid-cooled data center with limited systematic bias.

Table 2. Performance metrics of the virtual testbed on the minute-level validation dataset.

Variable	*NMBE*	*CVRMSE*
Liquid-side heat gain	−1.12%	2.68%
Air-side heat gain	4.76%	11.31%
Return air temperature	4.62%	5.29%
Return liquid temperature	0.95%	2.35%

4 Case Study

4.1 Coordinated setpoint optimization

The purpose of this optimization is to quantify the benefit of coordinating multiple cooling loops in a hybrid-cooled data center. A bounded grid-search procedure (Lerman 1980) is used as a transparent and robust method to compare different levels of supervisory coordination using the validated Modelica virtual testbed.

The validated Modelica testbed is exported as a co-simulation Functional Mock-up Unit (FMU) (Broman et al. 2013) and coupled with a Python-based grid-search algorithm. The FMU captures the dynamic response of the hybrid cooling system, including the local PI controllers, cooling equipment, and rack thermal loads. At each control interval, the grid search evaluates candidate air- and liquid-side temperature setpoints and selects the feasible combination with the lowest cooling energy consumption. The embedded PI controllers continue to regulate equipment-level variables such as pump speed, valve position, and cooling capacity.

To evaluate the value of coordination, two control cases are considered. The first case is a fixed-setpoint baseline, where all supervisory setpoints remain constant. The second case is coordinated optimization, where the liquid-side and air-side setpoints are optimized together. This comparison is used to determine whether multi-loop coordination provides additional energy or reliability benefits.

For the coordinated optimization case, the supervisory

DOI 10.3384/ECP219 226

decision vector is defined as

$$x(t_k) = \left[T^{set}_{liq,sup}(t_k), T^{set}_{air,sup}(t_k)\right]^T, \quad (3)$$

where $T^{set}_{liq,sup}$ is the liquid-loop supply temperature setpoint, and $T^{set}_{air,sup}$ is the air-side supply temperature setpoint. These variables represent practical supervisory coordination points associated with the CDU secondary loop and the air-side cooling loop.

For the coordinated case, the candidate setpoint grid is defined as:

$$G = G_{liq} \times G_{air}, \quad (4)$$

where G_{liq} and G_{air} denote the discrete candidate sets for the liquid- and air-side supply-temperature setpoints, respectively. Based on the operating ranges assumed in Section 2.2, $G_{\text{air}} = \{21, 21.5, 22\}$°C and $G_{\text{liq}} = \{23, 24, 25, 26, 27\}$°C.

The grid values are selected according to the allowable operating ranges of the cooling equipment and the thermal reliability limits used in the case study. At each supervisory interval t_k, Python saves the current FMU state and evaluates each candidate setpoint vector $x \in G$. For each candidate, the FMU is simulated over a short evaluation horizon H, starting from the same state at t_k. The FMU reports the cooling power trajectory and thermal responses, including rack inlet air temperature and liquid return temperature.

The objective is to minimize cooling energy over the evaluation horizon:

$$E_{cool}(x, t_k) = \int_{t_k}^{t_k+H} P_{cool}(\tau)\, d\tau. \quad (5)$$

The total cooling power is calculated from the FMU outputs:

$$P_{cool}(\tau) = \sum_{j \in C} P_j(\tau), \quad (6)$$

The selected candidate must satisfy the thermal reliability constraints over the evaluation horizon:

$$T_{air,in,i}(\tau) \le T^{max}_{air,in}, \ \tau \in [t_k, t_k + H], \quad (7)$$

$$T_{liq,ret}(\tau) \le T^{max}_{liq,ret}, \ \tau \in [t_k, t_k + H]. \quad (8)$$

The candidate must also remain within the allowable equipment operating range:

$$x^{min} \le x \le x^{max}. \quad (9)$$

The optimal supervisory setpoint is selected as the lowest-energy feasible candidate:

$$x^*(t_k) = \arg\min_{x \in G} E_{cool}(x, t_k). \quad (10)$$

If no feasible candidate is available, the most conservative cooling setpoint in the grid is applied to maintain thermal safety. After the optimal candidate is selected, the FMU state is restored to t_k, and $x^*(t_k)$ is applied to the FMU for the next supervisory interval. The embedded PI controllers then track the updated setpoints. This process is repeated throughout the simulation period.

4.2 Results

4.2.1 Thermal response to load variations

Figure 4 illustrates the distinct dynamic responses of the liquid- and air-side cooling loops to variations in IT load. January 16 is selected as a representative case because it contains several characteristic load ramp-up events.

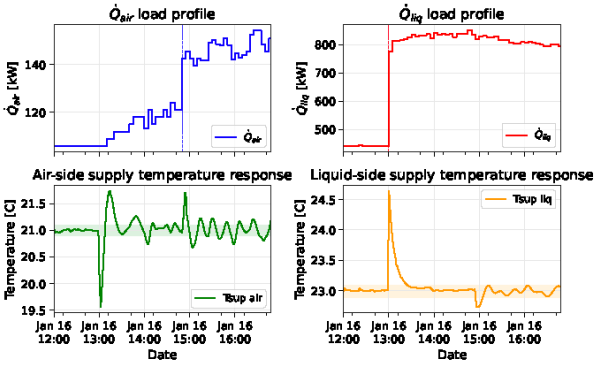


Figure 5. Ramp-up event and thermal response of the air- and liquid-side cooling loops.

$\dot{Q}_{\text{liq}}$ increases stepwise from 440 kW to 780 kW, and the liquid-side supply temperature $T_{\text{sup,liq}}$ briefly overshoots to approximately 24.5°C before settling near 23.0°C within several minutes. This indicates a strong liquid-side sensitivity to abrupt load disturbances. By contrast, $\dot{Q}_{\text{air}}$ increases more gradually from 105 kW to 155 kW, and $T_{\text{sup,air}}$ remains close to 21.0°C with smaller fluctuations. This smoother response is due not only to the smaller change in $\dot{Q}_{\text{air}}$, but also to the larger thermal inertia of the air-side data hall. The air-side temperature also responds to changes in $T_{\text{sup,liq}}$, which reveals coupling between the two loops in the mixed-supply cascade cooling system. Since both supply temperatures return to their stable operating ranges after the disturbance, the control system demonstrates effective disturbance rejection and thermal recovery. These results support coordinated AHU and CDU optimization for hybrid-cooled data centers.

4.2.2 Optimized dynamic setpoints

A 15-min interval is adopted for supervisory setpoint adjustment, consistent with the characteristic response time of the cooling system (Yu et al. 2026). Figure 6 compares the fixed and optimized setpoints over a 24-h operating period on August 26, selected to illustrate the supervisory control response under varying weather and cooling-load conditions. The figure also shows the outdoor wet-bulb temperature in Denver, located in ASHRAE Climate Zone 5B (Cool–Dry), and the cooling loads $\dot{Q}_{\text{air}}$ and $\dot{Q}_{\text{liq}}$. At each control interval, the supervisory setpoint vector x, consisting of $T_{\text{air,set}}$ and $T_{\text{liq,set}}$, is selected as the feasible candidate that minimizes cooling energy consumption, $x^*(t_k) = \arg\min_{x \in G} E_{\text{cool}}(x, t_k)$, subject to the thermal reliability constraints and allowable operating ranges. As a result, lower wet-bulb temperature and reduced cooling load do not necessarily correspond to the highest op-

DOI 10.3384/ECP219 226

timal setpoints. Higher setpoints are selected only when they provide a sufficient reduction in cooling energy while maintaining thermal feasibility. During this low-load period, the chiller operates at a relatively low load, so further increasing the setpoints yields only limited chiller energy savings while increasing AHU fan energy consumption.

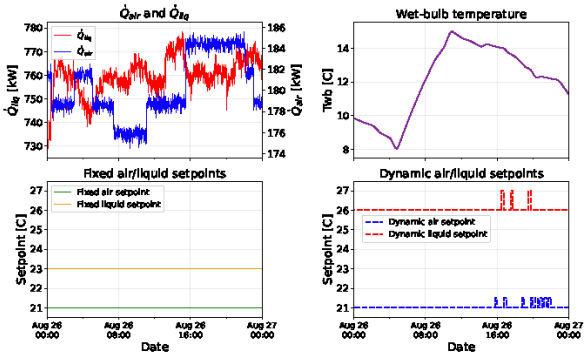


Figure 6. Optimized dynamic supply-temperature setpoints under dynamic weather and cooling loads.

4.2.3 Thermal constraint reliability

The optimized strategy maintains thermal reliability while allowing short-term setpoint adjustments to reduce cooling energy. As shown in Figure 7, the fixed strategy keeps both air- and liquid-side supply temperatures nearly constant and well below their prescribed limits. In contrast, the dynamic strategy introduces larger transient variations as the setpoints adapt to changing cooling conditions. Nevertheless, both temperatures remain within the allowable ranges, indicating that the optimized control achieves greater operational flexibility without compromising thermal constraints.

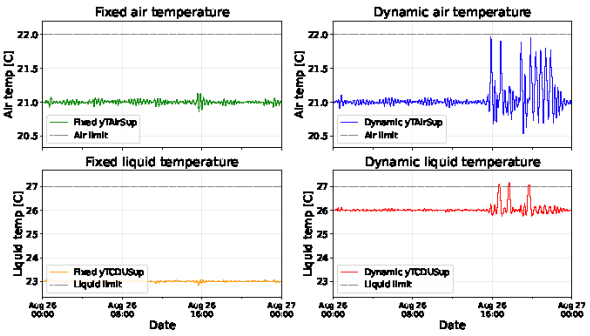


Figure 7. Thermal constraint reliability under fixed and optimized dynamic setpoints.

4.2.4 Energy analysis

The cooling tower alone cannot reject the full cooling load during this period, and the chiller provides trim cooling to maintain the required thermal conditions. As shown in Figure 8, the optimized strategy reduces total cooling energy by 8.3%. The dominant saving comes from the chiller, whose energy consumption decreases from 1121 kWh to 897 kWh, corresponding to a reduction of 19.9%. The AHU fan, cooling tower fan, condenser-water pump, and CDU pump energy consumptions remain nearly unchanged. In contrast, the chilled-water pump energy increases from 152 kWh to 177 kWh, due to the higher chilled-water mass flow required to maintain the thermal constraints.

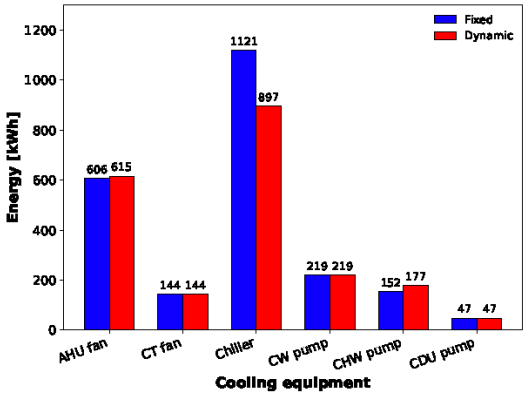


Figure 8. Cooling energy breakdown by equipment under fixed and optimized dynamic setpoints.

5 Conclusion and Future Work

This study developed a Modelica-based virtual testbed for a hybrid-cooled data center and demonstrated its capability to capture the coupled thermal dynamics of the air- and liquid-cooling subsystems. The main findings are summarized as follows:

(1) The ramp-up analysis shows that the liquid-side loop responds more strongly to IT-load variations, while the air-side loop exhibits smaller and slower temperature changes.

(2) The coupled transient responses indicate that the air- and liquid-side loops should be coordinated rather than operated independently.

(3) In the case study, the proposed dynamic setpoint optimization strategy reduced total cooling energy consumption by 8.3% compared with fixed-setpoint operation while maintaining the required thermal constraints. The energy savings were primarily attributed to a reduction in chiller energy consumption from 1121 kWh to 897 kWh, corresponding to a 19.9% decrease.

Future work will focus on extending the scope and fidelity of the virtual testbed. First, additional temporally varying and spatially non-uniform IT load profiles will be evaluated to assess the robustness of the proposed modeling and control framework under more representative computing workloads. Second, the testbed will be extended toward chip-to-facility thermal modeling and validation to better characterize the interactions among chip-level heat generation, rack-level liquid cooling, room-level air cooling, and facility-level cooling infrastructure. Third, advanced control approaches, includ-

DOI 10.3384/ECP219 226

ing multi-agent reinforcement learning, will be investigated for coordinated operation of multiple AHUs and CDUs.

6 Acknowledgment

This research was supported by the National Science Foundation, United States, under Award No. ECCS-2446782. This research was also supported by a 2025/2026 Rising Researcher Grant from Penn State's Institute for Computational & Data Sciences (RRID: SCR_025154). This manuscript has been authored by UT-Battelle LLC under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). This work was also authored in part by the National Laboratory of the Rockies for the US DOE under Contract No. DE-AC36-08GO28308. The US government retains and the publisher, by accepting the article for publication, acknowledges that the US government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for US government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (https://www.energy.gov/doe-public-access-plan).

References

ASHRAE (2023). *ASHRAE Guideline 14-2023: Measurement of Energy, Demand, and Water Savings*. ASHRAE. Peachtree Corners, GA.

ASHRAE TC9.9 (2021). *Thermal Guidelines for Data Processing Environments*. 5th. Atlanta, GA, USA: ASHRAE.

Brewer, Wesley et al. (2024). "A digital twin framework for liquid-cooled supercomputers as demonstrated at exascale". In: *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, pp. 1–18.

Broman, David, Christopher Brooks, Lev Greenberg, Edward A Lee, Michael Masin, Stavros Tripakis, and Michael Wetter (2013). "Determinate composition of FMUs for co-simulation". In: *2013 Proceedings of the International Conference on Embedded Software (EMSOFT)*. IEEE, pp. 1–12.

Chen, Xin, Xiaoyang Wang, Ana Colacelli, Matt Lee, and Le Xie (2025). "Electricity demand and grid impacts of AI data centers: Challenges and prospects". In: *arXiv preprint arXiv:2509.07218*.

Fu, Yangyang, Wangda Zuo, and Kyri Baker (2020). "Multi-market optimization of a data center without storage systems". In: *The American Modelica Conference*, pp. 23–25.

Fu, Yangyang, Wangda Zuo, Michael Wetter, James W VanGilder, and Peilin Yang (2019). "Equation-based object-oriented modeling and simulation of data center cooling systems". In: *Energy and Buildings* 198, pp. 503–519.

Ganesh, Viswanathan, Hongjun Li, Jiyuan Su, and Wangda Zuo (2026). "A Physics-Informed Reinforcement Learning Framework for Economic-Thermal Co-Optimization of Crypto Mining Data Centers". In: *Proceedings of the 2026 IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*. Accepted for publication.

Gheni, Mashhur, Henner Kerskes, and Konstantinos Stergiaropoulos (2026). "Operational analysis of the cooling system in a direct liquid-cooled data center: a measurement and simulation study on the impact of supply water temperature". In: *Applied Energy* 403, p. 127061.

Khalaj, Ali Habibi and Saman K Halgamuge (2017). "A Review on efficient thermal management of air-and liquid-cooled data centers: From chip to the cooling system". In: *Applied energy* 205, pp. 1165–1188.

Lazic, Nevena, Craig Boutilier, Tyler Lu, Eehern Wong, Binz Roy, MK Ryu, and Greg Imwalle (2018). "Data center cooling using model-predictive control". In: *Advances in Neural Information Processing Systems* 31.

Lerman, PM (1980). "Fitting segmented regression models by grid search". In: *Journal of the Royal Statistical Society Series C: Applied Statistics* 29.1, pp. 77–84.

Li, Hongjun, Sen Huang, and Wangda Zuo (2026). "Control-Oriented Prototype Energy Model for Data Centers". In.

Li, Li, Wenli Zheng, Xiaodong Wang, and Xiaorui Wang (2014). "Coordinating liquid and free air cooling with workload allocation for data center power minimization". In: *11th international conference on autonomic computing (ICAC 14)*, pp. 249–259.

Qu, Shengli, Kaiwen Duan, Yuxiang Guo, Yiwei Feng, Chuang Wang, and Ziwen Xing (2024). "Real-time optimization of the liquid-cooled data center based on cold plates under different ambient temperatures and thermal loads". In: *Applied Energy* 363, p. 123101.

Sickinger, David E, Otto D Van Geet, Suzanne A Belmont, Thomas Carter, and David Martinez (2018). *Thermosyphon cooler hybrid system for water Savings in an Energy-Efficient HPC data center: Results from 24 months and the impact on water usage effectiveness*. Tech. rep. National Renewable Energy Laboratory (NREL), Golden, CO (United States).

Stahlhut, Maximilian, Markus Sekulla, Peter Großöhme, Axel Auweter, and Thorsten Urbaneck (2025). "Data Centers With Direct Liquid-Cooled Servers: Experimental Analysis and Modeling". In: *Energy Science & Engineering* 13.7, pp. 3605–3619.

Van Zetten, Jeffrey D, Michael E Cholette, and Keivan Bamdad (2026). "Energy and carbon savings in data centres through liquid-to-chip cooling with differential temperature control". In: *Advances in Applied Energy*, p. 100269.

Vertiv (2024). *Vertiv™ 360AI: High Density Reference Design for AI, AI Reference Design #014*. Reference Design GR-00101 (R6/24). Accessed 2026-05-31. Vertiv. URL: https://www.vertiv.com/48eb98/globalassets/documents/ai-hub-reference-design/vertiv-360ai-reference-design-014.pdf.

Wetter, Michael, Wangda Zuo, Thierry S Nouidui, and Xiufeng Pang (2014). "Modelica buildings library". In: *Journal of Building Performance Simulation* 7.4, pp. 253–270.

Yu, Min Gyung, Alex Vlachokostas, Karthik Devaprasad, Matt Cornachione, Stephanie Johnson, Tim A Yoder, and Timothy I Salsbury (2026). "Optimizing chilled water systems with cooling towers via virtual power metrics and extremum-seeking control". In: *Journal of Building Performance Simulation* 19.2, pp. 181–194.

Zhao, Jing, Ziyi Chen, Haonan Li, and Dehan Liu (2024). "A model predictive control for a multi-chiller system in data center considering whole system energy conservation". In: *Energy and Buildings* 324, p. 114919.

DOI **10.3384/ECP219 226**

Integrated Microclimate Model in Modelica for Data Center Chiller Farms

Kaustubh Pradeep Phalak, Pouya Javidmand, Gang Wang, Zhanwei He, Zhong Guo

Trane, USA, Pouya.Javidmand@tranetechnologies.com

Abstract

Data centers are increasing in cooling density while also facing tighter energy and operational flexibility requirements. Air cooled chiller plants are attractive for many data center applications because they reduce/no water use and can be deployed modularly, but closely spaced chillers can experience hot condenser exhaust recirculation. This recirculation raises local condenser entering air temperature, reduces coefficient of performance, and can create unequal capacity and efficiency penalties across a chiller array.

Computational fluid dynamics (CFD) remains the preferred tool for detailed airflow design studies, but repeated CFD simulations are computationally expensive for long horizon weather simulation, sequence evaluation, and control co-simulation. This paper presents a reduced order Gaussian plume-based microclimate model implemented in Modelica and coupled with a data center chiller farm model. The chiller farm computes time varying heat rejection and condenser air flow for each chiller. The microclimate block receives those source terms together with ambient temperature, wind speed, and wind direction, then computes local condenser inlet conditions across a m x n chiller farm. The model is calibrated using CFD generated data and validated against additional CFD cases (both exclusive datasets).

An illustrative dynamic simulation shows that, at an ambient temperature of 85°F and wind speed of approximately 16 m/s, the hottest downstream condenser inlet reaches about 102.8°F and the COP difference between favorable and unfavorable microclimate positions is approximately 11.8%. Current validation error is still higher than desired for strong quantitative predictive claims, but the framework already provides a fast, physically interpretable basis for plant simulation, recirculation diagnostics, sequence studies, and future recirculation aware controls development.

Keywords: data center cooling, air cooled chiller, microclimate, Gaussian plume, reduced order model

1 Introduction

Modern data centers are placing new demands on cooling infrastructure because artificial intelligence, high performance computing, and cloud workloads are increasing both total heat rejection and local power density. Air cooled chiller farms are frequently considered for these sites because they are modular, avoid large evaporative water consumption, and can be integrated into plant configurations with high reliability requirements. Their performance, however, depends strongly on the temperature of the air entering each condenser. When condenser entering air temperature rises, condensing pressure and compressor lift increase, cooling capacity can decrease, and plant energy use and power usage effectiveness can worsen.

$$T_{CondIn} \uparrow \rightarrow COP_{chiller} \downarrow$$

The ambient air temperature measured at a weather station is often not the temperature seen by every condenser in a dense outdoor chiller yard. Each operating chiller discharges hot air upward through condenser fans. Under calm conditions, buoyancy and fan momentum can carry the plume upward. Under wind, the same plume can bend across the yard and enter the suction region of downstream machines. The result is a spatially varying microclimate in which chillers in the same plant, operating under the same nominal ambient weather condition, can experience materially different condenser inlet temperatures.

CFD is well suited to resolving these 3D flow interactions and remains the preferred tool for detailed design validation of chiller yard geometry, screens, nearby buildings, and wind exposure. The difficulty is that CFD is not well matched to repeated plant level simulation. A single chiller yard CFD case can take on the order of two to four hours after wind condition, ambient temperature, geometry, and loading are specified. Changes in wind speed, wind direction, ambient condition, spacing, chiller geometry, or loading distribution generally require separate simulations. This cost is acceptable when the objective is detailed design evaluation, but it is not acceptable when the objective is to run many weather hours, compare many chiller sequences, or evaluate controller behavior in closed loop simulation.

DOI 10.3384/ECP219 234

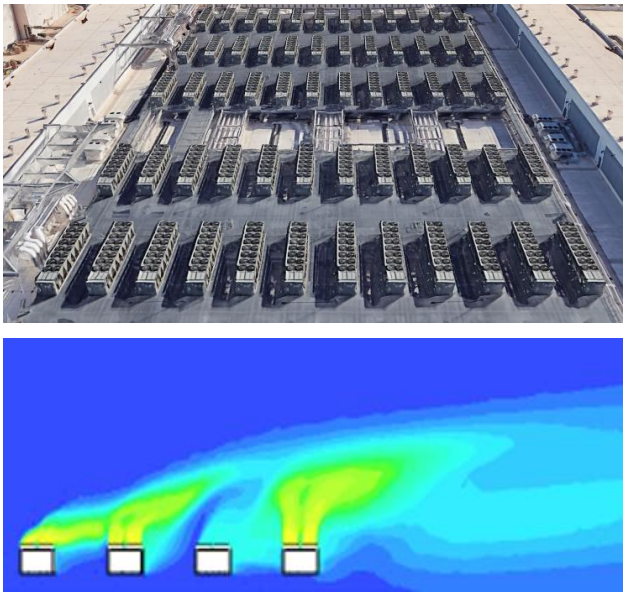


Figure 1. Representative data center chiller farm and illustrative CFD simulation of hot condenser exhaust recirculation across the yard.

This paper targets the gap between detailed CFD and conventional plant simulation. The objective is not to replace CFD for final design studies. Instead, the objective is to create a reduced order microclimate model that is fast enough to run inside a Modelica chiller farm simulation while retaining the dominant dependencies needed for controls oriented and long horizon studies. The formulation uses a Gaussian plume inspired source target model with heat rejection, condenser air flow, wind advection, buoyancy influence, spatial decay, and superposition across the chiller array. Because this compact representation necessarily omits many site specific flow details, its empirical coefficients are calibrated using CFD generated data and then evaluated against additional CFD cases.

The main contribution is therefore a Modelica integrated framework in which local condenser inlet temperatures are no longer assumed to be identical to ambient air temperature. The microclimate block receives dynamic source terms from the chiller farm model and returns spatially resolved inlet temperatures to each chiller. This enables plant simulation to represent the operational consequence of hot air recirculation: different machines in the same yard can have different efficiency, capacity, and sequencing value under the same weather condition.

2 Background and Modeling Gap

Air cooled chiller performance ratings and manufacturer maps show a strong dependence on condenser entering air temperature. AHRI 550/590 and 551/591 provide standardized performance rating context for vapor compression chiller equipment [1], while data center thermal guidance from ASHRAE TC 9.9 emphasizes the importance of reliable thermal management in continuously operating data processing environments [2]. Data center energy flow treatments further connect heat rejection performance to facility level cooling efficiency and operating margin [8]. For plant simulation, the relevant implication is direct: if local condenser inlet temperature is underestimated, cooling capacity may be overpredicted, and compressor power may be underpredicted.

The physical behavior of chiller exhaust is related to the broader literature on buoyant plumes and atmospheric dispersion. Morton et al. [3] established foundational theory (figure 2) for turbulent buoyant plumes, Briggs developed practical plume rise correlations [4], and Gaussian plume modeling practice is summarized by Turner and by Venkatram and J. The [5,6]. Small scale Gaussian plume studies, such as Brusca et al., show how simplified dispersion formulations can provide useful engineering estimates when calibrated and interpreted within their domain of validity [7].

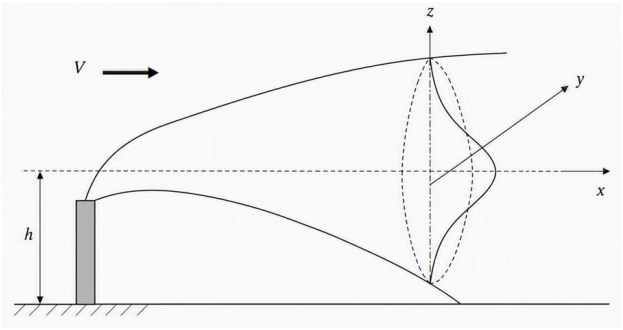


Figure 2. Conceptual representation of a wind advected buoyant plume

Figure 2 shows stack height, buoyant rise, downwind advection, and crosswind spreading that supports the reduced order source target decay formulation. The chiller yard problem differs from classical far field atmospheric dispersion in several important ways. The region of interest is near the equipment, where plume dimensions are comparable to chiller spacing. Condenser fan discharge, local suction at condenser inlets, equipment geometry, and wind direction can dominate the recirculation pattern. These details are precisely why CFD remains necessary for detailed design studies. At the same time, plant control and system simulation need models that can be evaluated many times across weather files, load profiles, and candidate operating sequences. A reduced order model is useful when it preserves the leading physical dependencies and makes its calibration status explicit.

Modelica is a natural environment for this type of coupled analysis because it supports modular system simulation across thermal, fluid, electrical, and control domains. The

DOI 10.3384/ECP219 234

Modelica Buildings Library, for example, has established Modelica as a practical environment for building and plant system simulation [9]. Existing Modelica chiller plant models can represent equipment behavior, hydronic loops, supervisory controls, and weather inputs, but they often provide a single ambient air temperature signal to all air-cooled chillers. That assumption removes the array level microclimate effect. The model presented here adds a microclimate layer between weather data and individual chiller condenser inputs so that local recirculation effects can be evaluated within the same plant simulation.

3 Modelica Coupling and Modeling Intent

The coupled model separates plant themo-fluid simulation from yard microclimate physics (figure 3). The chiller farm model computes the dynamic operating state of each chiller, including heat rejection and condenser air flow. The microclimate block then estimates how those thermal source terms are transported across the yard by wind and buoyancy. Its principal output is a matrix of condenser entering air temperatures, one value for each chiller.

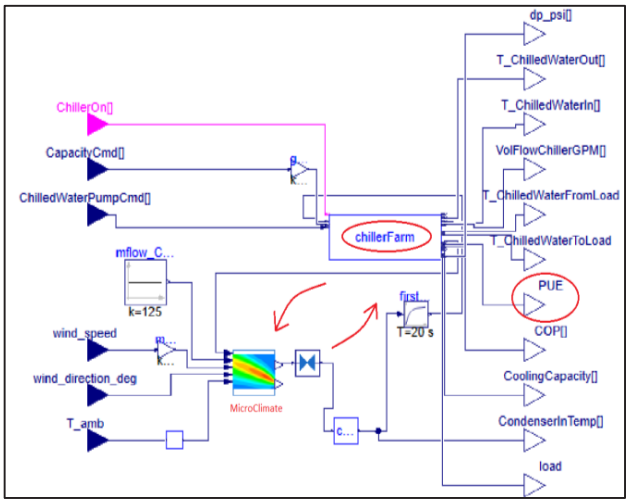


Figure 3. Modelica Integration of the Microclimate Block with the Data Center Chiller Farm Model

This separation is important for both clarity and reuse. The microclimate block does not estimate chiller part load ratio, compressor power, or plant heat rejection internally. Instead, it receives *QCon_flow[i,j]* and *mConAir_flow[i,j]* from the chiller farm model, along with *TAirAmb, vWin,* and *winDir* from the weather boundary condition. It returns *TAirIn[i,j]* to the chiller model. This architecture allows the same microclimate component to be used in sequence studies, long horizon simulations, and future controller development while allowing the chiller model to determine how inlet temperature affects capacity and coefficient of performance.

In closed loop plant studies, a first order lag can be placed in the feedback path if needed to avoid a hard algebraic loop between chiller performance, heat rejection, and condenser inlet temperature. The present paper focuses on the reduced order microclimate formulation and its plant level interpretation rather than on a specific controller.

4 Reduced Order Microclimate Model

The example represented as a regular 4 by 5 array of air cooled chillers. Each chiller can act as a source of hot exhaust and as a target receiving recirculated air from other sources. A representative geometry was used for the paper example based on an air cooled chiller with 300 Ton capacity. The formulation combines four effects:

- source temperature rise from heat rejection and condenser air flow,
- buoyancy influence that tends to lift warmer plumes,
- wind driven transport and spatial decay, and
- superposition of source contributions across the array.

For source chiller (i,j), the condenser air temperature rise associated with heat rejection is estimated from a sensible energy balance:

$$\Delta T_{src,i,j} = \frac{Q_{con,i,j}}{\dot{m}_{air,i,j} c_p}$$

where Q_{con} is condenser heat rejection, m_air is condenser air mass flow rate, and c_p is the specific heat capacity of air. A buoyant velocity equation is then written as:

$$V_{bou,i,j} = \sqrt{g\,\beta\,\Delta T_{src,i,j} L_{char}}$$

β is approximated as $1/T_{amb}$ for ideal gas behavior. L_{char} is a characteristic chiller length. This term is not intended to resolve a detailed fan jet; it provides a compact measure of the tendency for warm exhaust to rise relative to wind advection.

For wind active conditions, the source target vector is decomposed into downwind and crosswind distances relative to the wind direction. Contributions are applied only to target chillers that are downwind of a source. The baseline source target influence is,

$$\alpha_{base} = \alpha_0 exp\left(-\frac{k_{dow} d_{dow}}{L_{char}}\right) exp\left(-\frac{k_{cro} d_{cro}}{L_{char}}\right)$$

α_0 is a baseline recirculation coefficient. k_{dow} and k_{cro} control downwind and crosswind decay. Under calm wind

DOI 10.3384/ECP219 234

conditions, the model uses isotropic distance decay rather than wind aligned decay.

The final source target coefficient is reduced when buoyancy dominates over wind:

$$\alpha = \alpha_{base}[1 - tanh(k_{bou}\Pi)]$$

$$\Pi = \frac{V_{bou} + V_{conFan}}{V_{wind}}$$

This expression makes the model more recirculation prone when wind advection is strong relative to buoyant rise, and less recirculation prone when the warm plume is expected to lift away from nearby condenser inlets.

The inlet temperature of each target chiller is computed by adding the ambient temperature to the sum of source contributions from other operating chillers:

$$T_{in,t} = T_{amb} + \sum_{s \neq t} \alpha_{s,t}(T_{out,s} - T_{amb})$$

The superposition approach is simple enough for repeated Modelica simulation while retaining the physical dependencies needed for plant level analysis.

Modelica implementation

The source temperature rise is calculated only for operating chillers with nonzero air flow and heat rejection:

```
for i in 1:nRow loop
  for j in 1:nCol loop
    if mConAir_flow[i, j] > 1E-6 and QCon_flow[i, j] > 0.0 then
      dTSou[i, j] := QCon_flow[i, j] / (mConAir_flow[i, j] * cpAir);
    else
      dTSou[i, j] := 0.0;
    end if;
```

The implementation then applies wind aligned or calm condition decay depending on the wind speed:

```
if haveWind then
  dDow := -dDisRow * cosWin + dDisCol * sinWin;
  if dDow > 1E-6 then
    dCro := abs(dDisRow * sinWin + dDisCol * cosWin);
    facBas := alpha0 * exp(-kDow * dDow / lenCha) * exp(-kCro * dCro / lenCha);
  end if;
else
  dEff := max(sqrt(dDisRow * dDisRow + dDisCol * dDisCol), lenCha);
  facBas := alpha0 * exp(-kCal * dEff / lenCha);
end if;
```

These code fragments illustrate the intended Modelica use: the block remains compact, signal based, and fast enough to be evaluated repeatedly as part of a larger plant model.

5 CFD Model Description, Calibration and Validation

The present study considers CFD simulations of a chiller farm with 20 units arranged in a 4×5 array (Figure 4) is used to generate data for parameter estimation. In the CFD simulations two principal CFD boundary condition variables are varied to generate data: incoming wind speed, v, and wind direction, α.

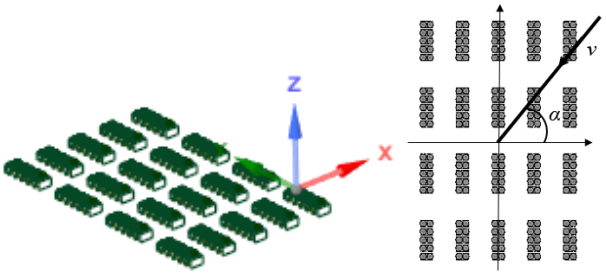


Figure 4. 3D Model of Chiller Farm and Definition of Wind Direction

A total of 65 CFD cases were completed over the selected combinations of wind speed and wind direction. Each simulation requires approximately 12 hours using 200 CPU cores.

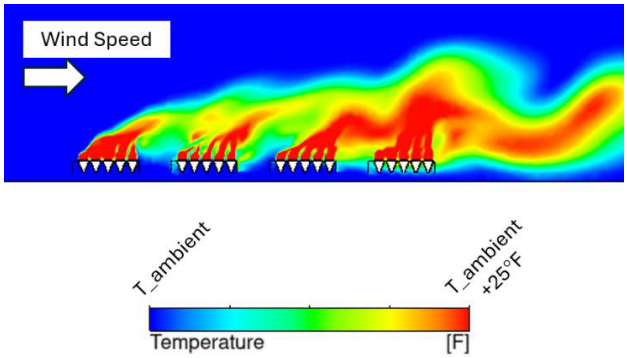


Figure 5. Temperature Distribution on Vertical Plane

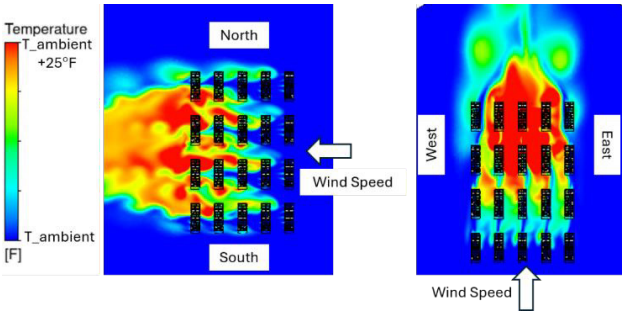


Figure 6. Temperature Distribution on Horizontal Plane

Figures 5 and 6 illustrate representative temperature fields from the CFD simulations. Figure 5 shows the temperature distribution on a vertical plane through the chiller array. Warm condenser exhaust from upstream chillers is transported toward downstream units, resulting in elevated local condenser inlet temperatures. Figure 6 shows the temperature distribution on a horizontal plane at the approximate condenser coil mid height. The orientation and spatial extent of the elevated temperature region vary strongly with wind direction.

DOI 10.3384/ECP219 234

The CFD results provide condenser inlet temperatures at individual chiller locations for calibration and validation of the reduced order model. The reduced order model contains empirical parameters because it cannot explicitly recover all near field plume physics, fan discharge details, and site specific mixing processes. The calibrated model is then validated against additional CFD cases that were not used in the fit. This workflow treats CFD as the higher fidelity design and calibration tool, while the reduced order Modelica block is used for repeated system simulation. The calibration data are not intended to make the Modelica block a standalone substitute for CFD; rather, they anchor a compact plume formulation so that it can reproduce the leading recirculation sensitivities at plant simulation speed.

Six tuning parameters were calibrated using least squares optimization. They control the buoyancy transition, baseline recirculation intensity, downwind decay, crosswind decay, calm wind decay, and the threshold between calm and wind active behavior.

Table 1. Range of tuning values and optimized amount

	Optimized Value	Low Range	High Range
k_{tanh}	0.22	0	10
α_0	0.14	0	2
$k_{downwind_alpha}$	0.43	0	4
$k_{crosswind_alpha}$	0.7	0	4
$k_{calm_decay_alpha}$	1.8	0	4
$V_{wind_threshold}$	0.87	0	2

The current calibration improves the reduced order response relative to the untuned model, but the validation error remains higher than desired for final quantitative prediction. However, it captures control relevant sensitivities and spatial trends, even if some chiller locations and operating cases still show significant error relative to CFD. This is consistent with the modeling intent: the block is intended first to bring recirculation sensitivity into fast plant simulation, while expanded CFD coverage is still needed before strong location by location temperature accuracy claims are appropriate.

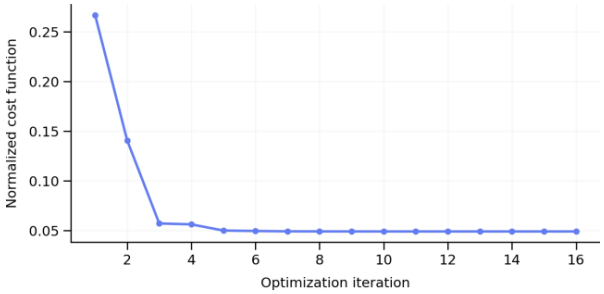


Figure 7. Calibration cost function reduction trend

Figure 7 shows reduction in normalized calibration cost during parameter estimation. The decrease indicates that the CFD data improve the fitted plume coefficients, but it should not be interpreted as full validation.

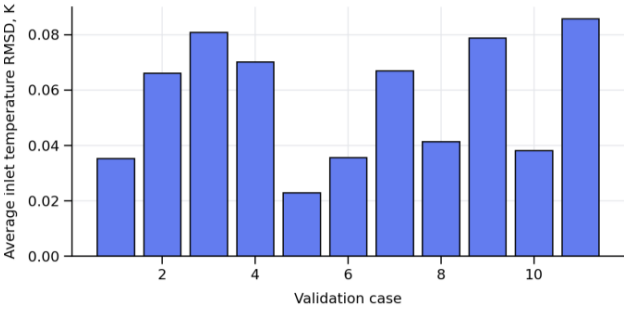


Figure 8. Case averaged inlet temperature error

Figure 8 shows case averaged inlet temperature RMSD across eleven validation cases. The validation set is separate from the calibration fit and is used to assess generalization across operating conditions.

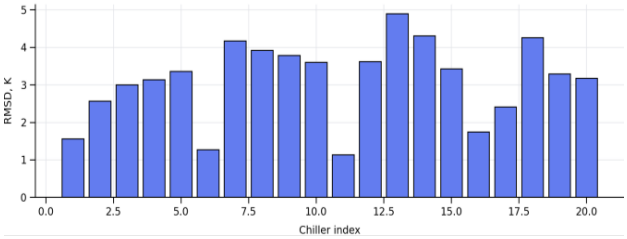


Figure 9. Chiller level RMSD across validation cases

Figure 9 shows RMSD of predicted inlet temperature by chiller index. The variation by position shows that the present calibration captures some locations better than others.

Additional CFD simulation results are expected in the next month before submission. Expanding the CFD data set across more wind directions, wind speeds, ambient conditions, and loading patterns should improve coefficient estimation and reduce spatial validation error. Until that broader calibration is complete, the results are best interpreted as a fast screening, diagnostic, and controls-development capability rather than as a final design prediction method.

6 Results and Discussion

The purpose of the coupled simulation is to expose microclimate effects that are invisible to plant models using a single ambient air temperature. The chiller farm model supplies heat rejection and condenser air flow for each unit at each time step. The microclimate block then converts those source terms and weather inputs into local condenser entering air temperatures. The resulting inlet temperature field feeds back into chiller performance,

DOI 10.3384/ECP219 234

allowing recirculation effects to appear in coefficient of performance, capacity, and plant level energy analysis.

The results in this section should be read in two layers. First, the dynamic simulation illustrates the kind of coupled behavior that becomes available when local condenser inlet temperatures are computed rather than prescribed as a common ambient value. Second, the validation plots show the present state of model maturity. The framework is already useful for identifying control relevant trends, but the current calibration does not yet support final quantitative design claims at every chiller location.

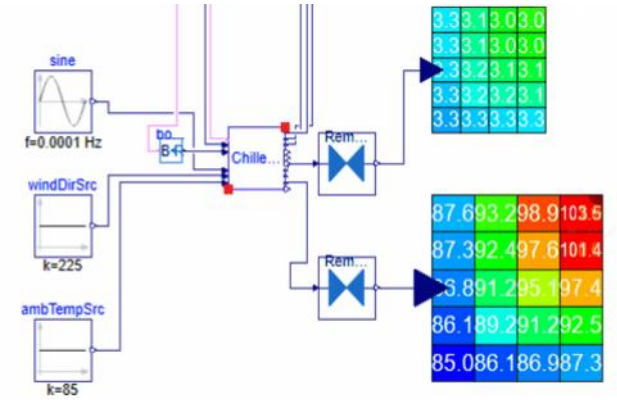


Figure 10. Modelica simulation results, condenser inlet temperature and Chiller COP responses.

Figure 10 shows an illustrative Modelica simulation in which wind speed changes over time and the condenser inlet temperature and chiller coefficient of performance respond. The wind speed reaches approximately 16 m/s during the high wind portion of the scenario. At the same time, the hottest downstream condenser inlet reaches about 102.8° F while ambient air remains at 85.0° F. This is a local inlet rise of about 17.8° F, or approximately 9.9° C, above ambient.

This result matters because a condenser inlet penalty of nearly 10°C is large enough to change equipment behavior in a plant simulation. Higher entering air temperature increases compressor lift, can reduce available capacity, and reduces efficiency. If a plant model used only the 85°F ambient boundary condition, it would miss the local condition that the downstream chiller experiences in the recirculation prone portion of the yard. The error is not merely a reporting issue: it changes the simulated operating point of the chiller receiving recirculated air.

The high wind condition is especially informative. Wind is sometimes assumed to improve heat rejection by increasing outdoor air mixing. At the array scale, however, wind can also bend vertical exhaust plumes into downstream condenser inlets. The net effect depends on direction, speed, spacing, heat rejection, and fan flow. A reduced order microclimate block is useful because it allows these dependencies to be evaluated repeatedly in the plant model instead of requiring a new CFD simulation for each operating point.

The same scenario shows that different chillers can operate at meaningfully different efficiency under the same ambient weather condition. A chiller receiving relatively fresh upstream air shows a COP of about 3.4, while the chiller exposed to the hottest recirculated air shows a COP of about 3.0. The difference corresponds to an illustrative COP reduction of roughly 11.8% between favorable and unfavorable microclimate positions.

This spatial variation is operationally important. In a conventional plant simulation with one outdoor air temperature, two identical chillers at the same load may appear to have the same available capacity and efficiency. In a real case, wind aligned recirculation can make one chiller a more expensive source of cooling than another. That difference can affect staging, load distribution, fan operation, and the interpretation of plant performance metrics such as PUE. The benefit of the Modelica integrated microclimate model is that these effects become available to the same simulation environment used for plant sequence studies.

The result also reframes how chiller sequencing should be evaluated. A sequence that is optimal under a uniform ambient assumption may be less favorable when wind direction places one subset of chillers in the exhaust path of another subset. Conversely, a sequence that looks equivalent in a conventional model may reduce recirculation exposure by shifting load toward units receiving fresher air. The present model is not yet mature enough to prescribe final operating rules on its own, but it can make these hypotheses testable in a dynamic plant simulation.

The dynamic scenario demonstrates the type of behavior the coupled framework can represent, but it should not be overinterpreted as a final accuracy claim. The validation plots show that the current CFD calibrated model still has higher error than desired, particularly at some array locations. This is not surprising for a compact Gaussian plume-based formulation applied to near field equipment recirculation, where local geometry and fan interactions can matter.

The current model is best interpreted as a physically informed, fast running engineering model. It is already useful for comparing operating strategies, identifying likely recirculation sensitive positions, testing whether recirculation aware sequencing logic has value, and exploring long weather sequences that would be infeasible with CFD alone. Stronger quantitative claims should wait until the expanded CFD data set is incorporated and the validation error is reduced.

DOI **10.3384/ECP219 234**

The validation figures are therefore important not only as a performance report but also as a guide for near-term improvement. The position dependent RMSD indicates that a single set of plume coefficients does not yet represent all source target interactions equally well. Additional CFD cases should improve the conditioning of the coefficient estimation problem and make it possible to distinguish parameter effects that are currently correlated, such as downwind decay, crosswind decay, and buoyancy suppression.

Even with the present validation caveat, the framework has practical uses. It can screen layout or spacing sensitivity before detailed CFD is commissioned. It can evaluate candidate chiller sequencing and load distribution strategies under changing wind direction. It can support long horizon weather sensitivity studies in which CFD would be computationally prohibitive. It can also help plan instrumentation by identifying which chiller positions are most likely to provide useful condenser inlet temperature measurements.

The key point is that CFD and the reduced order model serve different roles. CFD remains the detailed design tool. The reduced order Modelica model provides a fast approximation that brings microclimate sensitivity into plant simulation and controls development.

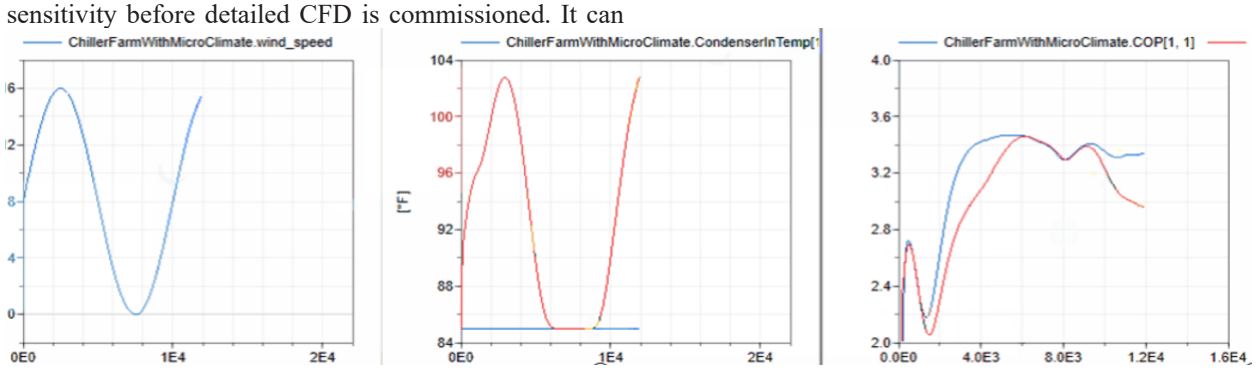


Figure 11. Modelica dynamic simulation showing wind speed, condenser inlet temperature, and COP response. The case illustrates how wind can move hot discharge air through the array and create local condenser inlet temperatures well above ambient.

7 Limitations and Future Work

The model remains a reduced order representation and should not be used as a direct replacement for detailed CFD in final chiller yard design validation. The present formulation evaluates plume interaction quasi steadily at each time step and represents missing near field physics through a small set of calibrated coefficients. It does not explicitly resolve screens, walls, nearby buildings, nonuniform approach flow, fan discharge details, or transient vortex structures.

Additional CFD results help providing broader coverage of wind direction, wind speed, ambient condition, and loading distribution. The expanded data set could improve coefficient estimation and reduce location dependent validation error. Future extensions may include explicit yard coordinates rather than only row and column spacing, irregular layouts, building and screen effects, field measurement validation, and integration with recirculation based supervisory control strategies.

8 Conclusions

This paper presented a reduced order Gaussian plume-based microclimate model implemented in Modelica and coupled with a data center chiller farm model. The chiller farm computes time varying heat rejection and condenser air flow for each chiller, while the microclimate block uses those source terms together with ambient temperature, wind speed, and wind direction to estimate local condenser inlet conditions across the array.

The work addresses an important gap between CFD and plant simulation. CFD remains the preferred tool for detailed design studies, but its runtime makes repeated weather, sequence, and controls studies impractical. The reduced order Modelica block is fast enough for plant level simulation and can reveal effects that a single ambient temperature assumption misses, including the illustrative high wind case in which the hottest condenser inlet rises about 17.8°F above ambient and the COP difference between favorable and unfavorable positions is about 11.8%.

The current model is appropriate for fast screening, diagnostics, sensitivity analysis, and controls development rather than final quantitative design prediction. Additional CFD cases improve calibration quality, reduce spatial validation error, and strengthen the model's usefulness for future recirculation-based plant simulation and supervisory control.

DOI **10.3384/ECP219 234**

References

1. AHRI, AHRI Standard 550/590 (IP) and 551/591 (SI): Performance Rating of Water Chilling and Heat Pump Water Heating Packages Using the Vapor Compression Cycle, 2023.
2. ASHRAE TC 9.9, Thermal Guidelines for Data Processing Environments, 5th ed., 2021.
3. B. R. Morton, G. I. Taylor, and J. S. Turner, "Turbulent gravitational convection from maintained and instantaneous sources," Proceedings of the Royal Society A, vol. 234, no. 1196, pp. 1-23, 1956.
4. G. A. Briggs, Plume Rise: A Critical Survey, U.S. Atomic Energy Commission, 1969.
5. D. B. Turner, Workbook of Atmospheric Dispersion Estimates: An Introduction to Dispersion Modeling, 2nd ed., CRC Press, 1994.
6. A. Venkatram and J. The, "Introduction to Gaussian plume models," in Air Quality Modeling, vol. I, 2003.
7. S. Brusca, F. Famoso, R. Lanzafame, S. Mauro, A. Marino Cugno Garrano, and P. Monforte, "Theoretical and experimental study of Gaussian plume model in small scale system," Energy Procedia, vol. 101, pp. 58-65, 2016.
8. Y. Joshi and P. Kumar, Introduction to Data Center Energy Flow and Thermal Management, Springer, 2012.
9. M. Wetter, W. Zuo, T. S. Nouidui, and X. Pang, "Modelica Buildings library," Journal of Building Performance Simulation, vol. 7, no. 4, pp. 253-270, 2014.

.

DOI **10.3384/ECP219 234**

Agent-Assisted Construction of Modelica System Models from Validated Components: A Data Center Cooling Case Study

Nithish Selvan[1] Johan Andreasson[2] Pratyush Agarwal[3]

[1]Modelon Engineering Pvt Ltd, India, nithish.selvan@modelon.com
[2]Modelon AB, Sweden, johan.andreasson@modelon.com
[3]Modelon Inc, United States, pratyush.agarwal@modelon.com

Abstract

We demonstrate that a validated Modelica component library, combined with an AI agent operating through a Model Context Protocol (MCP) server, can automatically construct and debug a simulation-ready system model from a vendor engineering specification. Starting from a validated 1 MW reference, the agent reads the Schneider Electric RD108 PDF, derives engineering parameters, writes 405 lines of Modelica for a 7.4 MW dual-loop data center cooling system, compiles, simulates, and identifies a 19 °C temperature error in the first run. It re-queries the library, selects the correct chiller record, and produces a validated model, all in a single working session. Access to a validated component library is what makes this practical: the agent selects and composes; the library guarantees the physics. We describe the MCP-based framework and discuss where the approach applies.

Keywords: Modelica, FMI, agentic AI, Model Context Protocol, system simulation, data center cooling, validated component libraries, AI-assisted modeling

1 Introduction

Starting from a vendor PDF and ending with a validated, simulation-ready Modelica model in a single working session: that is the capability this paper demonstrates. The model is a 7.4 MW dual-loop data center cooling system for NVIDIA GB200 AI racks, specified in the Schneider Electric RD108 reference design (Schneider Electric 2024). The complete sequence, from specification through model construction, simulation, diagnosis, and correction, runs without manual coding. The agent operates on Modelon Impact (Modelon AB 2026) through an MCP server, queries a validated DataCenter component library, writes Modelica source, compiles, runs a 24-hour dynamic simulation, finds a 19 °C temperature error in the LT loop on the first run, and corrects it by selecting the right chiller record from the library.

This works because of a validated Modelica library whose components carry 20 years of accumulated, customer-co-developed physics. The agent selects and composes from this library; it does not derive physics from scratch. The result is not a sketch or a generated document. It is a Modelica model that compiles on the platform, runs a dynamic simulation, and meets engineering KPIs against a published reference design.

Data center infrastructure is changing rapidly. Growth in AI compute demand is pushing facilities toward high-density liquid-cooled architectures, and direct-to-chip cooling is moving from research into production. Facility designers must evaluate a wider range of architectures under real time pressure. The path from a vendor reference design to a simulation-ready model is slow, and much of the work is systematic: reading specifications, deriving flow rates, selecting components, and wiring them together. This is where an agent is useful.

The remainder is organized as follows. Section 2 describes the agent framework. Section 3 introduces the DataCenter library. Section 4 presents the case study, including a first-build failure and its resolution. Section 5 discusses what the result shows and where the approach applies. Section 6 identifies future directions. Section 7 concludes.

2 The Agent Framework

The agent runs on Modelon Impact through a Model Context Protocol (MCP) server (Anthropic 2024; Hou et al. 2025). MCP is a recently standardized open protocol for connecting LLM agents to external tools and data sources. The server exposes platform capabilities as typed tools: query libraries and project models, read and write Modelica source, compile and run simulations, and extract scalar KPIs and time-series results. Each tool call returns a structured JSON response, and the full session is logged and auditable.

2.1 Workflow

The process is iterative, as shown in Figure 1. The agent reads the engineering specification and loads domain context, then queries the library for compatible component records and derives engineering parameters. This is the pre-build stage: assumptions are captured in a structured table, giving the engineer visibility of all extracted and derived values before model assembly. The agent then writes the model via `write_model`, triggers compilation, and launches a dynamic simulation. KPIs come back through `get_scalar_results`. On a pass, confirmed findings go to the knowledge base; on failure, the agent re-queries

DOI 10.3384/ECP219 242

the library and iterates. A representative subset of the tools is listed in Table 1.

2.2 Technical Foundations

The framework is built on open standards throughout. **Modelica and FMI** (Modelica Association 2023; Modelica Association 2022) provide equation-based physical models with compile-time correctness checks, covering unit consistency, conservation laws, and port compatibility, plus vendor-neutral binary exchange. **Python** is the automation and agent runtime layer. **JSON** carries structured data between tools; **YAML** is used where readability matters. **Markdown** captures domain knowledge in a format that domain experts can write and review without tooling. No single vendor controls any part of this stack. For facilities with asset lifecycles measured in decades, that is not an abstract concern: it is the difference between a workflow that stays usable and one that needs re-engineering each time a tool vendor changes its API.

Three guardrails are built into the framework and apply regardless of domain. A *parameter transparency step* presents a structured engineering summary (Table 3) to the engineer before model assembly; the workflow continues in a single session, but all extracted and assumed values are visible before any Modelica is written. *Verification tagging* requires every entry written to the knowledge base to carry a provenance tag (`observed`, `confirmed`, or `verified`). *Provenance enforcement* is implemented as an MCP tool that blocks any report containing a quantitative claim not backed by a citation to the experiment that produced it. All three are enforced through tool calls, so execution stops until the required conditions are met.

The case study in Section 4 was executed using Claude Code (Anthropic) with the `claude-sonnet-4-6` model as the agent runtime. The framework is LLM-provider-agnostic; the MCP tool interface and guardrails do not rely on model-specific behaviour.

3 The DataCenter Modelica Library

The agent's ability to select correct components, and to fix incorrect ones, depends on the library it draws from. The *Data Center Library (DCL)* is a commercial Modelica library developed at Modelon for system-level simulation of data center cooling infrastructure. At version 1.3, it builds on the Modelon LiquidCooling (v4.0), VaporCycle (v5.0), and Calibration (v1.4) libraries for validated thermodynamic component models. It covers air-cooled and liquid-cooled architectures at two fidelity levels: fast empirical representations for system-level studies and detailed physics-based components for transient analysis. For the agent, this translates to a selection space spanning seven component categories — chillers at two fidelity levels, cooling towers, CDU pods, CRAH units, pumps, fans, and rack types (air-cooled, hybrid, and liquid variants) — alongside 9 complete reference system examples the agent draws on for composition patterns.

3.1 Vendor CDU Records

The library includes over 15 CDU records parameterised from publicly available manufacturer specifications, spanning 800 kW to 2 MW across the ASHRAE TC 9.9 liquid-cooling thermal classes for AI and high-density deployments (ASHRAE Technical Committee 9.9 2021). Each record is paired with a calibration workflow for fitting against reference performance data; the agent selects which record to instantiate rather than generating component physics. A 1,840 kW record matching the RD108 HT loop specification is the example used in this case study. Records of this kind are not available in community libraries such as the Modelica Buildings Library (v13.0.0, 2025) (Wetter et al. 2014): although that library includes generic liquid-cooled data center models, it does not provide datasheet-parameterised CDU records for direct-to-chip AI rack cooling. The DataCenter library was developed to fill this gap, which is one reason this case study cannot be reproduced with a community-library-only toolchain.

3.2 Reference System Examples

The library provides a progression of validated reference system models: air-cooled systems, standard and economiser liquid-cooled chiller plants, and two complete facility models, `DataCenter1MW` validated against Schneider Electric RD48 (Schneider Electric 2023) and `DataCenter7MW`, the agent-built 7.4 MW dual-loop model described in Section 4. These reference systems carry the conventions the agent draws on: component patterns, diagram layout, naming, and port wiring. Rather than generating from first principles, the agent extends and re-composes validated reference architectures already in the library.

4 Case Study: Scaling a Validated 1 MW Reference Model to 7.4 MW

The task is to build a simulation-ready 7.4 MW dual-loop data center cooling model conforming to Schneider Electric Reference Design 108 (RD108) (Schneider Electric 2024), the Tier III AI reference design for NVIDIA GB200 deployments. The starting point is `DataCenter1MW`, a validated 1 MW model built to Reference Design 48 (RD48) (Schneider Electric 2023). The case study covers the complete cycle: validated baseline (§4.1), topology change (§4.2), parameter extraction (§4.3), model construction (§4.4), first simulation failure (§4.5), diagnosis and correction (§4.6), and final validated result (§4.7). For each step, the description covers what information was available, what the agent did, and what the outcome showed.

4.1 Starting Point: 1 MW Validated Baseline

The 1 MW reference model (`DataCenter1MW`, Figure 2) is a single-loop cooling architecture: 996 kW IT load (12 racks $\times$ 83 kW), one chiller with water-side economizer,

DOI 10.3384/ECP219 242

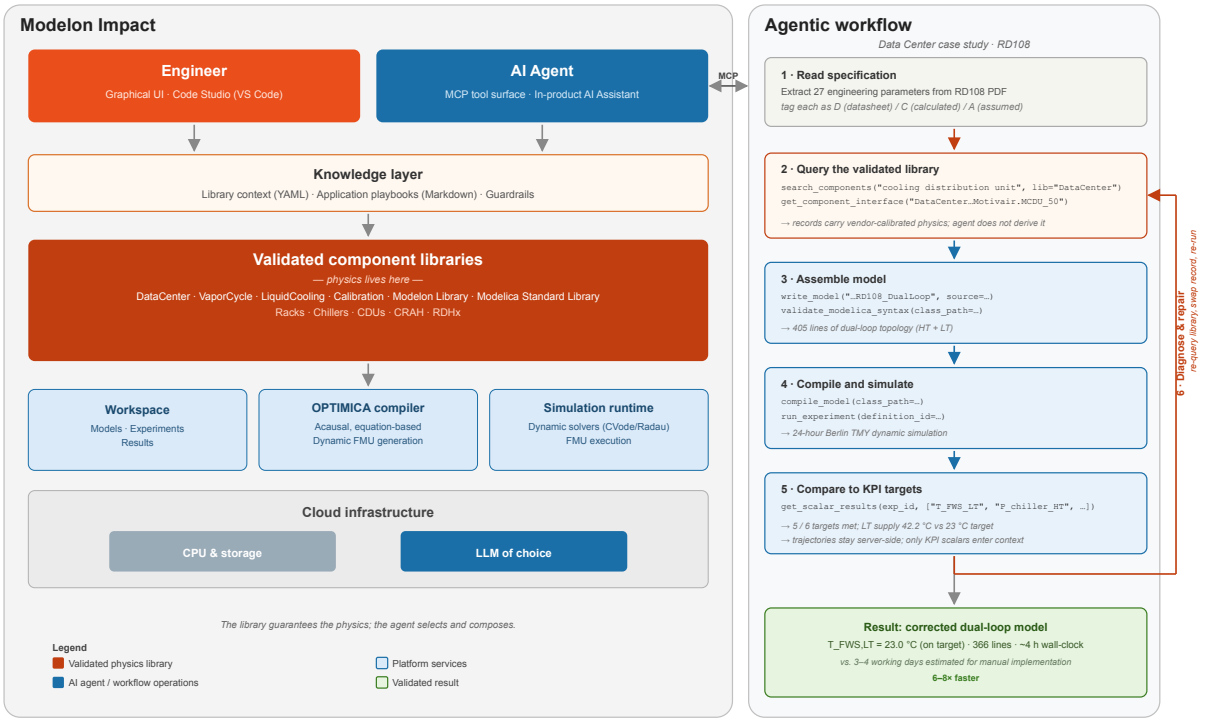


Figure 1. Agent framework (left) and workflow loop for the RD108 case study (right). The AI agent and engineer share the same Modelon Impact platform services; validated component libraries carry the physics. The iterative loop—read specification, query library, assemble model, compile, simulate, evaluate KPIs, diagnose and repair—runs entirely through typed MCP tool calls, with no manual coding at any step. The *LLM of choice* in the architecture is `claude-sonnet-4-6` (Anthropic) for the case study presented here.

Table 1. Representative MCP tools exposed by the framework. Each tool has a typed JSON signature; the agent invokes them through the standard MCP protocol.

Tool	*Purpose / Returns*
`get_workspace_libraries`	Enumerate the Modelica libraries loaded in the workspace, with versions
`list_project_models`	List Modelica model class names present in the project
`read_model`	Read the full Modelica source of a named model
`write_model`	Write or overwrite Modelica source for a named model
`compile_model`	Compile a model to FMU; return compilation log and status
`run_experiment`	Launch a dynamic simulation experiment; return experiment identifier
`get_scalar_results`	Extract named scalar KPIs from a completed experiment (JSON)
`get_case_results`	Extract time-series trajectories for named variables
`get_component_interface`	Query a library component's parameters, ports, and type information

one cooling tower, and facility water supply at 22 °C. KPIs are validated against RD48 targets (Table 2) under a 24-hour Berlin TMY dynamic simulation.

This model is not just a reference point. The validated conventions it encodes, covering component patterns, diagram layout, naming, and port wiring, give the agent a correct foundation to extend rather than a blank page to fill.

Table 2. Validated KPIs for the 1 MW reference model against Schneider Electric RD48 targets, 24-hour Berlin TMY dynamic simulation. All primary targets pass; PUE is within 2.5 % of the RD48 design point.

KPI	*RD48 Target*	*Result*	*Status*
PUE	1.12	1.148	Close
P_{IT} total	996 kW	996 kW	Pass
T_{FWS} supply	22 °C	22 °C	Pass
T_{TCS} supply	26 °C	26.8 °C	Pass
$\dot{Q}_{\mathrm{CDU}}$	806 kW	790 kW	Pass

DOI 10.3384/ECP219 242

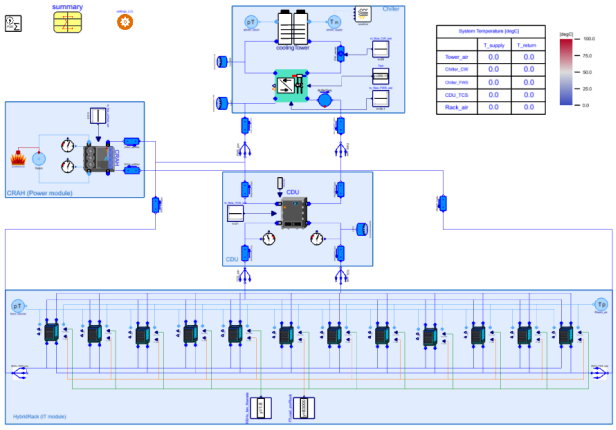


Figure 2. Validated 1 MW reference model `DataCenter1MW` in Modelon Impact. Single-loop topology: one chiller with water-side economizer, one cooling tower, CDU pods, and CRAH air handlers.

4.2 The Challenge: 7.4 MW Dual-Loop AI Facility

RD108 is not a scaled-up version of RD48. It is a fundamentally different topology (Figure 3):

- 7,392 kW total IT load: 48 AI racks × 132 kW (liquid-cooled via CDU pods) plus 48 networking racks × 22 kW (air-cooled)
- *Dual-loop cooling*: a high-temperature (HT) loop at 37 °C supplying AI rack CDUs, and a low-temperature (LT) loop at 23 °C supplying networking racks
- Tier III concurrently-maintainable architecture (Uptime Institute classification)

The supply temperatures map to ASHRAE TC 9.9 liquid-cooling thermal classes (ASHRAE Technical Committee 9.9 2021): the HT loop at class W40 (direct-to-chip cooling for AI accelerators) and the LT loop at class W27 (networking equipment). PUE targets align with ASHRAE Standard 90.4 (ASHRAE 2022b).

A topology change is harder than a capacity increase. The agent cannot simply scale the 1 MW model: it must construct new loop boundaries, select independent chiller records for two separate circuits, and derive flow rates and valve sizes for each loop. The input is the Schneider Electric RD108 PDF specification. No simulation-ready data is provided; all engineering values must be derived from the specification and the validated 1 MW reference model.

4.3 Pre-Build Data Extraction

Before model assembly, the agent reads the 10-page RD108 specification, extracts relevant parameters, and computes derived values (Table 3). The specification is extracted from PDF to plain text using `pypdf`; tables of rack capacities, supply temperatures, and flow parameters are parsed directly from the text layer and passed to the agent as structured Markdown. The structured output is then presented as a Markdown document, giving the engineer visibility of all extracted values as the session proceeds. Some values are read directly from the datasheet (D); others are calculated from first principles (C) or assumed where the specification is silent (A).

Resolving a reference design to validated library components, and computing all derived parameters needed to instantiate them, is the agent's core task at this stage. The human-review gate makes every assumption (source A) visible before model assembly begins. If an assumption is wrong, the engineer corrects it here, not after a failed simulation.

4.4 Model Assembly via MCP

The agent assembles the model entirely through MCP tool calls (Figure 4). Starting from the pre-build summary, it queries the workspace libraries, lists existing project models, reads the 1 MW reference to extract component conventions, and writes 405 lines of Modelica composing the 7.4 MW dual-loop topology via `write_model`. Compilation runs on the platform; a 24-hour Berlin TMY dynamic experiment launches immediately. No manual coding or copy-paste steps are involved. The full path from parameter table to first compiled model takes one session.

4.5 First Simulation: Most KPIs Pass, One Fails

Results from the first build are shown in Table 4. Five of six KPIs pass or are close to target. The LT loop supply temperature reads 42.2 °C against a 23 °C target, a 19 °C shortfall. The failure is specific, has a clear thermal signature, and gives the agent something concrete to work from.

4.6 Diagnosis and Resolution

Reading $T_{\mathrm{FWS,LT}}$ = 42 °C from `get_scalar_results`, the agent traces the LT loop thermal balance. Networking rack load is 1,056 kW; adding chiller-side heat rejection brings the required chiller capacity to approximately 1.88 MW. The LT chiller record in the first build is rated 1,125 kW, short by a factor of roughly 1.7. The 1,881 kW record had been identified as the correct match during the pre-build library query (Table 3) but was not used when the model was written. This is a record-selection error, not a physics error: the thermal equations are correct; the wrong record was instantiated.

The agent calls `get_component_interface` to confirm the 1,881 kW record (Figure 5), replaces the chiller instantiation, removes redundant components identified during the diagnosis, and re-runs the 24-hour Berlin TMY simulation. The Modelica source reduces from 405 to 366 lines in the process.

DOI 10.3384/ECP219 242

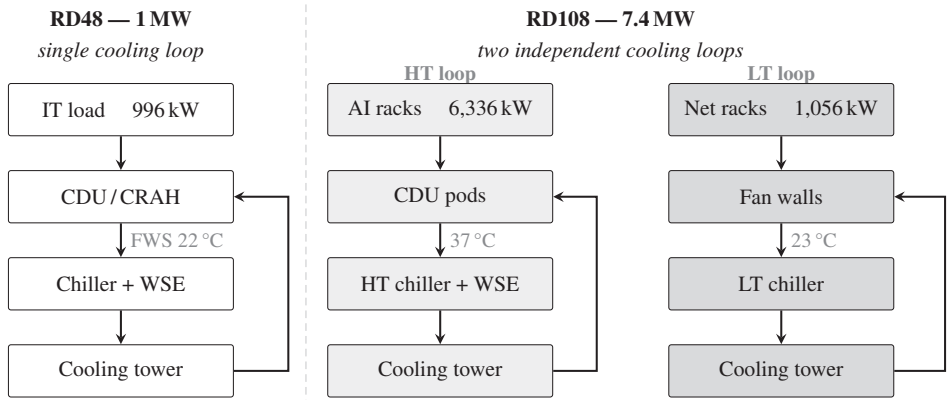


Figure 3. Topology comparison: RD48 (left) versus RD108 (right). The 1 MW RD48 design routes all IT load through a single cooling circuit at 22 °C. RD108 requires two independent loops operating at different supply temperatures: an HT loop at 37 °C for AI rack CDU pods and an LT loop at 23 °C for networking rack fan walls. The topology change is not a scaling problem. The agent must construct new loop boundaries, select independent chiller records, and derive separate flow rates and valve sizes for each circuit.

Table 3. RD108 design parameters extracted from the Schneider Electric specification (Schneider Electric 2024) and used to configure the 7.4 MW model. Source key: D = Datasheet, C = Calculated, A = Assumed.

Parameter	*Value*	*Unit*	*Src*
System			
PUE target	1.10	—	D
Total IT load	7,392	kW	D
Cooling budget (PUE − 1)	740	kW	C
TCS medium	Ethylene glycol 25 %		A
FWS medium	Propylene glycol 30 %		A
AI racks — HT loop (48 racks)			
Capacity per rack	132	kW/rack	D
Liquid fraction	87	%	D
TCS supply / return	40 / 50	°C	D
Cold air to rack	28	°C	D
Air mass flow	1.0	kg/s per rack	C
Liquid mass flow	3.0	kg/s per rack	C
Total AI rack capacity	6,336	kW	D
Networking racks — LT loop (48 racks)			
Capacity per rack	22	kW/rack	D
Air ΔT	12	K	A
Air mass flow	1.82	kg/s per rack	C
Total NET rack capacity	1,056	kW	D
CDU — HT loop (6 units)			
FWS supply / return	37 / 47	°C	D
Capacity per CDU	920	kW	C
FWS / TCS mass flow	24.2	kg/s per CDU	C
Fan wall — LT loop (6 units)			
FWS supply / return	23 / 33	°C	D
Capacity per fan wall	312	kW	C
FWS mass flow	8.21	kg/s	C
Chiller selection and valve sizing			
HT chiller capacity	5,520	kW	C
HT FWS mass flow	145.3	kg/s	C
LT chiller capacity	1,872	kW	C
LT FWS mass flow	49.3	kg/s	C
K_v, HT condenser valve	224	—	C
K_v, LT condenser valve	42	—	C
Chiller record, HT	5,549 kW library record		C
Chiller record, LT	1,881 kW library record		C

DOI **10.3384/ECP219 242**

Figure 4. Agent console output during model assembly. `compile_model` returns success and the checklist shows the full sequence: CDU record query, model write, compile, and simulation launch, all without manual intervention. Every action is logged and attributable.

Table 4. First-build KPIs for `DataCenter7MW` against RD108 targets, 24-hour Berlin TMY dynamic simulation. $T_{\mathrm{FWS,LT}}$ fails by 19 °C due to an undersized LT chiller record; all other KPIs pass or are close to target.

KPI	*RD108 Target*	*Result*	*Status*
PUE	1.10 (annualized)	1.153	Close
P_{IT} total	7,392 kW	7,392 kW	Pass
AI racks (48 × 132 kW)	6,336 kW	6,336 kW	Pass
Networking racks (48 × 22 kW)	1,056 kW	1,056 kW	Pass
$T_{\mathrm{FWS,HT}}$ (CDU supply)	37 °C	41.1 °C	Close
$T_{\mathrm{FWS,LT}}$ (networking)	**23 °C**	**42.2 °C**	**Fail**
$\dot{Q}_{\mathrm{CDU}}$ (3 pods)	5,512 kW	≈5,500 kW	Pass

Figure 5. Agent console output during diagnosis and correction. After observing $T_{\mathrm{FWS,LT}}$ = 42 °C against a 23 °C target, the agent traces the 1.125 MW chiller as undersized for the 1.88 MW LT load (≈1.7× shortfall), re-queries the library with `get_component_interface`, and selects the 1,881 kW replacement record. The fix is a single record substitution, not a topology change.

4.7 Validated Result

After the correction the model topology is unchanged (Figure 6); only the LT chiller record was updated.

The corrected run delivers $T_{\mathrm{FWS,LT}}$ at 23.0 °C, exactly on target. Table 5 gives the final validated KPIs.

Specification to validated model in a single working session. The source reduced from 405 to 366 lines in the correction pass: the diagnosis identified redundant instantiations that were cleaned up alongside the record substitution, leaving a cleaner model rather than a patched one. For context on effort: a concurrent manual implementation of the same RD108 model by an experienced simulation engineer is estimated at three to four working days; the agent completed the full cycle, including diagnosis and correction, in approximately four hours across multiple iterations.

The complete build required a single user prompt: two `write_model` calls, two compilations, and two 24-hour simulations, with one agent-initiated correction loop and no manual coding steps.

4.8 Knowledge Accumulation

Confirmed findings from each session go back into the knowledge base. The chiller sizing rule for LT loops, which must cover rack load *plus* heat rejection, the dual-loop topology pattern, the K_v scaling relationship, and the confirmed CDU record for the RD108 HT loop are all

DOI 10.3384/ECP219 242

Table 5. Final validated KPIs for `DataCenter7MW` against RD108 targets, 24-hour Berlin TMY dynamic simulation. $T_{\mathrm{FWS,LT}}$ passes at 23.0 °C after chiller record correction. PUE and $T_{\mathrm{FWS,HT}}$ deviations are explained by Berlin TMY climate versus the Paris-calibrated RD108 reference.

KPI	*RD108 Target*	*Result*	*Delta*	*Status*
PUE	1.10 (annualized)	1.153	+0.053	Close[a]
P_{IT} total	7,392 kW	7,392 kW	0	Pass
AI racks (48 × 132 kW)	6,336 kW	6,336 kW	0	Pass
Networking racks (48 × 22 kW)	1,056 kW	1,056 kW	0	Pass
$T_{\mathrm{FWS,HT}}$ (CDU supply)	37 °C	41.1 °C	+4.1 °C	Close[b]
$T_{\mathrm{FWS,LT}}$ (networking)	**23 °C**	**23.0 °C**	**0**	**Pass**
$\dot{Q}_{\mathrm{CDU}}$ (3 pods)	5,512 kW	≈5,500 kW	−12 kW	Pass

[a] PUE gap attributed to: (1) single Berlin winter day vs. RD108 annualized Paris; (2) dry cooling towers in model vs. evaporative in RD108; (3) HT chiller in partial mechanical-cooling mode on simulation day 1. An 8,760-hour annual sweep is required for investment-grade PUE comparison.

[b] $T_{\mathrm{FWS,HT}}$ offset is consistent with Berlin TMY climate: the water-side economizer dominates for most of the simulation day, keeping HT chiller power at 2.8 kW while WSE contributes 5,494 kW. This is correct physics for a cold climate; the RD108 targets are calibrated for a warmer (Paris) reference climate.

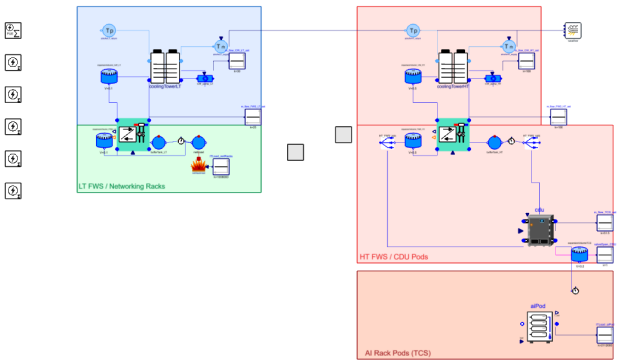


Figure 6. Agent-generated `DataCenter7MW` model in Modelon Impact. Dual-loop architecture: HT loop at 37 °C serving 48 AI rack CDU pods, LT loop at 23 °C serving 48 networking racks. The model topology is identical between the first and corrected runs; the fix was a chiller record parameter change, not a structural edit.

tagged `confirmed` in the workspace knowledge. The next session working on a similar facility starts with these results already in place.

4.9 Where Human Judgment Remains Essential

The agent produced a validated draft; it did not replace engineering judgment. Human review was needed at several points: deciding whether the dual-loop topology was the right choice for the customer's use case; interpreting the PUE gap given Berlin TMY versus the Paris-calibrated RD108 reference; deciding whether the $T_{\mathrm{FWS,HT}}$ deviation warranted a model correction or documentation; and signing off on Tier III staging decisions that require site-specific knowledge. The right split is to automate repetitive derivation and assembly, and to keep human judgment on decisions that require context the agent does not have.

5 Discussion

5.1 Why the Validated Library Matters

Agents work reliably when they select and compose validated components. Generating physics from first principles introduces uncertainty that is hard to bound and difficult to audit. The agent in this study never wrote a heat-exchanger equation or derived a chiller efficiency curve. Each library component carries its own verified physics; the agent relies on that rather than deriving it. Library correctness is established once, by domain experts with access to manufacturer data and calibration measurements. The agent takes on the derivation and assembly work: systematic, time-consuming, and requiring no engineering judgment.

The diagnosis episode shows why the library matters for correction as well as construction. The agent corrected a record-selection error by re-querying the library. Without the validated library, there would have been nothing to re-query. An approach that generates components from first principles leaves the customer with unverified physics, no calibration history, and no industrial reference cases. For systems with asset lifecycles measured in decades, that is a poor trade.

5.2 Reasoning versus Retrieval

Not all steps in the workflow are equivalent, and the distinction matters. Two contributions come primarily from *retrieval*: component naming conventions, diagram wiring patterns, and port connectivity style were read from the validated 1 MW reference model; component selection (CDU capacity, chiller record, rack type) was resolved by querying the library. These steps are reliable because the source material is validated, not because the agent reasoned carefully.

Three contributions required *reasoning*: constructing a dual-loop topology that does not exist in the reference model; deriving flow rates, valve K_v values, and heat rejec-

DOI 10.3384/ECP219 242

tion budgets for two independent circuits from first principles; and diagnosing the 19 °C LT error by tracing the thermal balance back to an undersized chiller record. These are not look-up tasks.

The split is intentional. Retrieval handles what the library already knows; reasoning handles what is new in the target design. Conflating the two overstates what the agent generates and understates what the library provides.

5.3 What Generalizes

The approach works here because a validated library exists for the domain, the reference design is documented clearly enough to extract engineering parameters, the KPIs are quantitative and machine-checkable, and the thermal physics fits the Modelica equation-based paradigm. These are real preconditions, not edge cases. Where they hold, the same toolchain applies.

5.4 Where the Approach Does Not Apply

The approach runs into trouble in three situations. If no validated library covers the target domain, the agent must generate components from scratch and the validation guarantee disappears. If the reference design is poorly specified, with missing flow rates, undefined media, or ambiguous capacity ratings, parameter uncertainty accumulates through the derivation step in ways the agent cannot resolve alone. If the physics is novel and not yet captured in any library, the bottleneck shifts back to domain-expert modelling; the agent can support that work but cannot lead it. The approach depends on validated libraries and well-specified systems. It is not a substitute for them.

The open-standards stack brings a practical benefit: the MCP layer, knowledge structure, and guardrails are not tied to any specific simulation tool or LLM provider. The agent can be pointed at a different platform or a different language model without changing the workflow. Validated commercial libraries, community libraries such as the Modelica Buildings Library (Wetter et al. 2014), and standards-aligned KPIs (ASHRAE 90.1 (ASHRAE 2022a), 90.4 (ASHRAE 2022b), Guideline 36 (ASHRAE 2021)) together cover a large portion of building and facility applications. The same MCP toolchain and guardrails are in use with thermal management, powertrain, HVAC, and vehicle dynamics libraries inside Modelon Impact. Library content changes between domains; the framework does not.

6 Future Work

Agent-generated models compile and simulate correctly, but the current toolchain does not yet cover everything a human engineer does in the Impact GUI. Diagram layout, including component placement, icon positioning, and connection-line annotations, must still be applied manually after model writing. Extending the MCP toolchain to handle these graphical aspects, along with finer modelling conventions such as media assignment and connector orientation, is the main near-term development target.

Two specific gaps came up during this study. The MCP compilation log is currently truncated, which limits how far the agent can trace structural equations during diagnosis. A full log surface and a `dump_equations` tool would make debugging significantly more effective. Library record naming also needs tighter documentation: the York YK family and the York YT family (which tops out at 3,133 kW) are easy to confuse, and a mis-selection of this kind is exactly what caused the first-build failure. Clarifying these conventions in the agent's library context reduces the chance of repeating it.

The longer-term aim is to generalise this intake workflow to a wider range of data center architectures, moving directly from vendor datasheets to validated models with less per-project configuration.

7 Conclusions

Starting from a vendor PDF, an agent constructed a 7.4 MW Modelica data center cooling model, ran a 24-hour dynamic simulation, diagnosed a 19 °C temperature error caused by an undersized chiller record, corrected it through a library re-query, and delivered a validated result, all within a single working session. This is a current capability, not a future one.

The combination of a validated component library and a structured MCP tool interface is what makes it work. Models compile on the platform, KPIs come back through the same interface, and failures drive library re-queries rather than manual debugging. The agent does not derive physics; the library has already done that. Building the workflow on open standards throughout, Modelica, FMI, Python, and MCP, keeps it portable and auditable as the tooling landscape evolves. For the Modelica community, the validated libraries built over decades are what make agent-assisted engineering trustworthy rather than speculative.

References

Anthropic (2024). *Model Context Protocol Specification*. URL: https://modelcontextprotocol.io/.

ASHRAE (2021). *ASHRAE Guideline 36: High-Performance Sequences of Operation for HVAC Systems*. Tech. rep. Atlanta, GA: American Society of Heating, Refrigerating and Air-Conditioning Engineers.

ASHRAE (2022a). *ASHRAE Standard 90.1: Energy Standard for Sites and Buildings Except Low-Rise Residential Buildings*. Tech. rep. Atlanta, GA: American Society of Heating, Refrigerating and Air-Conditioning Engineers.

ASHRAE (2022b). *ASHRAE Standard 90.4: Energy Standard for Data Centers*. Tech. rep. Atlanta, GA: American Society of Heating, Refrigerating and Air-Conditioning Engineers.

ASHRAE Technical Committee 9.9 (2021). *Thermal Guidelines for Data Processing Environments*. Tech. rep. Atlanta, GA: American Society of Heating, Refrigerating and Air-Conditioning Engineers.

Hou, Xinyi, Yanjie Zhao, Shenao Wang, and Haoyu Wang (2025). *Model Context Protocol (MCP): Landscape, Security*

DOI **10.3384/ECP219 242**

Threats, and Future Research Directions. arXiv: 2503.23278 [cs.CR]. URL: https://arxiv.org/abs/2503.23278.

Modelica Association (2022). *Functional Mock-up Interface for Model Exchange and Co-Simulation Version 3.0*. Tech. rep. Modelica Association. URL: https://fmi-standard.org.

Modelica Association (2023). *Modelica – A Unified Object-Oriented Language for Systems Modeling. Language Specification Version 3.6*. Tech. rep. Modelica Association. URL: https://specification.modelica.org/maint/3.6/MLS.html.

Modelon AB (2026). *Modelon Impact: cloud-based system simulation platform*. URL: https://modelon.com/modelon-impact/ (visited on 2026-05).

Schneider Electric (2023). *Modular Data Center AI Reference Design EU – Reference Design 48*. URL: https://download.schneider-electric.com/files?p_enDocType=Other+technical+guide&p_File_Name=Modular+Data+Center+AI+Reference+Design+EU+FINAL.pdf&p_Doc_Ref=RD48DSR0_EN (visited on 2026-05-27).

Schneider Electric (2024). *Reference Design 108: 7.4 MW Tier III AI Reference Design (NVIDIA GB200)*. URL: https://download.schneider-electric.com/files?p_enDocType=Other+technical+guide&p_File_Name=RD108DSR7-GB200.pdf&p_Doc_Ref=RD108DSR0 (visited on 2026-05-27).

Wetter, Michael, Wangda Zuo, Thierry S. Nouidui, and Xiufeng Pang (2014). "Modelica Buildings Library". In: *Journal of Building Performance Simulation* 7.4, pp. 253–270. DOI: 10.1080/19401493.2013.765506.

DOI **10.3384/ECP219 242**

A Unified Dynamic System Testing Framework

Scott Greenwood[1]

[1]Oak Ridge National Laboratory, Oak Ridge, Tennessee, USA, greenwoodms@ornl.gov

Abstract

Regression testing of time-dependent behavior recurs in every simulation ecosystem, yet each has rebuilt a similar harness tied to a single backend, a single metric, and a single reference baseline. This paper introduces the Dynamic Systems Testing Framework (DSTF), an open-source framework organized around six plug-in layers (Source, Discovery, Backend, Dataset, Metric, and MetricTree), four of them open to third-party additions through a registry. Acceptance criteria are expressed as a tree of metric leaves combined under roll-up operators, and baselines carry one of three roles that the framework enforces. Criteria are authored through a machine-readable contract, a schema and a JSON Patch that the framework validates and applies; an interactive HTML report is one client of that contract, and a script or an AI assistant is another. DSTF is exercised across five backends (Dymola, OpenModelica, FMPy, Julia/ModelingToolkit, and Python) on bundled demonstration libraries and the TRANSFORM library, with recorded data and live simulator output traversing the same scoring pipeline.

Keywords: regression testing, Modelica, FMI, Julia, validation

1 Introduction

Regression testing of time-dependent behavior recurs across dynamic system modeling workflows. Modelica library authors guard releases with tube comparison scripts (Wetter et al. 2014), and consumers of Functional Mock-up Units (FMUs) write per project FMPy drivers on top of the Functional Mock-up Interface (FMI) (Blochwitz et al. 2011; Dassault Systèmes and FMPy contributors 2024). Julia modelers extend `@test` with trajectory checks over ModelingToolkit (Ma et al. 2021), and experimentalists reshape recorded data into the same pass/fail pipelines before trusting a calibrated model. Each ecosystem has built its own harness, and each rebuild has widened the gap between them.

These harnesses share more structure than their interfaces expose: a source of tests, a backend to run them, a trajectory, a baseline, and a rule that decides "passed." In practice, the implementations are tied to one backend, one metric (e.g., a tolerance tube or normalized root mean square error [NRMSE]), and one reference. More complex grading criteria, dependent on several conditions holding at once, are not generally a feature. Extending the core infrastructure to other metrics and simulators is not supported without significant effort or a restructuring of the harness.

The Dynamic Systems Testing Framework (DSTF) addresses these gaps with six plug-in layers (Figure 1): Source, Discovery, Backend, Dataset, Metric, and MetricTree. Four of them (Discovery, Backend, Metric, and MetricTree) are extended through a registry. A registry is a runtime table of named implementations that user code and third-party packages add to, either in place or from a separately installed package, without editing the framework. A MetricTree combines several metric leaves under roll-up operators, so a single test can declare multiple criteria without hand-coded glue. Baselines carry one of three named roles, primary, companion, and `soft_check`. Acceptance criteria are authored through a machine-readable contract: a JSON Schema that the framework exports and an RFC 6902 JSON Patch (Bryan and Nottingham 2013) that a command-line interface (CLI) command validates and applies. The interactive HTML report is one client of that contract, and a script or an AI assistant is another. In this paper, DSTF is exercised across five registered backends (Dymola, OpenModelica, FMPy, Julia/ModelingToolkit (Ma et al. 2021; Bezanson et al. 2017), and a general Python backend) on the bundled demonstration/testing libraries and on the external TRANSFORM library (Greenwood 2020).

2 Background and Related Work

The closest prior art lives in the Modelica community's per library continuous integration (CI) setups, which share one shape: a driver simulates each model and compares its trajectory against a stored reference under one comparison rule. BuildingsPy (LBNL Simulation Research Group 2024) tests the Buildings library (Wetter et al. 2014) against stored Dymola results through pyfunnel, a tube comparator with axis-wise tolerances, and the TRANSFORM library (before DSTF) (Greenwood 2020) reused that driver while contributing an in-source convention: `UnitTest` components (ORNL TRANSFORM Project 2024) in which the user declares the variables

Notice: This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The US government retains and the publisher, by accepting the article for publication, acknowledges that the US government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for US government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (https://www.energy.gov/doe-public-access-plan).

DOI 10.3384/ECP219 251

to track, leaving the harness to autodiscover and execute each one. Regression testing of the Modelica Standard Library leans on csv-compare (Modelica Tools contributors 2024), which draws a tolerance tube around a stored curve and widens it in the x-direction at discontinuities. MoPyRegtest (Stelzig 2023) drives models through `.mos` scripting so that a suite runs on stock CI infrastructure, and OpenModelica's `OpenModelicaLibraryTesting` (OpenModelica 2024) orchestrates reference generation, translation, and simulation across a corpus of libraries, but only against the OpenModelica runtime. ReSim (Schaefer and Gall 2026), developed in the PHyMoS context, moves in the same multitool direction by exercising a library's suite across several Modelica tools. However, ReSim does not compose several acceptance criteria into one verdict or distinguish the role a baseline plays in the comparison, and its scope remains within the Modelica ecosystem rather than spanning the FMU, Julia, and recorded-data sources that this work treats uniformly. Each setup is expressive inside its host ecosystem and opaque outside it, so cross-tool comparison is a porting problem before it is a technical one.

FMU (Blochwitz et al. 2011) regression testing is mostly per project. FMPy (Dassault Systèmes and FMPy contributors 2024) simulates both model exchange and co-simulation FMUs from Python. Regression comparison against a stored reference is left to per project scripts. Vendor tools ship their own harnesses, such as Dymola's Testing Library (Dassault Systèmes 2025) and OpenModelica's regression testsuite, each tied to its host simulator's invocation conventions. OMSimulator (Ochel et al. 2019) addresses co-simulation orchestration under the System Structure and Parameterization (SSP) standard rather than regression testing of the resulting signals, and is complementary to this work rather than a predecessor.

In the Julia community, ModelingToolkit (MTK) (Ma et al. 2021) composes equation systems atop Julia (Bezanson et al. 2017). The ecosystem offers no trajectory-comparison harness beyond the standard `@test` machinery, so comparisons are hand-coded per project. JuliaHub's Dyad (JuliaHub 2025) is the closest Julia-side analogue to the per library Modelica harnesses. It is an acausal modeling language that lowers to MTK, with a `test` component keyword that autogenerates reference trajectories for `pkg> test` to check against, again through a tolerance against a single baseline.

Three shortcomings cut across the work above. First, acceptance criteria are not composable: pyfunnel offers one metric kind and the rest offer a scalar tolerance against a single reference, but a single test may need several checks at once, such as a faithful start-up transient, a steady-state envelope, correct event timing, and agreement with an independent backend. Second, the baseline is a single artifact. There is no way to declare that one reference is the regression anchor while another is visual context or an advisory cross-check, so comparisons that would otherwise be tracked are dropped rather than made load-bearing. Third, authoring happens in text files. Tube widths, weights, and thresholds are edited far from the plots that justify them, and every change requires a rerun. Apart from ReSim's multitool runs within Modelica, none of the surveyed tools spans backends, and each scores against a single reference under one rule. None offers a surface for authoring criteria in place; Section 6 is the response to that third shortcoming.

DSTF is complementary to the per library harnesses above, not necessarily a replacement. The Source layer registers a Recognizer for whichever in-source convention a library already uses, so existing Modelica, FMU, and Julia tests are picked up without rewriting. Its scope is held to regression. Calibration, parameter estimation, root-cause analysis, and test-failure ranking belong in downstream tools that consume its outputs, and parameterization, variant, and configuration management are likewise left to the surrounding toolchain. DSTF tests whatever configuration the host library hands it.

Standards work on simulation credibility supplies the process frame within which a regression harness is one instrument among several. The Credible Simulation Process (prostep ivip 2025a) prescribes a quality-assured, traceable workflow for producing and releasing simulation results, and the accompanying credibility assessment guidance (prostep ivip 2025b) sets out how that evidence is evaluated across company boundaries. However, neither document prescribes the mechanism by which a particular model's behavior is checked against a trusted result each time the model changes. DSTF supplies that mechanism for the regression slice of such a process, and the artifacts it stores are the kind of evidence a traceability requirement asks for: every baseline records the date it was added, named baselines carry a provenance block describing the run that produced them, and each report opens with a run-metadata banner. The relationship is therefore mutually supporting: the process standard states what evidence a credible release requires, and the framework produces that evidence on every run.

3 Architecture

Each of the six layers in Figure 1 has a defined interface. Recognizers, backends, metrics, and combinators install into a registry at import time. Source and Dataset are selected and declared rather than registered, for the reasons given below.

3.1 Source

A Source is the entry point. The user points DSTF at a folder, and the Source layer enumerates what is inside: the classes of a Modelica package tree, the folders of an FMU collection, the files of a recorded dataset. The Source does not decide which entries are tests; it inventories what is there and hands the list to Discovery. The layer carries no registry of its own, because a new kind of source arrives as a Recognizer for its contents plus a Backend where results must be produced rather than read.

DOI **10.3384/ECP219 251**

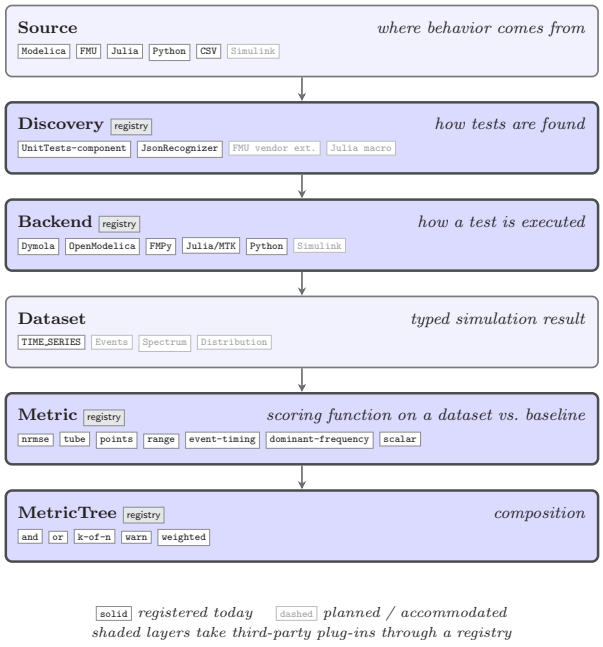


Figure 1. The six-layer plug-in architecture. Every layer has a defined interface, and new recognizers, simulators, metrics, or combinators install into the registry of their layer without disturbing the others.

3.2 Discovery and the Recognizer registry

Discovery decides which entries from the Source are tests and pulls from each the information DSTF needs: the model's name, how long to simulate, the solver tolerance, and the variables to track. Different libraries mark their tests differently, with a `UnitTests` component, an `experiment` annotation, or a naming convention, so Discovery uses a registry of Recognizers. A Recognizer is a small rule that says "a test looks like this" (e.g., extends a particular base class, instantiates a particular component, matches a class-name pattern) and "here is where to find each field." Libraries that already follow one of these conventions add a Recognizer rather than rewriting their tests.

3.3 Backend and capability declarations

The Backend runs a test. Given a test descriptor and a working directory, it simulates the model and returns the resulting trajectories. Rather than hard-coding what each backend can do, each one declares its capabilities, such as persistent worker support, a batch fallback, FMU export, and the Dataset kind it produces, and the orchestrator gates on those declarations rather than on class names. A backend is dispatched to a persistent worker only if it declares one. Persistent workers exist to amortize a simulator's start-up, library loading, and compilation cost across a suite instead of paying it once per test, which is the cost that dominates as a suite grows. The cross-backend chain of Section 7 asks for an FMU only from a backend declaring the export capability. Five backends are currently registered: Dymola, OpenModelica, FMPy, Julia/ModelingToolkit (Ma et al. 2021), and a Python backend that runs any module exposing a `simulate` function. The last need not wrap a simulator at all. A bundled example opens a recorded CSV and returns its columns, and the layers above treat the result exactly as they treat live simulator output, so recorded experimental data and simulations traverse the same scoring pipeline.

3.4 Dataset

A Dataset is whatever the Backend produces, in a form that a Metric can score. The kinds are a fixed enumeration rather than a registry: time series traces, plus declared kinds for scalars, events, spectra, and distributions. Each Metric declares the kinds it accepts, and a leaf handed a kind its metric does not accept fails with a diagnostic naming the mismatch rather than scoring something meaningless.

3.5 Metric

A Metric scores how close a simulation is to its baseline. Section 4 presents the criteria implemented today. Every metric registers one descriptor holding its configuration dataclass, its scoring function, the Dataset kinds it accepts, and the direction in which its score improves. From that descriptor the reporter's editor (Section 6) generates the metric's controls, the exported schema gains its fields, and the MetricTree can mix it freely with the others.

3.6 MetricTree

Tests are not scored by a single metric against a single reference. A MetricTree groups several leaves under interior nodes whose combinator decides how the children combine: `AND`, `OR`, `k-of-N` (k of N must pass), `weighted` (a weighted sum compared against a threshold), and `warn` – which scores its child and shows it in the report but never fails the test. Every leaf names the baseline it scores against; the semantics and the rules on which baselines a leaf may target are given in Section 4.

3.7 Registration pattern

Every registry above is reached the same way. The recognizers, backends, metrics, and combinators that ship with DSTF register exactly as a user-supplied one does: through a `register` call at import time, or from a separately installed package that declares an entry point, which DSTF loads before it dispatches any command. One entry-point group serves all four layers as a plug-in package may commonly span several of them. For example, a new ecosystem arrives as a Recognizer plus a backend and a backend producing a Dataset kind the built-in metrics do not accept brings its own metrics too. A plug-in that fails to load is a hard error, since a metric that quietly failed to register would silently change what a test means.

4 Acceptance Criteria as Composition

A test may need several criteria at once: an NRMSE bound on the start-up transient, a tube on the steady-state

DOI 10.3384/ECP219 251

window, and an advisory cross-check against a second simulator. DSTF expresses these as a MetricTree (Section 3), where each leaf is a metric scoring against a named baseline and each interior node combines its children's outcomes.

4.1 Leaf metrics

A leaf scores one tracked variable, or one number derived from it, against a named baseline and returns a verdict with a numeric score. Table 1 lists the current metrics. For example, NRMSE measures the average disagreement between a trajectory y and a reference y_{ref}, normalized by the reference's range:

$$\text{NRMSE}(y, y_{\text{ref}}) = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - y_{\text{ref},i})^2}}{y_{\text{ref,max}} - y_{\text{ref,min}}}. \qquad (1)$$

The summation in Eq. (1) is not taken over raw solver output. Every trajectory metric first interpolates the actual trajectory onto the reference time grid, so differing sample rates and variable-step solver output require no preprocessing on the part of the test author. Comparison then proceeds piecewise between the event boundaries that Modelica marks with duplicate time samples. At each boundary the pre-event or post-event value is selected, so that a discontinuity is never averaged across. A tube leaf instead checks whether the simulation stays inside a band drawn around the reference, whose width is a fixed offset, a relative percentage, or a time-varying profile declared as control points. It passes if and only if every reference-grid sample lies in-band, and the in-band fraction is its score (Figure 4). The band is drawn in the y-direction only, so a simulation that tracks the reference shape but arrives late leaves the tube for the duration of the shift. Time-axis tolerance is carried by other leaves rather than by the envelope. `points` applies a symmetric time-tolerance box around each checkpoint, and `event-timing` scores the shift of a discontinuity directly. New metrics join the table by registration (Section 3), and the tree treats them exactly as it treats these.

4.2 Combinators

A tree must reduce many leaf outcomes to one verdict, and the combinators differ in what they reduce. Most of them combine their children's verdicts, while `weighted` combines their scores and `warn` reports a child without letting it decide anything. An interior node T with children $T_1, \ldots, T_n$ passes when at least k of its children pass,

$$\text{pass}(T) \;\Leftrightarrow\; \sum_{i=1}^{n} [\text{pass}(T_i)] \;\geq\; k, \qquad (2)$$

where $[\cdot]$ is 1 for a passing child and 0 otherwise. `AND` is the case $k = n$, `OR` the case $k = 1$, and `k-of-N` names k explicitly. The `weighted` combinator instead collapses leaf scores s_k to a single aggregate S and compares it against a threshold θ:

$$S_{\text{weighted}} = \sum_{k=1}^{K} w_k s_k, \qquad \text{pass} \;\Leftrightarrow\; S_{\text{weighted}} \triangleleft \theta, \qquad (3)$$

where the comparison direction $\triangleleft$ is declared on the node itself rather than derived from what its children measure, defaulting to $<$ and set to $>$ where a higher score is better. A lower-is-better score such as NRMSE and a higher-is-better score such as a tube's in-band fraction carry no joint meaning under one threshold. The validator therefore rejects a `weighted` node whose leaf children mix score orientations, which makes the declared direction a checked property of the tree rather than an author's convention. The weights themselves remain the author's, since a normalized error and an in-band fraction live on different scales. The `warn` combinator wraps a single child that is still evaluated and reported but cannot fail the test. It is how an advisory check, whose disagreement should be visible but not load-bearing, is included in a tree. A `warn`-wrapped child under a `weighted` node contributes no numeric score and therefore fails the aggregate by design, so an advisory check belongs outside the weighted sum.

Table 1. Built-in leaf metrics. Each returns a verdict and a score; the last column says what the leaf needs from a baseline.

Metric	Checks	Needs
`nrmse`	normalized RMS error, Eq. (1)	trajectory
`tube`	every sample inside an envelope around the reference	trajectory
`points`	declared checkpoints within x- and y-tolerance	trajectory or none
`range`	signal stays within scalar bounds	none
`event-timing`	detected event instants match the reference's or declared ones	trajectory or none
`dominant-frequency`	fast Fourier transform (FFT) peaks match declared peaks	none
`scalar`	one number, a series reduction or a stored statistic, within tolerance or bounds	trajectory, statistic, or none

4.3 Baseline roles

Every leaf names which baseline it compares against, and baselines come in three roles (Figure 2). A *primary* baseline is the trusted reference the test must score against. The evaluator requires at least one leaf to target the primary outside a `warn` wrapper, so a test cannot accidentally pass while ignoring its anchor. A `soft_check` baseline is an advisory comparison, such as a run from a different simulator or a previous version of the model. Any leaf targeting it must sit under a `warn`, so disagreement is visible in the report but does not fail the test. A *companion* baseline is plot-only context (e.g., an experimental dataset).

DOI 10.3384/ECP219 251

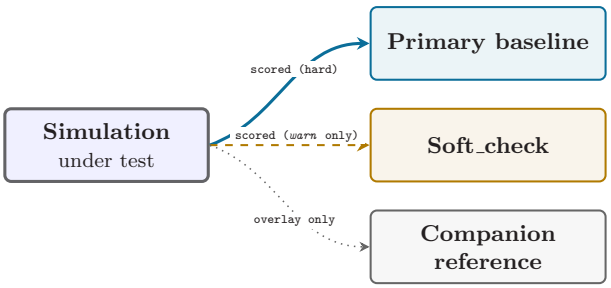


Figure 2. The three baseline roles. The solid arrow is the trusted reference (*primary*) that the test scores against, the dashed arrow is an advisory comparison (`soft_check`) reported but never failing the test, and the gray arrow is plot-only context (*companion*) overlaid for visual reference.

Leaves cannot score against it, but the reporter overlays it on the plot.

Baselines are not authored by hand but accepted from runs, either all at once or one test at a time, so a result can be inspected before it becomes the anchor that later runs are judged against. Trajectories produced elsewhere enter through the other two roles. Another regression system's primary is imported as a `soft_check`, and an experimental or cross-library dataset is registered as a companion.

4.4 Criteria beyond the trajectory

Not every regression that matters shows up in a trajectory: a model can produce the same curves while its translated form has grown a nonlinear system or doubled its state count. Each stored baseline therefore carries the translation statistics the backend reported (e.g., continuous states, scalar unknowns and equations, the sizes of the nonlinear and linear systems, and the event counter). Every run raises a structural warning wherever those numbers differ from the baseline's, which surfaces a change in the translated model even when the trajectories still agree. Warnings are advisory by construction, so the `scalar` leaf makes the same quantities load-bearing. It scores a single number against the baseline, within an absolute or relative tolerance or against declared bounds. That number is either a reduction of the leaf's tracked series, such as its maximum, mean, final value, or integral, or a stored statistic such as final CPU time. A CPU-time regression, a lost continuous state, or a nonlinear block that appeared between releases can therefore fail a test outright if desired.

4.5 A multi-criteria example

Figure 3 shows the tree for `MetricTreeTest`, a test in the bundled Modelica demonstration library. A root `AND` combines two hard-fail leaves against the primary Dymola baseline, an NRMSE bound on the start-up transient (the first 2.5 s) and a relative tube on the steady-state window after it, with a `warn`-wrapped leaf scoring the same trajectory against an OpenModelica `soft_check`. The JSON on the left is what the specification holds, and the tree on the right is what the report shows. The specification is

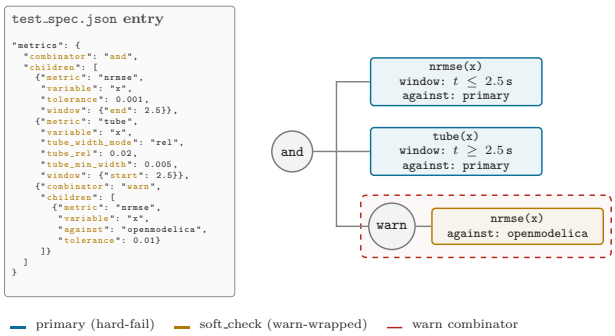


Figure 3. A multi-criteria test, `MetricTreeTest` from the bundled Modelica library. Two leaves guard the primary Dymola baseline under an `AND`, an NRMSE bound on the first 2.5 s and a relative tube afterwards; a `warn`-wrapped leaf against an OpenModelica `soft_check` reports cross-simulator disagreement without failing the test.

edited through any client of the contract described in Section 6.

5 Cross-Ecosystem Demonstration

DSTF ships four demonstration libraries, one per backend kind that double as DSTF test libraries. `ModelicaTestingLib` holds twelve tests covering every metric and every combinator, including the multi-baseline `MetricTreeTest` of Section 4 and a simulate-only path that skips comparison entirely. The same library runs against either Dymola or OpenModelica from one configuration, with DSTF picking whichever simulator is installed. The FMU demonstration pulls canonical FMUs (e.g., `BouncingBall`, `Dahlquist`, and `VanDerPol`) from the Modelica Association's Reference-FMUs repository (Modelica Association 2024). `JuliaMtkTestingLib` mirrors the Modelica library against ModelingToolkit (Ma et al. 2021) with eight tests over `nrmse`, `tube`, `range`, `points`, and `dominant-frequency` leaves and a composed tree. In each case the only ecosystem-specific additions are the Recognizer, which reads FMU metadata or `.jl` source rather than Modelica classes, and the backend; everything downstream of Discovery is identical.

The Python demonstration library shows that DSTF scores recorded data the same way that it scores a simulation. Alongside a conventional SciPy-based ordinary differential equation (ODE) example, `ConstantCsv` is not a simulator at all: its `simulate` function opens a CSV recorded on a test bench and returns the columns, with no model evaluation in between, and the MetricTree scores it against a stored reference exactly as it would a Modelica run. The recorded data path took a few dozen lines of Python and no changes to DSTF itself.

To verify DSTF's general applicability and scalability, the TRANSFORM Modelica library (Greenwood 2020), developed independently of DSTF, adopted it for unit testing. Its regression suite of more than three hundred tests

DOI 10.3384/ECP219 251

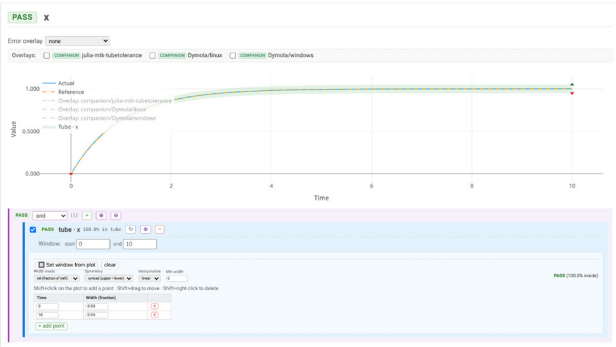


Figure 4. Interactive HTML report for `TubeToleranceTest` with the `tube` leaf's editor open: the control points that define the envelope are listed in the table and edited on the plot itself (Shift+click adds a point, Shift+drag moves one), and the leaf re-scores in the browser on every edit. The companion overlays offered above the plot are drawn but never scored.

is discovered from the library's `UnitTest` components and passes against stored baselines. Initial deployment helped find and resolve various minor implementation issues of DSTF as well as helped improve TRANSFORM test coverage and improved DSTF parallelization implementations on both Linux and Windows.

From a usability perspective, DSTF is built to run unattended as well as interactively. A run emits JUnit XML, which every mainstream CI service renders natively, and its exit status carries the verdict. Reruns can be executed in an incremental fashion where only the tests that failed previously are re-executed and their outcome is merged into the existing report, which keeps the loop short on a suite the size of TRANSFORM's.

6 Authoring Acceptance Criteria

Acceptance criteria are the specification of a test, and the third shortcoming of Section 2 is that this specification is authored in a text file. DSTF's answer is not a particular editor but a contract. The specification is plain JSON kept in the repository beside the library. The CLI exports the JSON Schema it is validated against, so the fields of a new leaf can be built from the schema alone and an invalid combination rejected before anything is saved. Every edit reaches the specification as an RFC 6902 JSON Patch (Bryan and Nottingham 2013), a small document naming the test and the operations applied to it, which a CLI command validates as a whole tree and only then writes, leaving every key the patch did not touch exactly as it was. Because the framework only ever sees the patch, it is agnostic to who wrote it.

The interactive HTML report (Figure 4) is the client built to show that the reporting and authoring loop can support an accessible human-in-the-loop process. These reports, when enabled, comprise an overall job report, a final report, and individual test reports. The individual reports enable per leaf controls generated from the same configuration definitions that the evaluator uses, so a control can never be out of sync with what the scorer accepts. The tree itself is editable in place, and for `tube`, `points`, `dominant-frequency`, and `range` the plot doubles as the editor, with tolerance bands, checkpoints, and FFT peak boxes dragged directly and re-scored in the browser as they move. When the author is satisfied, the report downloads the patch. This same process could however be accomplisehd through a purely script driven workflow or AI assistant working from the schema and the run's JSON outputs. Each emits the same patch and passes the same validation.

The patch is itself a small, reviewable artifact, so the multi-developer workflow reduces to ordinary version control. An edit made in a browser lands as an ordinary commit or pull request carrying the specification diff, with the patch attached as provenance for the change. Concurrent edits to different tests merge textually, and conflicting edits to one test surface as an ordinary conflict on a single JSON entry which can be managed by standard git best practices.

7 Discussion and Future Work

DSTF is a regression testing framework, and its scope is deliberately held there (Section 2). The contract of Section 6 is what makes tools built on its outputs easy to write. A tool that tracked how much a test's score drifts between runs could propose new tolerances without ever editing the specification. It would emit a patch like any other client, and validation would reject a bad proposal before it reached the repository.

Three areas stand out for future work. First, uncertainty quantification (UQ) enters through the Dataset layer, which already reserves kinds for data that is not a single trajectory: events, spectra, and distributions. Giving those the same standing as trajectories would also let the `event-timing` and `dominant-frequency` metrics share machinery with metrics added later. An ensemble run, one model swept over uncertain parameters, produces a sample per output instead of a single trajectory, and scoring it calls for metrics of a different shape: a quantile band doing for an ensemble what the tube does for one run, or a Kolmogorov–Smirnov statistic comparing a whole sample against a stored one. Each is a new plug-in rather than a change to the framework, and aligning them with the common UQ representation proposed by Bajand et al. (Bajand et al. 2025) would let a stored baseline carry the same uncertainty description that a Modelica, FMI, or SSP toolchain already produces. Sensitivity analysis reads those same records from the other side. Every leaf's score is stored with the trajectories it scored, which is the input a ranking tool needs. Second, the cross-backend chain of Section 3 exports a model as an FMU from one backend and replays it on another, so one backend's regression becomes a cross-check on another's. Making that automatic, so that every backend pair is compared without

DOI 10.3384/ECP219 251

an author asking for it, is the natural next step from the capability declarations. Third, tools that propose specification changes from evidence, plugged into each ecosystem's own test runner, are where DSTF meets the CI setups of Section 2.

8 Conclusion

DSTF is a unified regression testing framework for time-dependent system behavior, organized around a six-layer plug-in architecture. Whatever a backend produces is treated uniformly, so simulated trajectories from Modelica, FMI, Julia, or other tools and recorded experimental data flow through the same scoring pipeline. The MetricTree and the three-role baseline model handle the multi-criteria, multi-baseline cases that existing tools either ignore or hand-code. Acceptance criteria are authored through a machine-readable contract, a schema and a JSON Patch, of which the interactive report is one client and a script or an assistant another. That the same pipeline runs against a recorded CSV, three reference FMUs, a ModelingToolkit problem, and Modelica libraries on two simulators is the central claim: one abstraction in place of the per ecosystem harnesses that motivated the work. DSTF is open-source under an MIT license.[1] Backends, recognizers, and metrics are the three places most likely to attract third-party contribution.

9 Acknowledgments

AI Disclosure: This work used Claude Opus 5 to help brainstorm technical ideas, draft scripts, and organize the paper's structure. The author is fully responsible for the final content of the manuscript.

References

Bajand, Amin et al. (2025). "Towards a Common Standard for Uncertainty Quantification". In: *Proceedings of the 16th International Modelica & FMI Conference*. Lucerne, Switzerland: Linköping University Electronic Press. DOI: 10.3384/ecp218671.

Bezanson, Jeff, Alan Edelman, Stefan Karpinski, and Viral B. Shah (2017). "Julia: A Fresh Approach to Numerical Computing". In: *SIAM Review* 59.1, pp. 65–98. DOI: 10.1137/141000671.

Blochwitz, Torsten et al. (2011). "The Functional Mockup Interface for Tool Independent Exchange of Simulation Models". In: *Proceedings of the 8th International Modelica Conference*. Dresden, Germany: Linköping University Electronic Press, pp. 105–114. DOI: 10.3384/ecp11063105.

Bryan, Paul C. and Mark Nottingham (2013-04). *JavaScript Object Notation (JSON) Patch*. RFC 6902. DOI: 10.17487/RFC6902. URL: https://www.rfc-editor.org/info/rfc6902.

Dassault Systèmes (2025). *Automated Tests in Modelica with Dymola and the Testing Library*. Masterclass webinar recording, streamed live 18 September 2025 on the CATIA YouTube channel. URL: https://www.youtube.com/watch?v=wnk5YbEgTjM (visited on 2026-09-02).

Dassault Systèmes and FMPy contributors (2024). *FMPy: Simulate Functional Mockup Units (FMUs) in Python*. URL: https://github.com/CATIA-Systems/FMPy.

Greenwood, M. Scott (2020). "TRANSFORM – A Vision for Modern Advanced Reactor System-Level Modeling and Simulation Using Modelica". In: *Transactions of the American Nuclear Society*. Vol. 123. American Nuclear Society, Paper T123–32943. DOI: 10.13182/T123-32943.

JuliaHub (2025). *Dyad: An Acausal Modeling Language for Julia*. URL: https://juliahub.com/products/dyad.

LBNL Simulation Research Group (2024). *BuildingsPy: A Python package supporting the Modelica Buildings library*. URL: https://github.com/lbl-srg/BuildingsPy.

Ma, Yingbo, Shashi Gowda, Ranjan Anantharaman, Chris Laughman, Viral Shah, and Christopher Rackauckas (2021). "ModelingToolkit: A Composable Graph Transformation System for Equation-Based Modeling". In: *arXiv preprint arXiv:2103.05244*. DOI: 10.48550/arXiv.2103.05244.

Modelica Association (2024). *Reference FMUs: A collection of FMI reference models for testing and demonstration*. https://github.com/modelica/Reference-FMUs.

Modelica Tools contributors (2024). *csv-compare: Tool to Compare Curves from CSV Files Using an Adjustable Tolerance*. URL: https://github.com/modelica-tools/csv-compare.

Ochel, Lennart et al. (2019). "OMSimulator – Integrated FMI and TLM-based Co-simulation with Composite Model Editing and SSP". In: *Proceedings of the 13th International Modelica Conference*. Regensburg, Germany: Linköping University Electronic Press, pp. 69–78. DOI: 10.3384/ecp1915769.

OpenModelica (2024). *OpenModelica Library Testing*. URL: https://github.com/OpenModelica/OpenModelicaLibraryTesting.

ORNL TRANSFORM Project (2024). *TRANSFORM-UnitTests: In-source unit-test components for the TRANSFORM library*. URL: https://github.com/ORNL-Modelica/TRANSFORM-UnitTests.

prostep ivip (2025a). *prostep ivip SmartSE Recommendation 2025, PSI 11 V4.0 – Part H: Credible Simulation Process Framework with Detailed Process Description*. URL: https://www.prostep.org/en/medialibrary/publications/.

prostep ivip (2025b). *Smart Systems Engineering – SmartSE – Simulation Credibility Assessment*. White Paper, Version 1.0, December 2025. URL: https://www.prostep.org/en/medialibrary/publications/ (visited on 2026-09-02).

Schaefer, Matthias and Leo Gall (2026). *ReSim – Multitool Regression Testing for Library Development*. Proceedings of the 20th MODPROD Workshop, Linköping University. URL: https://wcc.ep.liu.se/index.php/MODPROD/article/view/1750.

Stelzig, Philipp Emanuel (2023). "MoPyRegtest: A Python Package for Continuous Integration-Friendly Regression Testing of Modelica Libraries". In: *Proceedings of the 15th International Modelica Conference*. Aachen, Germany: Linköping University Electronic Press, pp. 43–52. DOI: 10.3384/ecp20443.

Wetter, Michael, Wangda Zuo, Thierry S. Nouidui, and Xiufeng Pang (2014). "Modelica Buildings library". In: *Journal of Building Performance Simulation* 7.3, pp. 253–270. DOI: 10.1080/19401493.2013.765506.

[1] https://github.com/ORNL-Modelica/DynamicSystemsTestingFramework

DOI 10.3384/ECP219 251

Data-Driven Refinement of a Native Modelica Model Predictive Controller for Multi-Variable Gas Turbine Hydrogen Blending Control

Jim Harper[1] Paolo Pezzini[1]

[1]EPRI, USA, {jharper,ppezzini}@epri.com

Abstract

As gas turbine operators increasingly pursue hydrogen and hydrogen-blend fuels to decarbonize power generation, the need for advanced multi-variable control strategies that can safely manage the complex interactions between fuel composition, combustion dynamics, emissions, and power output has become critical. Hydrogen's lower volumetric energy density, wider flammability range, and faster flame speed fundamentally alter the combustion tuning landscape, requiring simultaneous coordination of fuel flow, inlet guide vane position, and fuel staging. These tasks may exceed the capability of traditional single-loop PID architectures. Model Predictive Control (MPC) offers a principled framework for this coordination, but its implementation within equation-based system modeling environments such as Modelica can be challenging per the authors experience, particularly for high-fidelity, physics-based gas turbine plant models where nonlinearity, stiff dynamics, and actuator constraints create significant practical challenges.

This paper extends the authors' prior work (**Harper 2026**), which demonstrated the feasibility of a fully native Modelica MPC architecture including Kalman filter state estimation, prediction horizon matrices, and a quadratic programming solver implemented entirely within the Modelica language for simultaneous control of hydrogen blend ratio, generator load, and turbine firing temperature. Where the prior paper established the architectural framework and initial proof of concept, the present work focuses on the systematic, data-driven refinement of both the internal plant model and the constrained optimization solver required to achieve robust closed-loop performance when coupled to a nonlinear, physics-based gas turbine simulation.

Key contributions include an updated open-loop system training data and identification procedure and the active-set quadratic programming solver was reformulated to employ explicit Lagrange multiplier computation through a full Karush-Kuhn-Tucker (KKT) system. Results from an iterative closed-loop simulation campaign spanning multiple configurations demonstrate progressive convergence from initial instability to successful simultaneous tracking of all three controlled variables across transient setpoint trajectories. The best configuration at $T_s = 0.01$ s achieves hydrogen blend ratio tracking within one percent steady-state error, generator load tracking within approximately half a megawatt, and turbine firing temperature offset of approximately six kelvins, with no fuel-control valve saturation during normal operation. A subsequent investigation re-identifies the discrete state-space model at $T_s = 1.0$ s which is two orders of magnitude closer to the dominant plant time constants, and replaces the online Kalman covariance propagation with a fixed-gain observer derived from the offline discrete algebraic Riccati equation (DARE). Closed-loop simulation at the longer sample period confirms stable multi-variable control with physically meaningful prediction-matrix gains, while highlighting that model mismatch across a wide operating envelope remains the primary barrier to tracking accuracy at this time scale.

Keywords: Model Predictive Control, Modelica, Quadratic Programming, KKT Saddle-Point System, Hessian Regularization, Active-Set Method, Gas Turbine, Hydrogen Blending, Kalman Filter, System Identification

DOI **10.3384/ECP219 258**

Nomenclature

Symbols

A	Discrete state-transition matrix $[s \times s]$
a	Discrete pole, $\exp(-T_{\text{mpc}}/\tau)$
B	Discrete input matrix $[s \times n]$
C	Output matrix $[s \times s]$
E_1	Output-tracking weight parameter matrix $[Ns \times Ns]$
G	QP gradient vector $[n_{\text{dec}} \times 1]$
G_{ss}	Steady-state gain matrix $[s \times n]$
H	QP Hessian matrix $[n_{\text{dec}} \times n_{\text{dec}}]$
J	MPC cost function
K	Kalman filter gain matrix $[s \times s]$
K_{kf}	Fixed Kalman gain from offline DARE solution $[s \times s]$
N	Prediction horizon
n	Number of control inputs
n_{dec}	Number of QP decision variables, $N \cdot n$
P	Kalman error covariance matrix $[s \times s]$
Q_y	Output tracking weight (scalar)
Q_{kf}	Kalman process noise covariance matrix
ε_i	Slack variable on the i-th soft output constraint
ρ	Soft-constraint slack penalty weight
R_{ref}	Stacked setpoint reference vector $[Ns \times 1]$
R_u	Move-suppression weight (scalar)
R_{kf}	Kalman measurement noise covariance matrix
s	Number of outputs / states
T_{mpc}	MPC execution sample period [s]
u_k	Control input vector at step k $[n \times 1]$
x_k	State vector at step k $[s \times 1]$
$x_{\text{aug},k}$	Augmented (Kalman-corrected) state at step k
y_k	Output vector at step k $[s \times 1]$
ΔU	Control-move decision vector $[n_{\text{dec}} \times 1]$
ε	Hessian regularization parameter
Λ	Move-suppression weight matrix, $\text{diag}(R_u)$
λ_i	i-th eigenvalue of the Hessian
μ	KKT Lagrange multiplier vector
Φ_F	Forced-response (input-to-output) prediction matrix $[Ns \times n_{\text{dec}}]$
Φ_R	Free-response (state-to-output) prediction matrix $[Ns \times s]$
τ	Dominant first-order time constant [s]

Abbreviations

DLE	Dry Low Emissions
DARE	Discrete Algebraic Riccati Equation
FCV	Fuel Control Valve
FSR	Fuel Stroke Reference
GT	Gas Turbine
IGV	Inlet Guide Vane
KKT	Karush–Kuhn–Tucker
MPC	Model Predictive Control
OAT	One-At-a-Time (perturbation)
QP	Quadratic Program

Subscripts and Superscripts

$(\cdot)_k$	Value at discrete time step k
$(\cdot)_{ss}$	Steady-state value
$(\cdot)^T$	Matrix or vector transpose
aug	Augmented (Kalman-corrected) quantity
fire	Firing (combustor exit) condition
mpc	Associated with the MPC controller
ref	Reference or setpoint value

Introduction

Gas turbines in service today are predominantly controlled using proportional-integral-derivative (PID) compensation arranged within a min/max selector structure (**Rowen 1983**; **Hagglund 2023**). Rather than a single independent loop, conventional turbine control coordinates several PID loops, each regulating a distinct objective such as startup, acceleration, load (speed), and exhaust or firing-temperature limiting, through low-value and high-value selectors that arbitrate which loop commands the fuel and airflow actuators at any given instant. This arrangement enforces both input constraints (actuator travel and rate limits) and output constraints (temperature, speed, and pressure-ratio limits) in a manner that is transparent, computationally inexpensive, and well proven across the installed fleet. PID function blocks are available in virtually every distributed control system (DCS) and turbine controller, and enhancement techniques such as feed-forward compensation, anti-windup logic, gain scheduling, and Smith predictors further extend their effective range (**Guzman 2024**). Modern multivariable design methods are likewise used when tuning these structures to address loop interaction and robustness. In practice, selector-based control works well and remains the industry standard.

Nevertheless, selector-based architectures face increasing difficulty as the number of simultaneously active objectives and cross-couplings grows. Because only one loop commands a given actuator at a time, a selector structure coordinates rather than jointly optimizes the competing objectives, and interactions between manipulated and controlled variables must be managed indirectly through tuning and loop-pairing choices. Hydrogen blending intensifies these challenges: hydrogen's lower volumetric energy density, wider flammability limits, and faster flame speed tightly couple fuel staging, total fuel flow, and inlet guide vane (IGV) position, so that hydrogen blend ratio, generator load, and turbine firing temperature must be satisfied simultaneously, often in the presence of blend-measurement delays (**Harper 2024**). Extending selector or cascade PID structures to coordinate these interacting objectives introduces tuning complexity that grows rapidly with the number of loops and can degrade performance when operating conditions shift.

Model Predictive Control (MPC) offers a complementary approach that can address these specific gaps. By embedding an explicit process model within the controller,

DOI 10.3384/ECP219 258

MPC predicts future system behavior over a receding horizon and computes control actions that jointly optimize all channels while respecting inequality constraints on both inputs and outputs within a single formulation, rather than arbitrating between separately tuned loops (**Wang 2009**). MPC has demonstrated advantages in managing multivariable interactions, respecting hard constraints, and compensating for measurement delays in industrial applications including power generation (**EPRI 2019**), and several DCS vendors now offer MPC function blocks that make such multivariable control increasingly accessible to plant operators. MPC is not a wholesale replacement for well-designed selector logic; rather, it provides a systematic means of coordinating the strongly coupled, constraint-limited objectives introduced by hydrogen blending. It does, however, require more computational resources, an accurate internal model of the process, and careful attention to constraint formulation and solver robustness, each of which is central to this paper.

In prior work presented at ASME Turbo Expo 2026 (**Harper 2026**), the authors described the development and initial implementation of a native Modelica-based MPC for multi-variable gas turbine control with hydrogen fuel blending. That effort was motivated by the absence of a publicly available, general-purpose MPC package in the current Modelica language, and by practical difficulties encountered during earlier co-simulation studies in which a Modelica gas turbine model was exported as a Functional Mock-up Unit (FMU) and controlled by an MPC implemented in MATLAB/Simulink (**Harper 2024**). The co-simulation approach, while functional, introduced data-exchange latency and solver synchronization overhead that complicated direct comparison between traditional PID control (implemented in Modelica) and advanced MPC strategies (implemented in MATLAB).

Relationship to Prior Work and Scope

The MPC architecture employed in this study; including the Kalman filter observer, the augmented state formulation, the prediction-horizon matrices, the active-set quadratic-programming (QP) solver, and the incremental SISO $\rightarrow$ constrained-SISO $\rightarrow$ MIMO development progression; was established in the companion ASME Turbo Expo paper (**Harper 2026**). The system-identification methodology, the gas turbine physical model, and the derivation of the MPC cost function and constraint formulation are documented there in full and are not reproduced here. The present paper *assumes* that architectural foundation and is deliberately scoped to the *data-driven refinement* required to advance the controller from demonstrated feasibility to robust closed-loop performance. Readers seeking the underlying formulation, the model construction, or the rationale for a native-Modelica implementation are referred to (**Harper 2026**).

Contributions of This Paper Relative to (Harper 2026)

Whereas the companion paper established architectural feasibility, the contributions of the present work are:

1. **Explicit KKT solver reformulation (S1 $\rightarrow$ S2).** The residual-based active-set heuristic of the prior solver is replaced by a full Karush–Kuhn–Tucker saddle-point system with explicit Lagrange-multiplier computation, enabling principled constraint activation/deactivation and post-simulation diagnostics.

2. **Dual regularization and line search.** Hessian regularization ($\varepsilon_H = 10^{-6}$) and $(2,2)$-block regularization ($\varepsilon_{\mathrm{KKT}} = 10^{-8}$), together with a step-length line search and active-set safety limits, eliminate the numerical singularities that terminated simulations under the original solver.

3. **Constraint alignment and anti-windup.** Identification and remediation of an IGV input-constraint mismatch that induced Kalman-filter integrator windup and persistent firing-temperature tracking failure.

4. **Refined open-loop system identification.** An updated step-test procedure and re-identified discrete B matrix that corrects the coupling structure used for prediction.

5. **Sample-period extension and fixed-gain observer.** Re-identification of the state-space model at $T_s = 1.0$ s and replacement of online Kalman covariance propagation with a fixed-gain observer derived from the discrete algebraic Riccati equation (DARE).

6. **Constraint-handling formalization.** An explicit treatment distinguishing *hard* input/actuator constraints from *soft* (slack-penalized) output constraints, together with bumpless-transfer initialization.

Rationale for a Native Modelica QP Solver

A reasonable question is why a bespoke active-set QP solver was developed in the Modelica language rather than interfacing an established C library such as qpOASES through the Modelica external-C function interface. The answer follows directly from the objective of this work and its predecessor (**Harper 2026**): to provide a *fully self-contained, native-Modelica* MPC capability that can be reused across Modelica tools and deployment targets without an external toolchain.

- **Elimination of cross-language coupling.** The motivation for a native implementation arose precisely from the data-exchange latency and solver-synchronization overhead encountered in earlier MATLAB/Simulink co-simulation (**Harper 2024**).

DOI **10.3384/ECP219 258**

Linking external C code reintroduces the cross-language build and synchronization fragility the project set out to remove.

- **Portability and reusability.** A native solver is exercised entirely by the Modelica translator and therefore travels with the model across tools, FMU exports, and target platforms, without requiring the host environment to compile or link third-party C sources.
- **Community value.** A publicly documented, language-native MPC component addresses the absence of such a capability in the current Modelica ecosystem and is directly usable in any Modelica systems model.

Accordingly, the QP solver is a *means* to a native-Modelica MPC, not the primary contribution of this paper. A performance benchmark against external C solvers such as qpOASES is intentionally out of scope, since raw solve speed is not the design objective, native integration and portability are the focus. Such a comparison is noted as possible future work in Section 9.

To eliminate these co-simulation barriers and to provide the starting point for a reusable MPC capability for the broader Modelica community, the authors developed a suite of MPC components entirely within the Modelica language: a discrete-time Kalman filter observer for state estimation, a quadratic programming (QP) solver based on the active-set method, and three controller variants of increasing complexity—a basic SISO MPC, a constrained SISO MPC with augmented state dynamics, and a 3×3 Multi-Input Multi-Output (MIMO) MPC with a configurable prediction and control horizon. System identification was performed using ridge regression on open-loop step-response data from a physics-based gas turbine model built in Modelon Impact using the Thermal Power Library (**Harper 2026**).

The prior testing demonstrated that the native Modelica MPC architecture was feasible: the 3×3 MIMO controller successfully executed multi-variable control on both a linear state-space surrogate (settling times of 10–15 seconds with zero steady-state error) and the full nonlinear physics-based gas turbine model (settling times of 60–80 seconds with small but non-zero steady-state errors in hydrogen blend and turbine firing temperature). However, the paper also identified several open issues: control oscillations during the first 20 seconds of simulation, residual steady-state offsets in turbine inlet temperature, and the observation that the QP solver's constraint handling relied on a residual-based heuristic rather than explicit Lagrange multiplier computation, limiting the ability to diagnose and resolve constraint-related performance issues.

The present work addresses the gap between feasibility and robust functional performance of the native Modelica MPC. Achieving feasibility, defined by demonstrating that all algorithmic components compile, execute, and produce physically reasonable control actions, is a necessary but insufficient condition for practical deployment. Bridging the remaining distance to robust performance required systematic investigation along two parallel tracks:

First, the data-driven track addressed fidelity of the internal plant model used by the MPC for prediction. This included updated open-loop system identification procedures, reconciliation of the QP Hessian scaling convention with the Modelica parameter interface, and alignment of the reference trajectory vector ordering with the controller's state indexing. Each of these corrections was identified through iterative closed-loop testing and post-simulation diagnosis of internal MPC states.

Second, the algorithmic track addressed robustness of the constrained optimization solver. The active-set QP solver was reformulated to employ a full KKT system with explicit Lagrange multiplier computation, replacing the residual-based constraint management of the original implementation. Two regularizations were introduced: Hessian regularization to guarantee positive definiteness under small eigenvalue conditions, and regularization to eliminate numerical singularity when active constraint normals become linearly dependent which occured when state constraints at adjacent prediction horizon steps produced nearly parallel rows in the constraint matrix. A step-length line search was added to prevent constraint boundary violations during update (preventing one constraint action from violating another), and active-set safety limits were introduced to ensure robustness under high-constraint-count scenarios.

Additionally, analysis of input constraint alignment between the MPC formulation and the external plant model constraints revealed that a mismatch on the IGV channel lower bound caused integrator windup that corrupted the Kalman state estimate and produced persistent firing temperature tracking failures.

The remainder of this paper is organized as follows. Section 1 summarizes the gas turbine model, the MPC architecture from (**Harper 2026**), and the starting-point performance that motivates the present refinements. Section 2 describes the updated system identification procedure and its impact on the discrete state-space model. Section 3 presents the reformulated QP solver with explicit KKT solve, dual regularization, and line search. Section 4 analyzes constraint handling, the IGV windup mechanism, and anti-windup remediation. Section 5 addresses initialization and bumpless transfer. Section 7 presents results from the iterative simulation campaign, including detailed analysis of the best configuration and QP solver diagnostics. Section 8 discusses practical lessons for MPC implementation in Modelica. Section 9 offers conclusions and identifies directions for future work.

DOI **10.3384/ECP219 258**

1 Background and Prior Work

1.1 Gas Turbine Model

The plant model used in this study is a single-shaft, open-cycle gas turbine constructed in Modelon Impact using components from the Thermal Power Library. The model comprises a compressor with inlet guide vanes (IGVs), a single-can combustor with separate natural gas and hydrogen fuel injection circuits, a turbine, and a synchronous generator connected to an infinite bus. Fuel delivery is managed through four actuators: a hydrogen flow control valve (H_2 FCV), a blended gas fuel supply regulator (FSR) that governs total fuel flow, a natural gas and hydrogen pressure control valve (PCV), and a natural gas shut off valve (SOV) that closes in the event of a 100% hydrogen demand. Figure 1 shows the major components of the GT System Model.

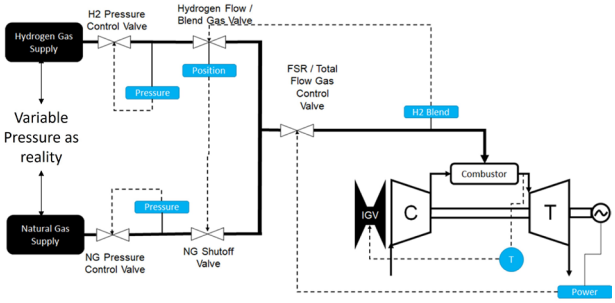


Figure 1. Gas Turbine System Model.

Hydrogen is introduced upstream of the combustor through a dedicated supply header. The volumetric hydrogen blend fraction is defined as

$$x_{H_2} = \frac{\dot{V}_{H_2}}{\dot{V}_{H_2} + \dot{V}_{NG}}, \tag{1}$$

where $\dot{V}_{H_2}$ and $\dot{V}_{NG}$ denote the volumetric flow rates of hydrogen and natural gas, respectively. The model captures the thermodynamic impact of varying fuel composition on combustor heat release, turbine inlet temperature, and exhaust conditions.

1.2 MPC Architecture from GT2026-176679

The MPC architecture developed in the Turbo Expo paper (**Harper 2026**) consists of three principal components implemented entirely in the Modelica language:

Kalman Filter Observer. A discrete-time linear Kalman filter estimates the internal state vector $\hat{x}_k$ from noisy measurements. The predict–correct cycle is

$$\hat{x}_{k|k-1} = A\,\hat{x}_{k-1|k-1} + B\,u_{k-1}, \tag{2}$$

$$K_k = P_{k|k-1}C^T\left(C\,P_{k|k-1}C^T + R_{kf}\right)^{-1}, \tag{3}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\left(y_k - C\,\hat{x}_{k|k-1}\right), \tag{4}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, and $C \in \mathbb{R}^{s\times n}$ are the discrete state-space matrices, R_{kf} is the measurement noise covariance, and K_k is the Kalman gain.

MPC Controller. The controller solves a receding-horizon optimization at each sample instant. Given the current augmented state $x_{\text{aug},k} = [\hat{x}_k^T,\ u_{k-1}^T]^T$, the cost function over a prediction horizon of N steps is

$$\begin{aligned} J = &\frac{1}{2}\Delta U^T\left(\Phi_F^T E_1\,\Phi_F + \Lambda\right)\Delta U \\ &+ \left(\Phi_F^T E_1\left(\Phi_R\,x_{\text{aug},k} - R_{\text{ref}}\right)\right)^T\Delta U, \end{aligned} \tag{5}$$

where $\Delta U = [\Delta u_k^T, \Delta u_{k+1}^T, \ldots, \Delta u_{k+N-1}^T]^T$ is the stacked vector of control increments, Φ_F and Φ_R are the forced and free response prediction matrices, E_1 is the block-diagonal output weighting matrix, Λ is the control effort penalty, and R_{ref} is the stacked reference trajectory.

The optimization is cast as a quadratic program (QP):

$$\min_{\Delta U}\ \tfrac{1}{2}\Delta U^T H\,\Delta U + G^T\Delta U \quad \text{s.t.} \quad A_c\,\Delta U \le b_c, \tag{6}$$

where $H = \Phi_F^T E_1\,\Phi_F + \Lambda$ is the Hessian and $G = \Phi_F^T E_1(\Phi_R\,x_{\text{aug},k} - R_{\text{ref}})$ is the gradient. The inequality constraints $A_c\,\Delta U \le b_c$ encode limits on control increments (ΔU constraints), absolute input values (input constraints), and predicted output values (state constraints).

QP Solver. The original solver (designated S1) employed an active-set method that iterated over constraint activations and deactivations, applying a full primal step at each iteration without explicit computation of Lagrange multipliers.

1.3 Starting-Point Performance

(**Harper 2026**) reported that the 3×3 MIMO MPC achieved settling times of 10–15 seconds with zero steady-state error on the linear state-space surrogate, and settling times of 60–80 seconds with small but non-zero steady-state errors on the full nonlinear plant model. Key limitations included oscillatory behavior during the first 20 seconds after controller activation, persistent steady-state offsets in turbine firing temperature, and the observation that the QP solver occasionally failed to converge when multiple constraints became simultaneously active.

1.4 Experiment Matrix: S1 to S2

The refinement campaign reported in this paper is organized around two solver configurations:

- **S1 (Baseline):** The original active-set QP solver from (**Harper 2026**), with implicit constraint handling and full-step primal updates.
- **S2 (Refined):** The reformulated solver with explicit KKT saddle-point system, Hessian regularization, $(2,2)$-block regularization, step-length line search, and active-set safety limits.

DOI **10.3384/ECP219 258**

Both configurations were tested with progressively refined system identification data, prediction matrices, and constraint specifications, yielding multiple simulation configurations over the course of the study.

2 Open-Loop System Identification Refinement

2.1 Updated Testing Procedure

System identification was performed by applying step changes to each manipulated variable (MV) and recording the resulting transient responses of all three process variables (PVs). The three MV–PV pairings are:

- MV_1: H_2 FCV command $\rightarrow$ PV_1: hydrogen blend fraction, PV_2: generator power, PV_3: firing temperature;
- MV_2: FSR valve command $\rightarrow$ PV_1, PV_2, PV_3;
- MV_3: IGV position $\rightarrow$ PV_1, PV_2, PV_3.

Step magnitudes were selected to span the expected operating range without driving the plant into nonlinear saturation regions. Each step test was run for sufficient duration to capture the full settling behavior of the slowest channel.

Figure 2 presents the open-loop step response data used for system identification. Twenty-nine distinct operating points were tested by applying sequential step changes to each manipulated variable—hydrogen FCV, FSR, and IGV—while recording the resulting transient responses of all three process variables. The wide range of valve commands (0.01–1.0 for H_2 FCV, 0.01–0.50 for FSR, 0.01–0.60 for IGV) ensured that the identified B matrix captured the full coupling structure across the operating envelope. Notable features include the strong H_2 FCV $\rightarrow$ blend coupling (blend fraction exceeding 0.95 at high FCV commands), significant power swings (-20 to $+480$ MW reflecting uncontrolled operation), and wide temperature variation (765–1375 K), confirming the nonlinear plant behavior that the MPC's linear internal model must approximate.

2.2 Non-Dimensionalization Strategy

All inputs and outputs were normalized to a $[0, 1]$ range using

$$\tilde{z} = \frac{z - z_{\min}}{z_{\max} - z_{\min}}, \tag{7}$$

where $z_{\min}$ and $z_{\max}$ are the physical minimum and maximum values of each signal. The scaling ranges used for the three PVs were:

Variable	Min	Max	Units
H_2 blend fraction	0.0	1.0	–
Generator power	0	120	MW
Firing temperature	800	1800	K

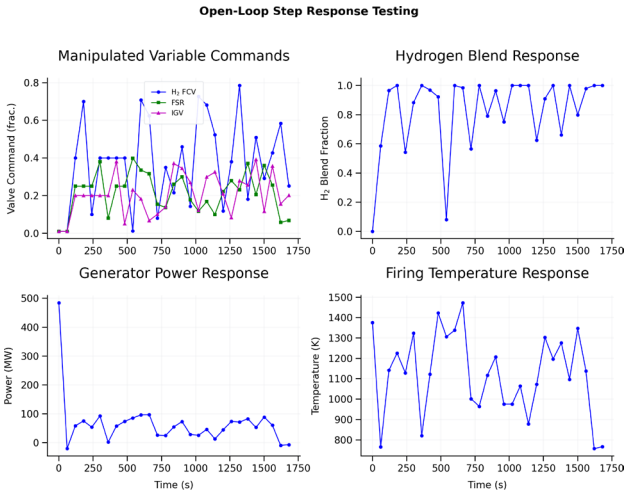


Figure 2. Open-loop step response testing. Top left: manipulated variable commands applied sequentially to each valve. Top right: hydrogen blend fraction response. Bottom left: generator power response. Bottom right: firing temperature response. The 29 operating points span the full actuator range used for discrete state-space identification.

Non-dimensionalization ensures that the QP cost function weights Q and Λ operate on comparable numerical scales across channels, and that the B matrix entries reflect true physical coupling rather than artifacts of unit differences.

2.3 Discrete State-Space Extraction via Ridge Regression

The continuous-time step-response data were sampled at the MPC sample period T_s and used to estimate the discrete state-space matrices A_d and B_d via ridge regression. The state transition is modeled as

$$x_{k+1} = A_d x_k + B_d u_k, \qquad y_k = C x_k, \tag{8}$$

where $C = I_{3\times 3}$ (identity, since the states are chosen to be the measured outputs in non-dimensionalized coordinates). The ridge regression objective is

$$\min_{A_d, B_d} \sum_{k=1}^{N_{\text{data}}} \left\| x_{k+1} - A_d x_k - B_d u_k \right\|^2 + \alpha \left(\|A_d\|_F^2 + \|B_d\|_F^2 \right), \tag{9}$$

where α is the regularization parameter and $\|\cdot\|_F$ denotes the Frobenius norm.

The identification above characterizes the discrete input-to-output gains that dominate steady-state tracking. The transient fidelity of the resulting state-space model is qualified indirectly: the diagonal entries of the identified A_d matrix (Eq. 17) correspond to first-order time constants consistent with the dominant plant dynamics (≈ 26 s on the power and temperature channels), and the closed-loop settling behavior reported in Section 7 provides an end-to-end check of the model's dynamic adequacy. A more formal dynamic validation, comparing the open-loop step responses of the identified model against the physics-based

DOI 10.3384/ECP219 258

plant on a channel-by-channel basis, is documented in the companion paper (**Harper 2026**) and is not reproduced here.

2.4 Updated B Matrix and Physical Interpretation

The updated B_d matrix obtained from the refined identification procedure is

$$B_d = \begin{bmatrix} 3.92\times10^{-4} & 9.41\times10^{-5} & 9.79\times10^{-5} \\ 7.41\times10^{-5} & 3.84\times10^{-4} & 3.54\times10^{-4} \\ 1.43\times10^{-4} & -2.93\times10^{-5} & -6.60\times10^{-5} \end{bmatrix}. \tag{10}$$

The diagonal entries confirm the pairings: the H_2 FCV has the strongest effect on hydrogen blend (B_{11}), the FSR has the strongest effect on power (B_{22}). The IGV has the strongest effect on power (B_{23}) and a weaker but important effect on firing temperature (B_{33}). Notably, the IGV-to-temperature coupling ($B_{33} = -6.60\times10^{-5}$) is approximately 5.8× weaker than the FSR-to-power coupling ($B_{22} = 3.84\times10^{-4}$), which explains the slower temperature tracking response observed in closed-loop results. It is important to note that the B_d matrix entries above correspond to $T_s = 0.01$ s. Because the discrete B_d matrix scales approximately linearly with the sample period for $T_s \ll \tau$ (where τ is the dominant plant time constant), re-identification at a longer sample period produces proportionally larger entries, as discussed in Section 6.

2.5 Sample Time and Horizon Considerations

The initial MPC configuration used $T_s = 0.01$ s with a prediction horizon of $N = 2$ steps, giving an effective look-ahead of only $N \times T_s = 0.02$ s. The dominant plant time constants, estimated from step-response data, are approximately 26 seconds for the power and temperature channels. Despite this apparent mismatch—the look-ahead is more than three orders of magnitude shorter than the plant dynamics—the controller functions because the incremental control action Δu_k at each step is small (bounded by ΔU constraints at ±0.05), and the receding-horizon mechanism effectively integrates these small corrections over many sample periods.

However, the short effective horizon limits the controller's ability to anticipate and preemptively avoid constraint saturation. Another strategy, explored in Section 6, is to increase T_s to a value commensurate with the plant dynamics and re-identify the state-space model at the new rate.

3 QP Solver Refinement: From Basic Active-Set to Regularized KKT

3.1 Original Solver Limitations (S1 Baseline)

The S1 solver implemented a standard active-set iteration: at each step, it identified the most violated inactive constraint, added it to the working set, solved the equality-constrained subproblem, and removed constraints with negative residuals. Three limitations were improved in this refinement:

1. **Implicit constraint handling:** The orginal QP solver did not compute explicit Lagrange multipliers λ_i for active constraints. Instead, constraint deactivation decisions were based on residual magnitudes, which provided less diagnostic information and could lead to premature constraint dropping.

2. **Full-step update:** The primal variable was updated regardless of whether the full step violated any inactive constraint. This could cause the iterate to jump across constraint boundaries, requiring additional iterations to recover.

3. **Singular KKT matrix:** When two or more active constraints had nearly parallel normal vectors, which occurred when state constraints at adjacent prediction steps ($k+1$ and $k+2$) produced nearly identical rows in the constraint matrix, the KKT system became singular and `Modelica.Math.Matrices.solve` threw an "A is singular" error, terminating the simulation.

3.2 Explicit KKT Saddle-Point Formulation

The S2 solver replaces the implicit residual-based approach with an explicit KKT system. At each active-set iteration, the following saddle-point system is assembled and solved:

$$\underbrace{\begin{bmatrix} H_{\text{reg}} & A_{\text{act}}^T \\ A_{\text{act}} & -\varepsilon_{\text{KKT}} I \end{bmatrix}}_{\text{KKT matrix}} \begin{bmatrix} \Delta x \\ \lambda \end{bmatrix} = \begin{bmatrix} -\nabla f \\ b_{\text{act}} - A_{\text{act}} x \end{bmatrix}, \tag{11}$$

where $H_{\text{reg}} = H + \varepsilon_H I$ is the regularized Hessian, A_{act} is the submatrix of active constraint rows, λ is the vector of Lagrange multipliers for the active constraints, and $\nabla f = H_{\text{reg}} x + G$ is the gradient of the objective at the current iterate.

The explicit Lagrange multipliers provide two key capabilities: (1) a constraint is dropped from the active set when its multiplier $\lambda_i < 0$, indicating that the constraint is not binding at the optimum; and (2) the multiplier values serve as diagnostic signals that quantify the "cost" of each active constraint, enabling the identification which constraints most limit performance.

3.3 Hessian Regularization

The $(1,1)$ block of the KKT matrix is regularized as

$$H_{\text{reg}} = H + \varepsilon_H I, \qquad \varepsilon_H = 10^{-6}. \tag{12}$$

This ensures that H_{reg} is strictly positive definite even when H has small eigenvalues. The eigenvalue spectrum of H in this application is dominated by the diagonal entries of Λ (the control effort penalty), which are on the order of 10^{-2}. With $\varepsilon_H = 10^{-6}$, the regularization shifts the

DOI 10.3384/ECP219 258

smallest eigenvalue by less than 0.01%, producing negligible perturbation of the optimal solution while guaranteeing numerical stability of the linear solve.

3.4 $(2,2)$-Block Regularization

A critical modification in the S2 solver is the addition of $-\varepsilon_{\text{KKT}} I$ to the $(2,2)$ block of the KKT matrix, with $\varepsilon_{\text{KKT}} = 10^{-8}$.

Without this regularization, the $(2,2)$ block is zero, and the KKT matrix is singular whenever the active constraint normals (the rows of A_{act}) are linearly dependent. This condition arises in the present application because the state constraints at prediction steps $k+1$ and $k+2$ are derived from the same Φ_F matrix with nearly identical row structures. The difference between the Φ_F rows at step 1 and step 2 is proportional to $(A_d - I)$, which has small entries when T_s is much smaller than the plant time constants.

The regularization is mathematically equivalent to relaxing each active constraint on the order of ε_{KKT}, a perturbation that is negligible relative to the physical constraint tolerances but sufficient to guarantee nonsingularity of the saddle-point system.

3.5 Step-Length Line Search

After computing the primal direction Δx, the S2 solver determines the maximum step length $\alpha^* \in (0, 1]$ that maintains feasibility of all inactive constraints:

$$\alpha^* = \min\left(1,\ \min_{i \notin \mathscr{A}} \left\{ \frac{b_i - a_i^T x}{a_i^T \Delta x} \;\middle|\; a_i^T \Delta x > 0 \right\}\right), \qquad (13)$$

where $\mathscr{A}$ is the current active set and a_i^T is the i-th row of the constraint matrix. The iterate is then updated as $x \leftarrow x + \alpha^* \Delta x$. This prevents the iterate from crossing constraint boundaries in a single step, reducing the number of active-set changes required for convergence and improving numerical stability.

3.6 Active-Set Safety and Iteration Limits

Two additional safeguards were introduced:

- **Active-set size limit:** The number of simultaneously active constraints is clamped to `maxActive` $= 50$, preventing array overflow when many constraints activate simultaneously during large transients.
- **Iteration limit:** The maximum iteration count was increased from 50 (S1) to 200 (S2) to accommodate the tighter constraint scenarios encountered with properly aligned input constraints.

3.7 Lagrange Multiplier Diagnostics

The explicit computation of λ_i for each active constraint enables a new class of post-simulation diagnostics. Logging the Lagrange multiplier vector at each sample instant allows:

- Identification which constraints are persistently active (non-zero λ_i over extended time intervals);
- Quantification the performance cost of each constraint (large $|\lambda_i|$ indicates significant impact on the objective);
- Detection near-infeasibility (very large $|\lambda_i|$ values suggest the constraint set is close to becoming infeasible);
- Informing constraint relaxation decisions (e.g., widening ΔU limits on the channel with the largest $|\lambda_i|$).

4 Constraint Handling and Anti-Windup

4.1 IGV Input Constraint Alignment

The most impactful single correction in the refinement campaign was the alignment of the MPC's IGV input constraint with the plant's physical minimum. The original MPC formulation specified the IGV lower bound as $u_3^{\min} = 0.0$ (fully closed), consistent with the normalized input range. However, the plant model implemented a max-select clamp that prevents the physical IGV position from dropping below approximately 0.04 (in normalized coordinates), corresponding to the minimum airflow.

This mismatch created an integrator windup mechanism: the MPC's internal model predicted that setting the IGV command to values between 0.0 and 0.04 would further reduce airflow and thereby affect firing temperature and power, but the plant's clamp prevented any actual change. The Kalman filter, observing that the predicted temperature change did not materialize, accumulated a growing residual, progressively corrupting the state estimate and producing persistent tracking errors.

Analysis of the simulation data confirmed this mechanism: in the best-performing configuration, the MPC commanded IGV values below the 0.04 plant minimum for 236 out of 701 sample points after controller activation (Figure 3). Aligning the MPC input constraint to $u_3^{\min} = 0.04$ eliminated the windup and restored stable temperature tracking.

4.2 ΔU Rate Constraints

The control increment is bounded symmetrically:

$$-\Delta u^{\max} \le \Delta u_{i,k} \le +\Delta u^{\max}, \qquad \Delta u^{\max} = 0.05, \qquad (14)$$

for all three channels and both prediction steps. These rate limits prevent large single-step actuator movements that could excite nonlinear dynamics or exceed physical actuator slew rates. The ± 0.05 value was selected empirically as a compromise between responsiveness (larger limits allow faster tracking) and stability (smaller limits reduce the risk of oscillation and constraint cycling in the QP solver).

DOI 10.3384/ECP219 258

Table 1. Consolidated MPC design parameters and the rationale for their selection. Values correspond to the final configuration (Result 35) at $T_s = 0.01$ s unless noted.

Parameter	Value	Selection rationale
Sample period T_s	0.01 s	Small relative to the ≈ 26 s plant time constants, keeping per-step linearization error negligible (Sec. 2.5).
Prediction horizon N	2	Minimal horizon sufficient for stability at $T_s = 0.01$ s; larger N deferred to future work.
ΔU rate limit	± 0.05	Empirical compromise between tracking speed and avoidance of oscillation and QP constraint cycling (Sec. 4).
IGV lower bound $u_3^{\min}$	0.04	Aligned to the plant max-select airflow clamp to remove integrator windup (Sec. 4).
State (output) bounds	± 5.0	Deliberately wide to avoid QP infeasibility far from the linearization point (Sec. 4).
Output weight Q_y	0.5	Weights output tracking error; scaled on non-dimensional outputs so channels are comparable.
Move suppression R_u	0.01	Penalizes control effort; sets the smallest Hessian eigenvalues (order 10^{-2}).
Ridge parameter α	0.01	Regularizes the least-squares fit of A_d/B_d (Eq. 9).
Hessian reg. ε_H	10^{-6}	Guarantees positive-definite H; shifts smallest eigenvalue by $< 0.01\%$ (Sec. 3).
KKT $(2,2)$ reg. ε_{KKT}	10^{-8}	Removes saddle-point singularity from near-parallel active constraint normals (Sec. 3).
Active-set limit	50	Prevents array overflow during large transients (Sec. 3).
Iteration limit	200	Increased from 50 (S1) to accommodate tighter constraint scenarios (Sec. 3).
MPC enable time	50 s	Allows the plant to reach a stable operating point and the Kalman filter to converge (Sec. 5).

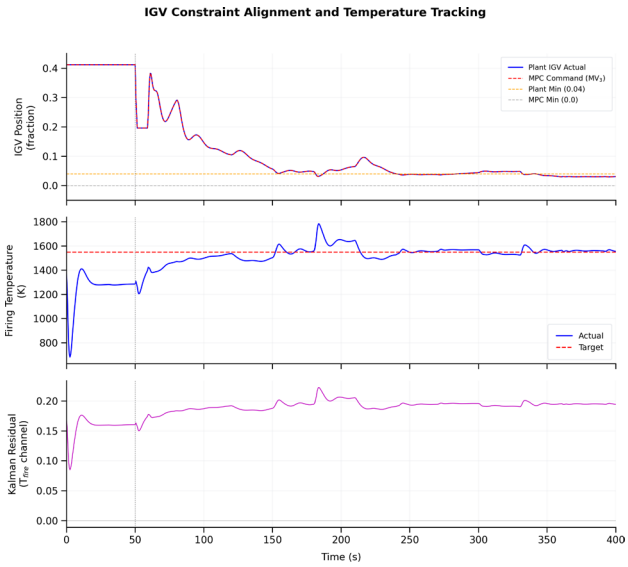


Figure 3. IGV constraint alignment and its effect on firing temperature tracking and Kalman residuals. Top: plant IGV position and MPC command, showing the gap between the MPC's commanded minimum and the plant's physical minimum. Middle: firing temperature tracking. Bottom: Kalman filter residual on the temperature channel.

4.3 Constraint Classification: Hard Inputs versus Soft Outputs

A central design decision in the refined formulation is the distinction between input constraints, which are enforced as hard bounds, and output constraints, which are best enforced softly. The two classes differ in both their physical origin and their effect on QP feasibility.

Input and rate constraints are physical: the actuators cannot travel beyond their mechanical stops, and the fuel and airflow valves have finite slew rates. These bounds are therefore retained as hard constraints. Critically, hard input constraints can always be satisfied. For any admissible current input, there exists a control increment that keeps the commanded input within its bounds, so the input-constraint set is never empty and cannot, by itself, render the QP infeasible. The valve position limits (normalized to $[0,,1]$), the IGV lower bound aligned to the plant airflow clamp at 0.04 (Sec. 4), and the symmetric rate limits $\Delta u^{\max} = 0.05$ are all enforced this way.

Output (state) constraints are fundamentally different. They bound a predicted quantity, $\hat{y}i,k+j = \Phi_F,\Delta U + \Phi_R,x\text{aug},k$, computed from a linear model of a nonlinear plant. Far from the linearization point, the prediction error can be large enough that no control increment satisfies both the hard input limits and every predicted-output bound simultaneously. Enforcing output constraints as hard bounds in that situation empties the feasible set and causes the QP to fail, which is the origin of the early-termination behavior observed in the S1 configuration (for example, Result 28).

For this reason the refined formulation treats output constraints as soft. Each output bound is relaxed by a nonnegative slack variable that is heavily penalized in the cost, so the predicted output is permitted to exceed its band only when doing so is unavoidable, and the optimizer recovers the hard-constrained solution whenever one exists. The

DOI 10.3384/ECP219 258

output constraint becomes

$$y_i^{\min} - \varepsilon_i ; \leq ; \hat{y}i, k+j ; \leq ; y_i^{\max} + \varepsilon i, \qquad \varepsilon_i \geq 0, \qquad (15)$$

and the QP objective is augmented with a quadratic penalty on the slack vector ε,

$$\min_{\Delta U, , \varepsilon} ; \tfrac{1}{2}, \Delta U^T H, \Delta U + G^T \Delta U + \rho, \varepsilon^T \varepsilon, \qquad \varepsilon \geq 0, \quad (16)$$

where the penalty weight ρ is chosen large enough that the slack is driven to zero whenever the hard-constrained problem is feasible.

It is worth emphasizing that softening the predicted-output constraint does not permit the physical plant to leave its admissible range. The hydrogen blend fraction, for instance, is an output (PV_1) bounded physically to $[0, , 1]$ by the mixing process and enforced through the hard limits on the H_2 FCV input; the slack in Eq. (15) only relaxes the controller's internal prediction, never the plant. The physical guarantee is provided by the hard input constraints, while the soft output constraints protect solver feasibility.

In the results reported here, output constraints were kept deliberately wide (bounds of ± 5.0 in normalized coordinates) as an interim measure that achieves the same practical effect: the output bounds never bind, so the QP remains feasible throughout each simulation. The explicit slack formulation of Eqs. (15) and (16) formalizes this behavior and is the intended refinement for operation with tighter, physically meaningful output limits.

4.4 Offset-Free Tracking

The present formulation augments the state with the previous input to penalize control increments, but it does not include an explicit integrating disturbance model. As a result, steady-state offsets can persist when the internal model gain differs from the true plant gain at the operating point. This is visible in the best configuration as a residual firing-temperature offset of approximately 6 K (Sec. 7), the channel with the weakest actuator authority and therefore the greatest sensitivity to gain mismatch.

Offset-free tracking can be recovered by augmenting the observer with a disturbance state, typically an integrating input or output disturbance, that is estimated by the Kalman filter and used to bias the target calculation so that the steady-state error is driven to zero regardless of model gain error. This is a well-established extension of linear MPC and is compatible with the native Modelica architecture, since it adds only additional states to the existing augmented model and observer. Implementing and evaluating disturbance-model augmentation is identified as future work.

5 Initialization and Bumpless Transfer

5.1 MPC Enable Time

The MPC controller is held inactive for the first $t_{\text{enable}} = 50$ seconds of simulation, during which the plant operates under its default open-loop or baseline control. This delay serves two purposes: (1) it allows the plant to reach a stable initial operating point from which the linear MPC model provides reasonable predictions, and (2) it allows the Kalman filter to converge to a meaningful state estimate before the MPC begins issuing control actions based on that estimate.

5.2 Initial Setpoint and u_{prev} Alignment

At the moment of MPC activation ($t = t_{\text{enable}}$), the controller's internal variables must be aligned with the plant's current state:

- The setpoints r_i are initialized to the current measured values of the corresponding PVs, ensuring that the initial tracking error is zero and the first Δu is driven only by the difference between the current and target setpoints;
- The previous control action u_{prev} is initialized to the current valve positions, ensuring that $\Delta u_0 = u_0 - u_{\text{prev}} = 0$ at the first active sample.

Failure to align u_{prev} produces a large spurious Δu at the first step, which can drive the plant into saturation and trigger a cascade of constraint activations in the QP solver.

6 Sample-Period Extension to $T_s = 1.0$ s

6.1 Motivation

The discussion in Section 2.5 identified the mismatch between the MPC look-ahead ($NT_s = 0.02$ s at the original sample rate) and the dominant plant time constants (≈ 26 s) as a fundamental limitation. To evaluate the impact of a longer sample period on closed-loop performance, the discrete state-space model was re-identified at $T_s = 1.0$ s using the same open-loop step-response data and ridge-regression procedure described in Section 2, and the Kalman filter was simplified to a fixed-gain observer.

6.2 Re-Identified Matrices at $T_s = 1.0$ s

The re-identified discrete matrices at $T_s = 1.0$ s are

$$A_d^{(1s)} = \begin{bmatrix} 0.9884 & -0.0190 & 0.0186 \\ -0.0178 & 0.9275 & 0.0161 \\ -0.0088 & -0.0264 & 0.9858 \end{bmatrix}, \qquad (17)$$

$$B_d^{(1s)} = \begin{bmatrix} 0.01784 & 0.00167 & 0.01254 \\ -0.00326 & 0.15670 & -0.01805 \\ 0.00005 & 0.02114 & -0.00186 \end{bmatrix}. \qquad (18)$$

DOI 10.3384/ECP219 258

The diagonal entries of $A_d^{(1s)}$ confirm stable poles (all less than unity), and the $B_d^{(1s)}$ entries are approximately two orders of magnitude larger than their $T_s = 0.01$ s counterparts, consistent with the linear scaling $B_d \approx T_s B_c$ for $T_s \ll \tau$. Notably, the dominant coupling $B_{22}^{(1s)} = 0.157$ (FSR $\rightarrow$ power) is now approximately $9\times$ larger than $B_{11}^{(1s)} = 0.018$ (H_2 FCV $\rightarrow$ blend), reflecting the relative authority of the two actuators at the one-second time scale. The IGV $\rightarrow$ temperature entry $B_{33}^{(1s)} = -0.00186$ retains the physically correct negative sign (increasing IGV opening reduces firing temperature by increasing airflow), and its small magnitude continues to explain the slower temperature tracking response.

6.3 Fixed-Gain Kalman Filter

At the original $T_s = 0.01$ s, the Kalman filter propagated the error covariance P and computed the gain K_k online at each sample instant (Equations 2–4). For the $T_s = 1.0$ s configuration, the online covariance propagation was replaced by a fixed gain K_{kf} computed offline from the steady-state solution of the discrete algebraic Riccati equation (DARE):

$$P_\infty = A P_\infty A^T + Q_{\text{kf}} - A P_\infty C^T \left(C P_\infty C^T + R_{\text{kf}}\right)^{-1} C P_\infty A^T, \tag{19}$$

$$K_{\text{kf}} = P_\infty C^T \left(C P_\infty C^T + R_{\text{kf}}\right)^{-1}. \tag{20}$$

This simplification eliminates the matrix inverse at each sample instant, removes the Q_{kf} and R_{kf} input connectors from the Modelica model, and ensures that the observer gain is always well-conditioned. The Modelica implementation uses a `when sample(0, sP)` clause to execute the predict–correct cycle exactly once per sample period, preventing the over-prediction that would occur if the algorithm section executed at the solver's internal time steps.

6.4 Prediction Matrix Update

The forced-response matrix Φ_F was recomputed from $B_d^{(1s)}$. Because Φ_R depends only on A_d (through CA and CA^2 for $N = 2$), and $A_d^{(1s)}$ was re-identified from the same data, Φ_R was also updated accordingly. The QP Hessian H, gradient G, and constraint matrices were regenerated with the new prediction matrices, and all parameters were verified against the Python reference values before simulation.

7 Results and Performance Analysis

7.1 Iterative Improvement Timeline

Table 2 and Figures 4 and 5 summarize the progression from the initial closed-loop configuration through the final refined result. Four milestones are highlighted:

- **Early configuration (Result 11):** Original A/B matrices and S1 solver. The H_2 blend saturates near 1.0 with the FCV driven to its upper limit, load diverges to over 250 MW, and firing temperature reaches 4826 K indicating fundamental errors in the prediction matrices. The FSR valve saturates for 700 of 701 post-enable sample points.
- **Mid-campaign (Result 28):** Re-identified B matrix and corrected non-dimensionalization. Load tracking improves markedly (2.16 MW final error), but persistent H_2 FCV saturation and a 968 K temperature offset reveal the IGV constraint mismatch described in Section 4. The simulation terminates early at $t = 351.5$ s due to a QP solver singularity.
- **Corrected matrices (Result 31):** Hessian scaling aligned. Firing temperature offset reduces to 47.6 K, but H_2 FCV remains saturated at 1.0, and load error is 28.27 MW—evidence that the unconstrained IGV lower bound continues to corrupt the Kalman state estimate.
- **Final configuration (Result 35):** S2 solver with KKT regularization, IGV constraint aligned to the plant minimum, and updated B matrix from refined open-loop testing. All three channels converge: H_2 error 0.70%, load error 0.53 MW, temperature offset 6 K. The H_2 FCV operates well within its range (0.047–0.458), confirming that the constraint and matrix corrections eliminated the saturation mechanism.

Table 2. Performance metrics at selected refinement milestones. SS = steady-state; RMSE computed over $t \geq 50$ s.

Metric	R11	R28	R31	R35
Solver configuration	S1	S1	S1	S2
H_2 SS error (%)	17.4	40.0	42.9	0.70
Load SS error (MW)	16.87	2.16	28.27	0.53
T_{fire} SS offset (K)	3029	968	47.6	6.0
H_2 RMSE (SS)	0.244	0.358	0.350	0.025
Power RMSE (MW)	142.0	16.6	15.2	3.1
T_{fire} RMSE (K)	2963	738	132	71
FSR saturation (pts)	700/701	17/604	17/701	16/701
H_2 FCV saturated	Yes	Yes	Yes	No
Sim. duration (s)	400	351.5	400	400

7.2 Best Result: Detailed Analysis

The final configuration whose detailed performance is shown in Figure 6 incorporates all corrections described in Sections 3–7: re-identified A_d/B_d matrices, aligned Hessian scaling, the S2 QP solver with KKT regularization, and IGV constraint alignment to the plant minimum. Compared with the earlier milestones shown in Figure 4, this configuration is the first to achieve simultaneous convergence on all three channels without H_2 FCV saturation.

The final steady-state performance is:

- **Hydrogen blend:** 0.705 vs. 0.700 target (0.70% error);

DOI 10.3384/ECP219 258

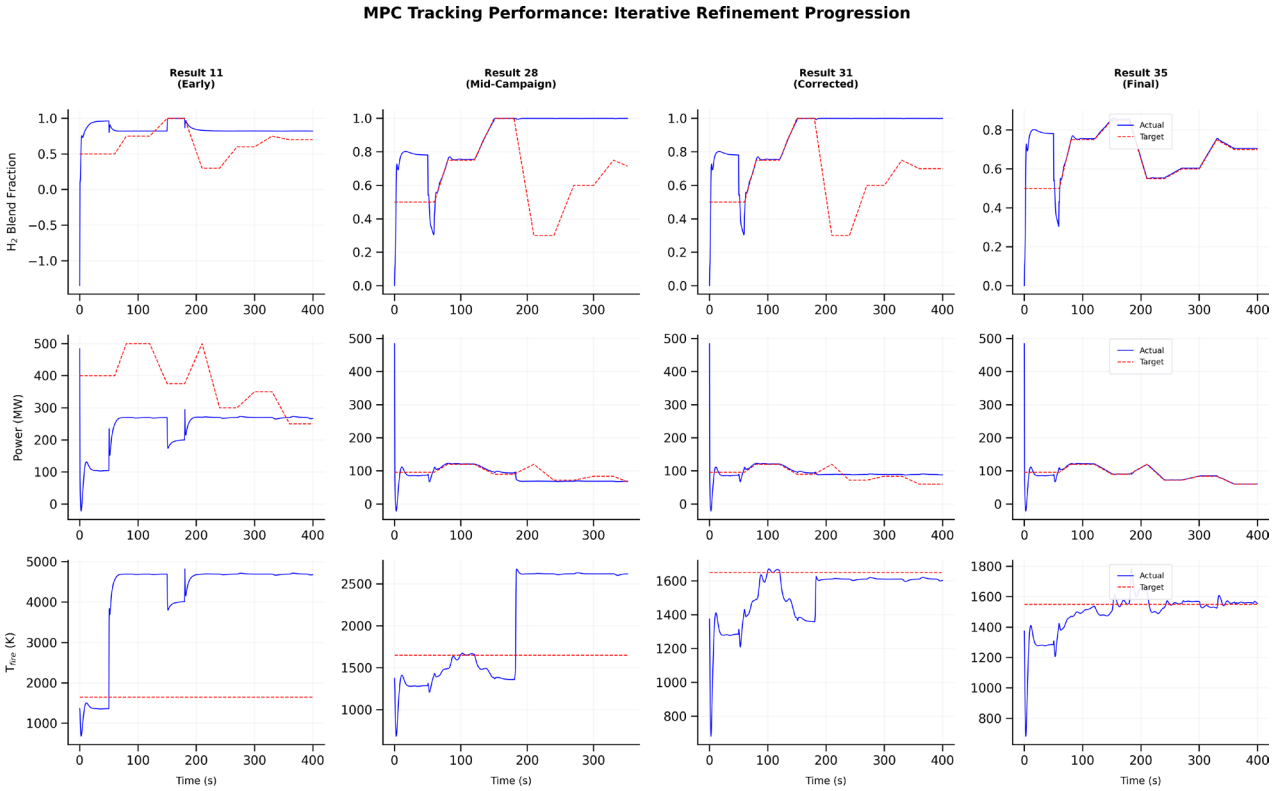


Figure 4. Process variable tracking across four refinement milestones. Rows: hydrogen blend fraction (top), generator power (middle), firing temperature (bottom). Columns progress from the early configuration (left) to the final configuration (right). Blue: plant actual; red dashed: setpoint target.

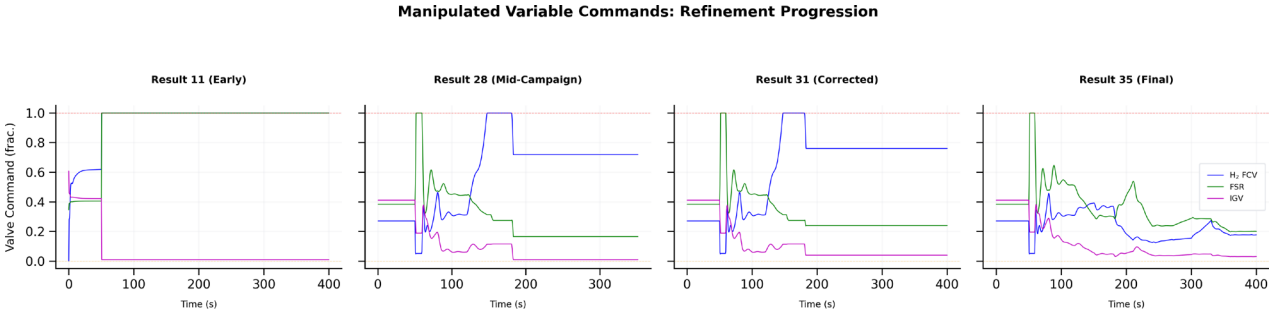


Figure 5. Manipulated variable commands across the same four milestones. The progression from near-permanent FSR saturation (left) to well-bounded valve operation (right) illustrates the cumulative impact of the matrix, constraint, and solver corrections.

DOI **10.3384/ECP219 258**

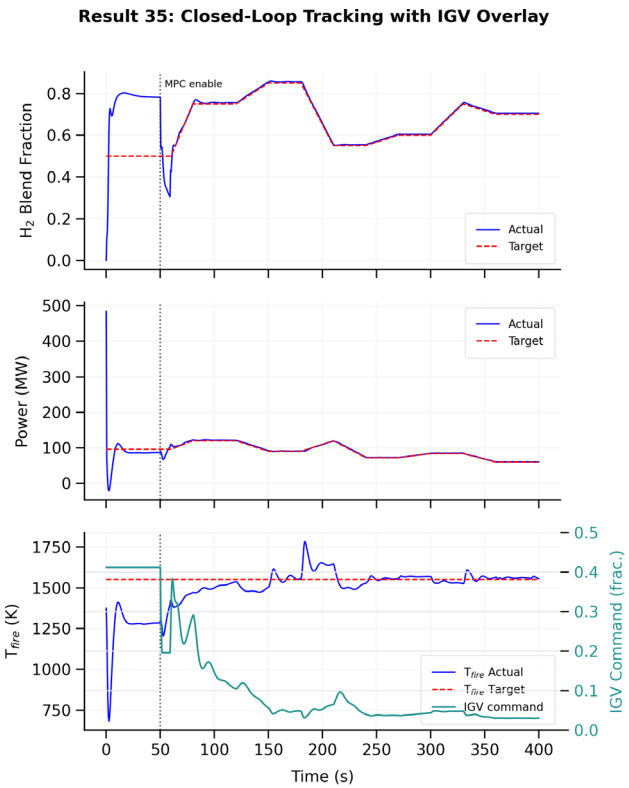


Figure 6. Result 35: closed-loop tracking of hydrogen blend ratio (top), generator power (middle), and firing temperature (bottom), with the IGV command overlaid on the firing-temperature axis to show the actuator responsible for temperature regulation. The MPC activates at $t = 50$ s (dashed vertical line).

- **Generator load:** 60.53 MW vs. 60.0 MW target (0.53 MW error);
- **Firing temperature:** 1556 K vs. 1550 K target (6.0 K offset).

Hydrogen blend settles within a 2% band at $t = 182.5$ s, load at $t = 331.5$ s, and firing temperature at $t = 337.5$ s.

The apparent weak coupling between firing temperature and generator power in Figure 6 merits comment, since a change in firing temperature would ordinarily be expected to move shaft power. Two effects explain the observed behavior. First, the two channels are regulated by different actuators with very different authority: power is controlled primarily by the FSR (with $B_{22} = 3.84 \times 10^{-4}$), whereas firing temperature is influenced most strongly by the IGV through airflow (with $B_{33} = -6.60 \times 10^{-5}$, roughly $5.8\times$ weaker). The controller therefore holds power on its setpoint by adjusting FSR while it trims temperature with the IGV, so the two setpoints are tracked semi-independently and the power trace shows little visible response to the temperature transient. The IGV command responsible for the temperature correction is plotted in Figure 7 and, for clarity, is overlaid on the temperature channel in Figure 6.

Second, and more fundamentally, the linear internal model does not capture the thermal and rotational inertia of the physical gas turbine. In the real machine, changes in firing temperature, load, and blend are moderated by the heat capacity of the hot-gas path and the rotor, so the rapid, tightly decoupled responses produced by the model may be faster and more idealized than a full-scale unit would exhibit. This model-inertia limitation, discussed further in Section 8, means that the present results should be read as a demonstration of the control architecture rather than a quantitative prediction of transient plant behavior.

Figure 7 shows the manipulated variable commands and control increments. The H_2 FCV operates within $[0.047, 0.458]$ with no saturation. The FSR valve briefly saturates at 1.0 during the initial transient (16 out of 701 sample points), corresponding to the large load ramp-up from 96 to 120 MW. The IGV operates within $[0.030, 0.412]$, approaching but not persistently sitting at the plant minimum.

7.3 QP Solver Convergence Diagnostics

The S2 solver's explicit Lagrange multiplier computation enables direct observation of constraint activity over the simulation. The ΔU constraints on the FSR channel (Δu_2) are the most frequently active, consistent with the large FSR movements required during load transients. The state constraints remain inactive throughout the simulation with the wide bounds (± 5.0) employed in the final configuration. The maximum number of QP iterations at any single sample instant was observed to be well below the 200-iteration limit, confirming that the regularized KKT system provides reliable convergence.

Table 3 characterizes the solver's convergence, infeasibility handling, and failure recovery directly, which are the

DOI 10.3384/ECP219 258

Table 3. QP solver diagnostics comparing the baseline (S1) and refined (S2) configurations. Values are drawn from the Result 35 simulation for S2 and the Result 28/31 runs for S1.

Diagnostic	S1 (baseline)	S2 (refined)
Constraint handling	Residual-based heuristic	Explicit KKT with Lagrange multipliers λ
Peak active-set size	Unbounded (array-overflow risk)	≤ 50 (clamped)
Iteration limit	50	200
Peak iterations / sample	Not logged	Well below the 200 limit
Singular KKT handling	None; standard-library solve aborts	$(2,2)$-block regularization $\varepsilon_{\text{KKT}} = 10^{-8}$
Step update	Full primal step	Line search $\alpha^* \in (0, 1]$
Warm starting	Cold-started each sample	Cold-started each sample (warm start is future work)
Infeasibility handling	Hard output bounds	Wide output bounds (± 5.0); soft-constraint slack planned
Most active constraint	–	Δu_2 (FSR rate)
State (output) constraints	–	Inactive with ± 5.0 bounds

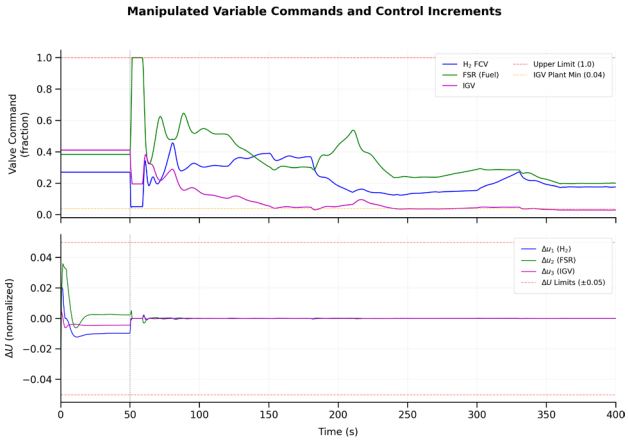


Figure 7. Result 35: Valve commands (top), including the IGV position, and MPC control increments Δu (bottom). Horizontal dashed lines indicate actuator limits and ΔU constraints (± 0.05). The IGV operates within $[0.030, , 0.412]$, approaching but not persistently sitting at the plant minimum.

properties most relevant to embedding a QP inside a native Modelica model. The refined solver converges in far fewer iterations than its limit, never triggers the singular-matrix abort that terminated the S1 runs, and completes the full simulation horizon. Consistent with the rationale in Section , these diagnostics are reported for the native solver on its own terms rather than as a head-to-head timing benchmark against external C libraries such as qpOASES; raw solve speed is not the design objective, and a cross-language comparison would reintroduce the very toolchain coupling the native implementation was created to avoid. A quantitative performance comparison against external solvers is identified as future work.

7.4 Temperature Tracking Discussion

Firing temperature is the slowest channel to settle (337.5 s) and retains the largest relative offset at the end of simulation. Two factors contribute to this behavior:

First, the IGV-to-temperature entry in the B matrix ($B_{33} = -6.60 \times 10^{-5}$) is approximately $5.8\times$ smaller than the FSR-to-power entry ($B_{22} = 3.84 \times 10^{-4}$), meaning that a unit change in IGV position produces a much smaller predicted temperature change than a unit change in FSR produces in power. The MPC therefore requires larger IGV movements to achieve a given temperature correction, and these movements are constrained by the ΔU limits.

Second, the IGV simultaneously affects both power (through airflow changes) and temperature (through the air-to-fuel ratio), creating a coupling that the MPC must balance. When the load target requires the IGV to be near its lower bound, the temperature channel loses effective control authority. Potential remedies include differential Q weighting (increasing Q_{33} to prioritize temperature tracking), reducing the control effort penalty λ_3 on the IGV channel, or increasing the prediction horizon to give the optimizer more room to plan multi-step temperature corrections.

7.5 Extended Sample Period Results ($T_s = 1.0$ s)

Figure 8 presents the closed-loop performance with $T_s = 1.0$ s, $N = 2$, the re-identified $A_d^{(1s)}/B_d^{(1s)}$ matrices, and the fixed-gain Kalman filter. The simulation spans 400 seconds with MPC activation at $t = 50$ s. Table 4 summarizes the tracking performance.

Table 4. Tracking performance at $T_s = 1.0$ s ($N = 2$) over $t > 50$ s. Metrics are computed against the time-varying setpoint trajectory.

Metric	H_2 Blend	Power [MW]	T_{fire} [K]
MAE	0.104	35.6	118.5
RMSE	0.151	38.9	141.2
Max $\lvert e \rvert$	0.645	60.1	395.2

Compared with the $T_s = 0.01$ s best result (R35), tracking accuracy is reduced: the hydrogen blend MAE increases from ≈ 0.025 to 0.104, the power MAE from 3.1 MW to 35.6 MW, and the firing temperature MAE from 71 K to 118.5 K. However, several observations sup-

DOI 10.3384/ECP219 258

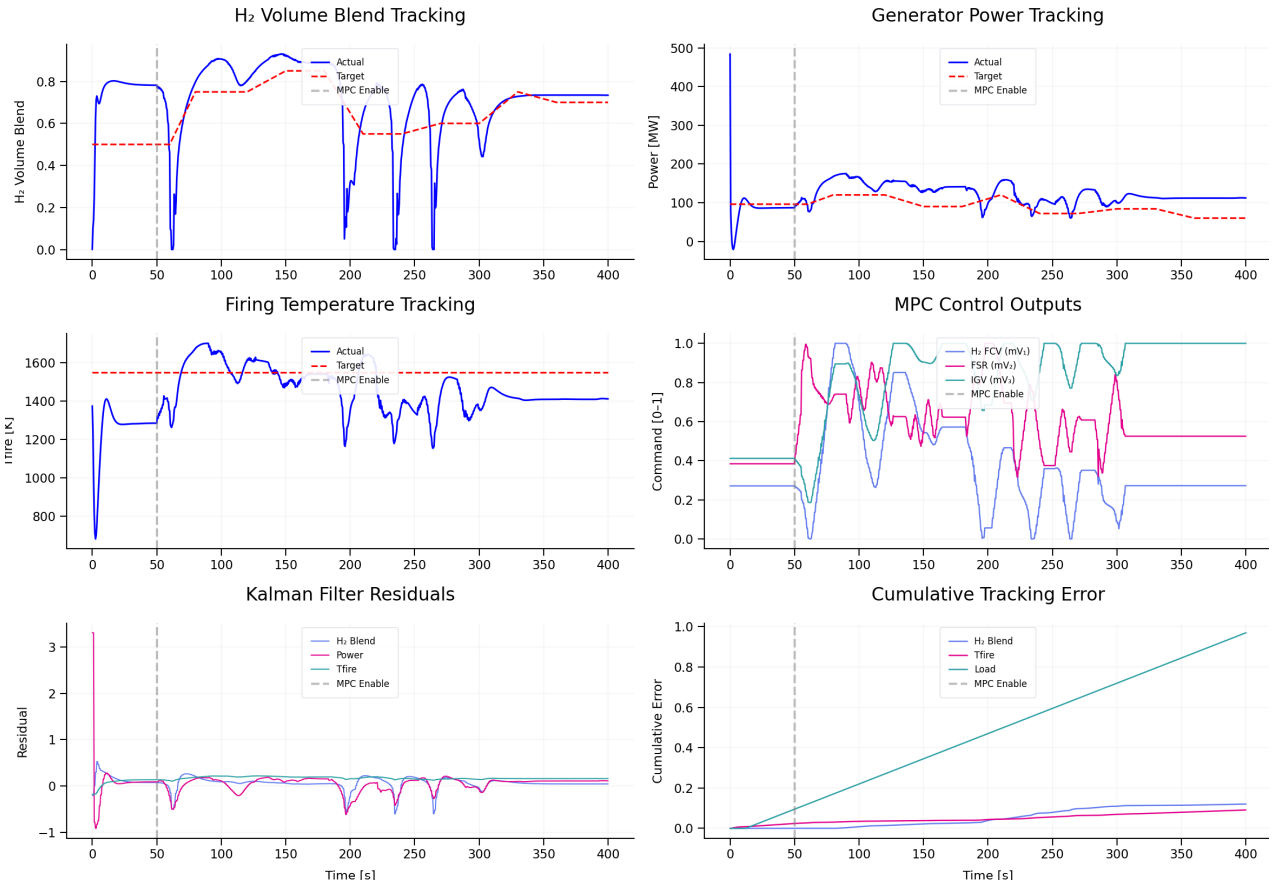


Figure 8. Closed-loop diagnostics at $T_s = 1.0$ s, $N = 2$. Top row: hydrogen blend (left) and generator power (right) tracking against time-varying setpoints. Middle row: firing temperature tracking (left) and MPC manipulated variable commands (right). Bottom row: Kalman filter residuals (left) and cumulative tracking error (right). The MPC activates at $t = 50$ s (dashed vertical line).

DOI **10.3384/ECP219 258**

port the value of this configuration as a foundation for further development:

1. **Stability:** The simulation completes the full 400 seconds without solver failure or divergence, confirming that the re-identified matrices, fixed-gain Kalman, and QP solver remain well-conditioned at the longer sample period.

2. **Physically meaningful gains:** The $B_d^{(1s)}$ entries are two orders of magnitude larger than their $T_s = 0.01$ s counterparts, producing QP gradient and Hessian values that are better scaled for the constraint boundaries. The forced-response matrix Φ_F now predicts output changes that are commensurate with the ΔU limits, rather than requiring thousands of incremental steps to produce observable effects.

3. **Bounded Kalman residuals:** The fixed-gain observer maintains bounded residuals throughout the simulation (Figure 8, bottom left), with mean residuals of 0.055 (H_2), 0.041 (power), and 0.171 (temperature), confirming that the DARE-based gain provides adequate state estimation without online covariance propagation.

4. **Active constraint management:** All three control outputs span a wide operating range (H_2 FCV: 0.0–1.0, FSR: 0.31–1.0, IGV: 0.19–1.0), indicating that the MPC is exercising its full control authority. The saturation observed at the actuator limits reflects the large setpoint excursions in the test trajectory rather than algorithmic failure.

The dominant source of tracking error is model mismatch: the linear state-space model, must predict plant behavior across a wide range of hydrogen blends (0.0–0.9), loads (60–120 MW), and temperatures (1000–1650 K). At operating conditions far from the linearization point, the gain magnitudes and cross-coupling ratios differ from the identified $B_d^{(1s)}$, producing prediction errors that the short horizon ($N = 2$, look-ahead $= 2$ s) cannot compensate. Increasing the prediction horizon to $N = 5$–10 (look-ahead 5–10 s) would allow the optimizer to plan multi-step corrections that partially compensate for gain mismatch, and is identified as a priority for future work.

8 Discussion: Practical Lessons for MPC in Modelica

8.1 Debugging in Equation-Based Environments

The most time-consuming aspect of the refinement campaign was not the algorithmic development itself but the diagnosis of failures in the closed-loop simulation. When the MPC produces incorrect control actions, the symptom (e.g., a PV diverging from its setpoint) is separated from the root cause by multiple layers: the plant model, the Kalman filter, the prediction matrices, the QP solver, and the constraint specification.

The key enabler for efficient debugging was the export of all internal QP state as logged variables in the Modelon Impact simulation output which enabled post-simulation analysis to pinpoint, for example, that a specific constraint was persistently active with a large $|\lambda|$, or that the gradient G had an unexpected sign on a particular channel.

8.2 Modelica-Specific Implementation Considerations

Implementing MPC natively in Modelica, rather than importing a compiled solver through the external C function interface, raises a set of language-specific issues that do not arise in procedural environments such as MATLAB or Python. Four were central to this work.

Clocked execution versus continuous integration. The MPC control law is a discrete algorithm, but it is embedded in a model whose plant dynamics are integrated continuously. Executing the predict-correct and QP-solve logic inside an ordinary algorithm section would cause it to run at every internal solver time step, producing many spurious updates per intended sample and corrupting the observer. The controller is therefore wrapped in a `when sample(0, sP)` clause so that the entire discrete cycle fires exactly once per sample period *sP*, independent of the integrator step size. Isolating all discrete state updates within this clause is essential for correct interaction between the discrete controller and the continuous plant.

Event semantics and discrete state. Because the QP solver mutates internal working-set arrays and the Kalman estimate on each sample, these quantities must be declared as discrete state that persists across events and is updated only inside the `when` clause. Care is required to ensure that no branch of the active-set iteration leaves a discrete variable unassigned at an event instant, which the Modelica translator treats as an error. The explicit KKT formulation helped here as well, since every iteration assigns the multiplier vector and primal step deterministically.

State initialization and bumpless transfer. Modelica evaluates initial equations once at $t = 0$, well before the controller is enabled at $t_{\text{enable}} = 50$ s. The setpoints, the previous-input vector u_{prev}, and the Kalman estimate must therefore be initialized from the plant's measured state at the enable instant rather than at $t = 0$, so that the first commanded increment is $\Delta u_0 = 0$ (Sec. 5). Handling this inside the same `when` clause that enables the controller avoids the large spurious first move that otherwise triggers a cascade of constraint activations.

Reliance on the standard-library linear solve. The KKT system is solved with `Modelica.Math.Matrices.solve`. This routine raises a hard "A is singular" error and terminates the simulation when the saddle-point matrix is rank

DOI 10.3384/ECP219 258

deficient, which is precisely the failure mode observed with the S1 solver. The $(2,2)$-block regularization of Sec. 3 guarantees a non-singular system and therefore a well-defined return from the standard-library solve at every sample, which is what allows the simulation to run to completion. A native implementation gains portability across Modelica tools at the cost of depending on the host library's error handling, so guaranteeing a well-conditioned system inside the model is a practical necessity rather than a numerical nicety.

8.3 Sample Time vs. Horizon Trade-Offs

The $T_s = 0.01$ s configuration demonstrated that a native Modelica MPC can achieve sub-percent steady-state tracking accuracy, but at the cost of an effective look-ahead ($NT_s = 0.02$ s) that is negligible relative to the 26 s plant time constants. The $T_s = 1.0$ s configuration explored in Section 6 extends the look-ahead to 2 s with $N = 2$ which is still short, but two orders of magnitude closer to the plant dynamics.

The key trade-off revealed by this comparison is between model fidelity and predictive reach. At $T_s = 0.01$ s, the linearization error per step is extremely small, so even an imperfect B matrix produces accurate one-step predictions; the MPC succeeds by making many small, nearly correct corrections. At $T_s = 1.0$ s, each prediction step spans a larger fraction of the plant's dynamic range, amplifying the effect of model mismatch, particularly the gain and cross-coupling errors that arise when the plant operates far from the identification point.

This finding suggests that the optimal path forward combines a moderate sample period ($T_s = 1$–5 s) with a longer prediction horizon ($N = 5$–10) and, critically, an improved internal model. The author's also note the Modelica gas turbine system model "inertia" may need to be adjusted to ensure physically meaningful changes in operation are appropriate for a large thermal system, where fast responses in load, temperature or blend seen in the model response may not be representative of the full GT system. Candidates include online model adaptation (recursive least squares or extended Kalman filter for the B matrix), gain scheduling based on the current operating point, or a piecewise-linear model that switches A_d/B_d matrices based on measured hydrogen blend and load level. The fixed-gain Kalman filter introduced in Section 6 provides a computationally efficient foundation for these extensions, as it eliminates the per-step matrix inverse and reduces the observer to a single matrix–vector multiply.

9 Conclusions and Future Work

This paper has presented the systematic, data-driven refinement of a native Modelica-based Model Predictive Controller for multi-variable gas turbine operation with hydrogen fuel blending. Starting from the proof-of-concept architecture described in (**Harper 2026**), the refinement campaign addressed both the internal plant model used for prediction and the constrained optimization algorithm used to compute control actions.

The principal findings are:

1. **Explicit KKT with regularization is helpful for constraint-heavy MPC in Modelica.** The reformulated QP solver (S2), employing a full KKT saddle-point system with Hessian regularization ($\varepsilon_H = 10^{-6}$) and $(2,2)$-block regularization ($\varepsilon_{\mathrm{KKT}} = 10^{-8}$), eliminated the numerical singularity that caused simulation failures in the original solver (S1) and provided explicit Lagrange multiplier diagnostics that accelerated constraint-related debugging.

2. **Constraint alignment is as important as the control algorithm.** A mismatch of 0.04 (in normalized coordinates) between the MPC's IGV lower bound and the plant's physical minimum caused integrator windup in the Kalman filter that produced persistent firing temperature tracking failures. This interaction between constraint specification and state estimation could not be predicted from the algorithmic formulation alone and required careful examination of simulation data to diagnose.

3. **Robust multi-variable tracking is achievable.** The best configuration achieves hydrogen blend tracking within 0.70% steady-state error, load tracking within 0.53 MW, and firing temperature offset of 6 K, with no H_2 fuel-control valve saturation and only brief FSR saturation during the initial transient.

4. **Sample-period extension is feasible and informative.** Re-identification of the state-space model at $T_s = 1.0$ s and replacement of the online Kalman covariance propagation with a fixed gain produced a stable closed-loop simulation with physically meaningful prediction-matrix gains. While tracking accuracy is reduced compared with the $T_s = 0.01$ s configuration (H_2 MAE of 0.104 vs. 0.025, temperature MAE of 119 K vs. 71 K), the result confirms that the native Modelica MPC architecture scales to sample periods commensurate with plant dynamics, and identifies model mismatch across the operating envelope as the primary remaining barrier.

Future work includes:

- Extended prediction horizons ($N = 5$–10) at $T_s = 1.0$ s to provide 5–10 s of look-ahead;
- Gain scheduling or online model adaptation to address the model mismatch identified at $T_s = 1.0$ s across the operating envelope;
- Differential Q/R weighting to improve firing temperature tracking speed;
- Testing on additional model configurations and operating scenarios.

DOI **10.3384/ECP219 258**

- Soft output constraints with explicit slack variables and penalty weights (Eqs. (15)–(16)), enabling operation with tighter, physically meaningful output limits without risking QP infeasibility;
- Offset-free tracking through integrating disturbance-model augmentation of the Kalman observer, to eliminate the residual firing-temperature offset;

10 References

References

[1] Jim Harper, Paolo Pezzini, Moritz Hubel, and David R. Noble. Hydrogen Blending and Partial Load Control Modeling: Updated Designs and Simulations. Paper presented at the 12th International Gas Turbine Conference: Advancing Turbomachinery Innovations and Strategies for Net-Zero Pathways, 2025.

[2] Harper, James, Paolo Pezzini, and Moritz Hubel. (2024) "Hydrogen Fueling and GT Demand Changes Impact Insights Through Modeling." In Turbo Expo: Power for Land, Sea, and Air, vol. 87981, p. V006T08A002. American Society of Mechanical Engineers, 2024.

[3] Song, Ge, Konstantin Filonenko, Xiaoqiao Wen, Razgar Ebrahimy, and Natasa Nord. "Modelica-Based Model Predictive Control for a CO_2 Heat Pump System: Case Study in Oslo." Journal of Building Engineering 115 (2025): 114501. https://doi.org/10.1016/j.jobe.2025.114501.

[4] L. Wang, Model predictive control system design and implementation using MATLAB®, Springer Science& Business Media, 2009.

[5] Hägglund, Tore. Process Control in Practice. Berlin/-Boston: Walter de Gruyter GmbH, 2023.

[6] Guzmán, José Luis, and Tore Hägglund. Feedforward Control: Analysis, Design, Tuning Rules, and Implementation. Berlin/Boston: Walter de Gruyter GmbH, 2024.

[7] Hoelemann, S., and D. Abel. "Modelica Predictive Control – An MPC Library for Modelica." at - Automatisierungstechnik, 2009, Vol. 57, pp. 187-194. http://dx.doi.org/10.1524/auto.2009.0766.

[8] Electric Power Research Institute. 2019. Steam Temperature Control for Combined-Cycle Units: Survey and Testing of Advanced Strategies. Palo Alto, CA: EPRI. Report No. 3002016316.

[9] Hoerl, A. E., and Kennard, R. W. (1970). "Ridge Regression: Biased Estimation for Nonorthogonal Problems," *Technometrics*, 12(1), pp. 55–67.

[10] Harper, James, Paolo Pezzini. Daniel Rhode, and Hubertus Tummescheit "Native Modelica Model Predictive Control for Multi-Variable Gas Turbine Hydrogen Blending.", GT2026-176679, ASME Turbo Expo 2026.

[11] Rowen, W. I. (1983). "Simplified Mathematical Representations of Heavy-Duty Gas Turbines." Journal of Engineering for Power, 105(4), pp. 865–869. https://doi.org/10.1115/1.3227494.

DOI **10.3384/ECP219 258**

Quantum Algorithms for Patent Portfolio Analysis Using OpenModelica and IBM Qiskit

Mark R. Myers, J.D., Ph.D.

McCormick, Paulding & Huber PLLC
Hartford, Connecticut, U.S.A.
myers@ip-lawyers.com

Abstract

Near-term quantum computers are hybrid machines: classical computers, microprocessors, and control hardware drive a quantum subsystem that runs algorithms aimed at solving problems whose cost grows super-polynomially with input size. Prototype processors from research groups and commercial vendors are now accessible through software platforms such as IBM Qiskit, which offers both a circuit simulator (Qiskit Aer) and cloud access to physical quantum devices. Designing such classical-quantum hybrid systems can draw on the equation-based modeling and Hardware-in-the-Loop (HIL) techniques implemented in tools like OpenModelica. This paper integrates IBM Qiskit into the Modelica ecosystem through the OpenModelica external C interface and the Qiskit C API. We exercise the resulting solver on a 0/1 knapsack test case drawn from patent portfolio analysis as an example of asset management.

Keywords: Quantum computing, knapsack problem, patent portfolio

1 Paradigm of Quantum Computing

Quantum computing departs from the von Neumann architecture that has defined classical digital computing for three-quarters of a century. Where a classical bit is fixed at 0 or 1, a quantum computer encodes information in a complex vector space and operates on it through superposition, entanglement, and interference. These ideas bear directly on computational complexity, cryptography, and high-performance computing (HPC).

1.1 Foundations and History

The conceptual foundations were laid in the early 1980s. Richard Feynman argued that simulating quantum systems on classical hardware incurs exponential overhead in the size of the state space, which motivated building computers whose substrate is itself quantum (Feynman 1982). Paul Benioff showed that a quantum-mechanical Turing machine was theoretically feasible (Benioff 1980), and David Deutsch described a universal quantum computer able to simulate any system that can be physically realized (Deutsch 1985).

These ideas stayed largely theoretical until 1994, when Peter Shor gave a polynomial-time quantum algorithm for integer factorization and discrete logarithms (Shor 1994). Since RSA and Diffie–Hellman rest on the hardness of factoring, Shor's result showed that a sufficiently large quantum computer could break widely deployed public-key cryptography. Two years later, Lov Grover gave a quantum search algorithm with a quadratic speedup for unstructured search (Grover 1996). Shor's and Grover's algorithms together set the field's expectation of useful quantum speedups.

1.2 Quantum Complexity Theory

Quantum computers do not solve hard problems by evaluating every answer in parallel, as is sometimes supposed. Instead, they evolve a quantum state so that the amplitudes of correct solutions reinforce through constructive interference while incorrect paths cancel.

The class *BQP* (Bounded-Error Quantum Polynomial-Time) captures the decision problems a quantum computer can solve in polynomial time with bounded error. Every problem in *P* lies in BQP, since classical logic can be expressed with reversible quantum gates such as the Toffoli (Nielsen and Chuang 2010). BQP overlaps *NP* but is believed distinct from it (Figure 1). Integer factorization, for example, lies in BQP and in *NP* yet, conjecturally, outside *P*. BQP is not, however, believed to contain the *NP*-complete problems. No pure quantum algorithm is known that solves combinatorial problems such as knapsack, the Traveling Salesman Problem, or SAT in polynomial time. For unstructured instances, Grover's algorithm gives only a quadratic speedup ($\mathcal{O}(2^{n/2})$ versus $\mathcal{O}(2^n)$), which shortens the runtime without changing the complexity class.

1.3 Hybrid HPC Ecosystem

Modern quantum hardware is integrated into traditional HPC environments rather than operated in isolation. Cloud-based Quantum-Computing-as-a-Service (QCaaS) models route bulk data processing and optimization loops to classical CPUs and GPUs, offloading specific non-classical subroutines to Quantum Processing Units (QPUs).

DOI 10.3384/ECP219 276

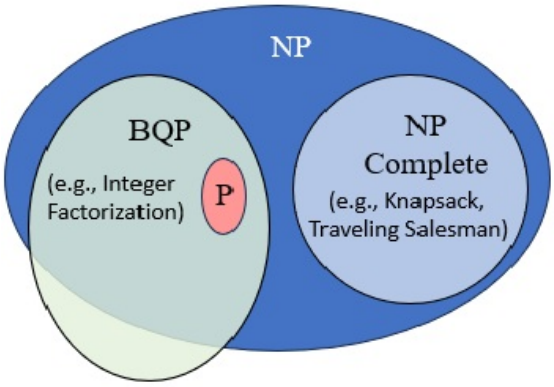


Figure 1. Containment relationships among complexity classes *P*, BQP, *NP*, and *NP*-complete. Adapted from (Sahu and Gupta 2023).

1.4 IBM Quantum Hardware and the Qiskit Software Stack

IBM provides cloud access to its fleet of superconducting quantum processors through Qiskit, an open-source Software Development Kit (SDK) organized as a multi-layered framework that abstracts the underlying hardware (Javadi-Abhari et al. 2024):

Circuit Construction (SDK): Qiskit allows developers to construct quantum circuits by initializing quantum registers and appending gates (Hadamard, CNOT, parameterized rotations).

Transpilation: Because each backend has its own qubit topology and native gate set, the Qiskit Transpiler rewrites abstract circuits to match these constraints, routing qubits via SWAPs (which exchange the states of two qubits) and lowering high-level gates into native instructions.

Execution Primitives: Qiskit exposes execution through two unified primitives. The *Sampler* returns the measurement-bitstring distribution of a circuit, suited to algorithms with discrete outputs (e.g., Grover-class search). The *Estimator* returns expectation values of user-supplied observables, used by variational algorithms such as the Variational Quantum Eigensolver (VQE).

Backends: A single Qiskit interface targets either the `qiskit-aer` simulator (statevector simulation with optional noise injection, limited by classical memory to roughly 30–40 qubits) or a physical IBM processor (e.g., Eagle®, Heron®, Osprey®) accessed over the cloud.

2 The Quantum Knapsack Problem

The Knapsack Problem is a central problem in combinatorial optimization and a clean model of a hard resource constraint: given a weight capacity and a set of items with values and weights, which subset maximizes total value without exceeding the capacity?

Its study, and that of its number-partition antecedents, spans more than a century (Mathews 1896), and the problem has served as a proving ground for the core techniques of combinatorial optimization — dynamic programming, branch-and-bound, relaxations, and approximation schemes (Kellerer et al. 2004) — growing from a logistical puzzle into a working tool for quantitative finance and, more recently, a target for quantum computing.

2.1 Classical Computational Complexity and Feasible Limits

To understand why the Knapsack Problem continues to challenge modern digital supercomputers despite decades of algorithmic optimization, one must examine its exact position within computational complexity theory.

2.1.1 Exponential Search Space

In its standard 0/1 formulation, an instance with n items presents a discrete choice for each item: either it is excluded (0) or included (1). This yields a total combinatorial search space of exactly 2^n possible subsets.

For a small portfolio of 20 assets, a classical computer evaluates $2^{20} \approx 1.05 \times 10^6$ combinations, a trivial task done in milliseconds. Scaling the pool to 100 assets raises the search space to $2^{100} \approx 1.27 \times 10^{30}$ combinations. Even at one trillion per second, an exhaustive search would take over 40 billion years, several times the age of the universe. That figure measures the exhaustive search space, not the practical difficulty of the standard instance: exact methods that sidestep enumeration (dynamic programming, branch and bound, and branch and cut) routinely dispatch 100-item 0/1 knapsack instances in a fraction of a second on commodity hardware (Kellerer et al. 2004).

2.1.2 Pseudo-Polynomial Time

Dynamic-programming algorithms such as Bellman's (Bellman 1957) solve the 0/1 Knapsack Problem in $\mathcal{O}(nW)$ time, where n is the number of items and W is the capacity. This looks like an efficient polynomial-time bound, but the appearance is misleading: the regime is called *pseudo-polynomial*.

The variable W is bounded not by the number of inputs but by the magnitude of a numeric value. In digital computers, a number W is represented in binary using $m = \log_2(W)$ bits. Expressed in terms of the actual length of the input data, the time complexity is therefore strictly exponential:

$$\mathcal{O}(n \cdot 2^m)$$

When capacity or asset values require high precision (currency to the cent, or fractional micro-shares), W grows very large, and classical solvers may run out of memory or time.

DOI 10.3384/ECP219 276

2.1.3 Correlated Instances and Reduction Failure

The practical "feasibility limit" of classical solvers depends on the structure of the data. In real asset allocation, values and their weights (costs or risks) are often strongly correlated: more valuable assets tend to cost more or carry higher risk.

Under strong correlation, standard reduction mechanisms such as bounding functions and the "core-problem" method (reduction to smaller, essential subproblems) break down. Pruning cannot eliminate sub-branches early, and solvers fall back to their worst-case exponential behavior. This is a hard ceiling for classical digital architecture.

2.1.4 Constrained Variants and Quantum Approaches

As classical improvements plateau against these limits, research has shifted toward more constrained variants that defeat standard reduction techniques. The Multi-Dimensional Knapsack Problem (MDKP) requires the chosen set to satisfy several capacity constraints at once, such as weight, volume, and budget. The Quadratic Knapsack Problem (QKP) assigns value not only to individual items but also to synergies between pairs of selected items.

To navigate these solution spaces, researchers have turned to quantum methods, several of which are under active investigation. Grover's search offers a provable quadratic speedup for unstructured search (Grover 1996), reducing the query cost of scanning high-dimensional combinations. The Quantum Approximate Optimization Algorithm (QAOA) is a hybrid method for Noisy Intermediate-Scale Quantum (NISQ) devices (Preskill 2018) that uses a classical outer loop to tune quantum parameters (Farhi et al. 2014). It can be adapted to knapsack constraints by "warm-starting" the circuit from the solution of a continuous relaxation, so the optimizer inherits the relaxation's approximation guarantees (Egger et al. 2021). Quantum annealing instead seeks the lowest-energy state of an Ising (equivalently, Quadratic Unconstrained Binary Optimization, or QUBO) formulation of the problem (Kadowaki and Nishimori 1998; Lucas 2014), the approach used by D-Wave hardware for large portfolio-balancing problems (Venturelli and Kondratyev 2019). Most recently, the Quantum Tree Generator prepares a superposition over a knapsack instance's feasible set that amplitude amplification then drives toward optimal selections (Wilkening, Lefterovici, Binkowski, Perk, et al. 2025), a method since extended to the quadratic and multidimensional variants (Wilkening, Lefterovici, Binkowski, Funck, et al. 2025).

Achieving a true *quantum advantage*, where a quantum computer cleanly outperforms the best classical supercomputers on a practical task, is not yet within reach here. Unlike integer factorization, which Shor's algorithm solves exponentially faster, the Knapsack Problem lacks a clean periodic structure, and current evidence suggests any quantum speedup is likely to be polynomial rather than exponential.

Consequently, near-term applications are expected to rely on hybrid architectures. Rather than replacing classical frameworks, Quantum Processing Units are likely to serve as coprocessors, invoked by classical CPUs to handle the hardest-to-optimize problem instances.

2.1.5 Knapsack Applications in Asset Management

The allocation of a finite capital budget among competing projects is the classic capital-rationing problem (Lorie and Savage 1955), and the knapsack model captures it directly: the "knapsack" is the budget or risk allowance, and the "items" are the candidate investments selected to maximize returns.

While traditional frameworks like Modern Portfolio Theory (MPT) focus heavily on statistical variance and covariance to build a continuous frontier, they often struggle with real-world, discrete operational limits. The knapsack model fills this gap by offering a structural, rule-based approach to explicit integer constraints, cardinality limits, and transaction friction.

This framework appears across corporate finance and wealth management. In capital budgeting, a firm with a fixed budget for new projects faces a binary fund-or-not decision for each one, and modeling it as a knapsack identifies the set of independent projects that maximizes aggregate Net Present Value (NPV) without breaching the ceiling. In asset-backed securitization, institutions pool illiquid assets such as mortgages, auto loans, and lease contracts into marketable securities whose profitability depends on which assets are chosen, a selection naturally posed as an MDKP with constraints on pool size, risk tranches, and yield. Portfolios of patents, trademarks, and other intangibles, increasingly central to corporate valuation, fit the knapsack formulation because acquisition and divestment decisions turn on time-varying cost and appreciation estimates.

2.1.6 Portfolio Binary and Cardinality Constraints

Selecting a basket of assets is rarely as simple as setting abstract percentage weights. Fund managers face strict operational limits. Minimum investment thresholds rule out holding an arbitrarily small fraction of every asset, and the 0/1 knapsack model captures this by forcing a binary include-or-exclude decision for each position. A cardinality cap, such as "at most 30 holdings," adds a linear constraint on the number of positions and contains transaction costs and oversight burden, a device studied widely in portfolio optimization (Chang et al. 2000). Redefining "weight" as an asset's volatility or Value-at-Risk and "value" as its expected return turns the knapsack into a tool for building maximum-yield portfolios under a hard risk budget. When assets are divisible into fractional shares, the problem relaxes to a continuous fractional knapsack, which admits a fast greedy solution and real-time capital weighting within a liquidity envelope.

Real markets demand optimization under heteroge-

DOI 10.3384/ECP219 276

neous constraints such as compliance mandates, transaction costs, and liquidity shifts. Whether solved by classical metaheuristics (such as genetic algorithms or reinforcement learning) or by hybrid quantum methods, the knapsack formulation remains a practical tool for disciplined capital allocation under hard constraints.

3 Patents as Intangible Assets

Managing patent collections as investable portfolios reframes intellectual property (IP) from a purely defensive legal instrument into a revenue-generating asset class. Applying portfolio-theory principles to IP lets executives and institutional investors diversify technology risk, build recurring licensing revenue, and surface latent balance-sheet value. This is a specific instance of a broader question: how the advent of fault-tolerant quantum computing can be expected to affect law and policy. For a survey of that landscape, see Myers (2026).

3.1 The Patent Asset Knapsack Problem

The 0/1 Knapsack Problem models a universal constraint: how to maximize the value of a choice when resources are strictly limited.

Consider a portfolio manager acquiring patents as intangible assets for a company that wants to make, use, and sell quantum processors. From a pool $P = \{p_1, p_2, \ldots, p_n\}$ of candidate patent families, she may acquire families up to a fixed cost budget W. Each item i has a value v_i (its utility) and a cost c_i (the resource cost of acquiring it). The objective is to select a subset whose total cost stays within the budget while the accumulated value of the selected families is as large as possible.

The "0/1" prefix specifies a strict binary constraint on item selection. For each item, the portfolio manager faces a binary choice:

0 (Exclude): Leave the item behind completely.

1 (Include): Acquire the entire legal rights of the patent.

Under this formulation, the legal rights conferred by a patent are treated as indivisible: a portfolio manager may not acquire a fractional share of a patent (e.g., rights limited to a single claim), nor may the same patent be selected more than once.

To express the 0/1 Knapsack Problem formally, we model the choices as a decision vector and set up an optimization framework consisting of an objective function and a capacity constraint:

Given:

- A set of items indexed from 1 to n.
- A vector of item values: $\mathbf{v} = [v_1, v_2, \ldots, v_n]$, where $v_i > 0$.
- A vector of item costs: $\mathbf{c} = [c_1, c_2, \ldots, c_n]$, where $c_i > 0$.
- A maximum cost budget: W.

We define a set of binary decision variables, x_i, for each item i:

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

The objective is to maximize the cumulative value of the chosen items:

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

The total cost of the selected patents must not exceed the acquisition budget. This is enforced by requiring the sum of the selected costs to be no greater than W:

$$\text{subject to } \sum_{i=1}^{n} c_i x_i \leq W \tag{2}$$

The computational complexity of the 0/1 Knapsack Problem is classified as *NP*-hard: there is no known classical algorithm that can find the exact optimal solution in polynomial time ($\mathcal{O}(n^k)$) for all possible instances. Because each of the n items has two choices (0 or 1), a brute-force search must evaluate all 2^n possible subsets, which becomes intractable beyond modest problem sizes.

4 OpenModelica & Qiskit Integration

This work pursues two objectives: to integrate the Qiskit C API as a callable computational engine within OpenModelica, exercised here in simulation (a self-contained statevector simulator, supplemented by an Aer noise study), with the exported OpenQASM 3 circuit description keeping a path open to physical IBM quantum hardware; and to use that integrated environment to demonstrate that an *NP*-hard combinatorial problem can be posed and solved within OpenModelica augmented by a quantum solver. We emphasize that this paper is a proof of concept: it demonstrates the feasibility of integrating Qiskit into the OpenModelica analysis environment. It is *not* an exercise in knapsack-algorithm development, and it makes no claim of quantum advantage, algorithmic or hardware-demonstrated. Those are separate endeavors to be pursued once a properly integrated OpenModelica-Qiskit platform has been established.

We demonstrate integration using a simple patent-portfolio Knapsack Problem as formulated in Section 3.1.

4.1 Quantum Brute-Force Enumeration

Although classical dynamic-programming solutions to the 0/1 Knapsack Problem exist (Section 2), the problem provides a natural vehicle for demonstrating quantum brute-force enumeration. An N-qubit Hadamard layer creates a uniform superposition over *all* 2^N candidate subsets simultaneously, after which classical post-processing on the statevector probabilities identifies the optimal feasible state. Recasting (1)–(2) in this enumeration form, we select a subset $S \subseteq \{0,\ldots,N-1\}$ that maximizes $\sum_{i\in S} v_i$ subject to $\sum_{i\in S} c_i \leq W$ for a budget W.

DOI 10.3384/ECP219 276

4.2 Prototype Implementation

MODKNAPSACK realizes this approach inside OpenModelica, a widely used open-source equation-based modeling tool (Fritzson 2015), by constructing and simulating a quantum brute-force circuit through the Qiskit C API and exposing the solver to OpenModelica through its external C function interface. A C solver (`knapsack.c`) drives the Qiskit C API (`libqiskit.so`) to construct the circuit, then runs a self-contained pure-C statevector simulator to evaluate it. The design spans three technical layers (Modelica, C, and the Qiskit library API) and contributes a correct, portable pattern for passing multiple typed outputs through OpenModelica's external C interface. Source code, together with compilation and linking instructions and a companion document detailing the implemented algorithms and circuits, is available at `https://github.com/Legal-Ingenuity/2026ModelicaConference`.

5 System Architecture

5.1 Layered Overview

Figure 2 shows the three-tier architecture. The *Modelica layer* consists of two entities in the package `ModKnapsack`: the external-function wrapper `ModCallKnapsack` and the simulatable model `KnapsackRun`. The *C layer* contains `callKnapsack` (the OpenModelica entry point) and `solve_knapsack` (the quantum solver proper). The *library layer* is the prebuilt Qiskit shared library `libqiskit.so`, accessed through the Qiskit C API headers.

5.2 Data Flow

At simulation time OpenModelica calls `ModCallKnapsack` with five inputs: the number of items `Npatents` (N), the value and cost arrays `Values`[N] and `Costs`[N], the budget `MaxCost` (W), and the circuit-drawing detail level `QDrawDetail`. On return, two outputs are consumed: the binary inclusion vector `PatentIndices`[N] and the execution status `ExStatus` (0 on success, -1 when no feasible subset exists, -2 on invalid input, and -3 when the instance exceeds the quantum solver's capacity, $N > N_{\max}$). A third output, `ActualCost`, is computed entirely within Modelica by summing `Costs`[i] for every i where `PatentIndices`[i] = 1.

6 The Quantum Circuit

6.1 Circuit Design Rationale

The circuit is an N-qubit *quantum brute-force enumerator*. Its design is deliberately simple: because the post-processor evaluates *every* 2^N basis state exhaustively, there is no advantage to biasing the initial state toward promising subsets. The circuit therefore places every qubit in an equal superposition, making all 2^N computational basis states equiprobable. This is the standard uniform-superposition preparation that also seeds more sophisticated algorithms such as Grover's search (Grover 1996).

For a reader new to quantum circuits, it helps to separate what the circuit does from what the surrounding algorithm does. The circuit's *sole* computational act is the Hadamard layer, whose purpose is to place the N item qubits into a uniform superposition that represents all 2^N candidate subsets at once—item i belongs to the subset encoded by basis state $|x\rangle$ exactly when bit i of x is 1. The barrier that follows is only a drawing separator and performs no operation, and the final measurement returns a *single*, randomly drawn subset into the `sel` register—not the optimum.

Crucially, the optimum is chosen by the *classical* post-processor rather than by the measurement: the pure-C statevector simulator exposes the amplitudes of all 2^N basis states, and an exhaustive scan keeps the highest-value subset whose cost stays within the budget (Section 8). The quantum register therefore *enumerates* the search space while the classical layer *selects* the answer, so this brute-force construction makes no claim of quantum speedup. Because it uses only single-qubit Hadamard gates and no entangling operations, the circuit exercises superposition but not entanglement. Entanglement enters only with the Grover solver whose results are reported in Section 11. A citation-backed walkthrough with flowcharts, written for readers unfamiliar with quantum-circuit notation, accompanies the source in the repository's algorithm companion.

6.2 Circuit Structure

The circuit uses two named registers:

- **item**$[0..N-1]$: quantum register, N qubits, one per item.
- **sel**$[0..N-1]$: classical register, N bits, measurement targets.

Three instruction layers follow in sequence:

1. **Superposition layer.** A Hadamard gate H is applied to every item qubit, transforming the initial state:

$$|0\rangle^{\otimes N} \xrightarrow{H^{\otimes N}} \frac{1}{\sqrt{2^N}} \sum_{x=0}^{2^N-1} |x\rangle. \tag{3}$$

   Basis state $|x\rangle$ encodes the subset in which item i is included iff bit i of x is 1.

2. **Barrier.** A barrier instruction spanning all N qubits acts as a visual separator between the superposition and measurement layers in circuit diagrams. It has no effect on the unitary evolution.

3. **Measurement layer.** Each qubit i is measured into classical bit i. After the H layer, every basis state is measured with equal probability $1/2^N$. The Born-rule outcome (Nielsen and Chuang 2010) is recorded in register **sel**: each basis state is observed with probability equal to its squared amplitude, $P(x) = |\alpha_x|^2$ (Eq. 4).

DOI 10.3384/ECP219 276

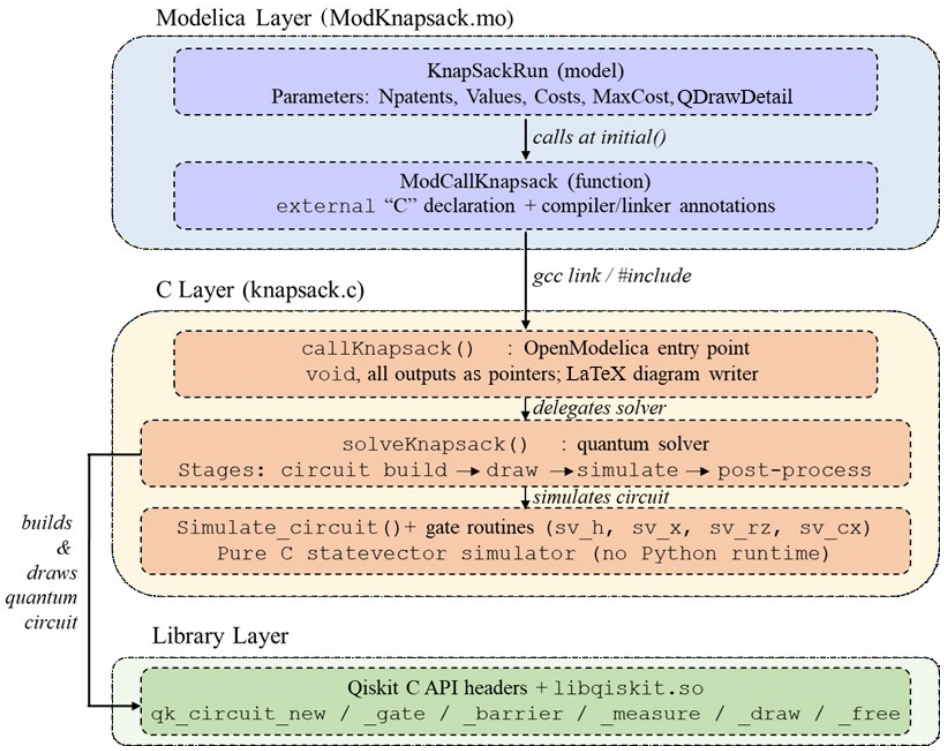


Figure 2. Three-tier architecture of MODKNAPSACK. Control flows top-down, and data flows back up via C pointer arguments.

6.3 Circuit Diagram

Figure 3 shows the circuit for a generic N-item problem, drawn directly in Ti*k*Z for portability. The `write_latex_circuit()` routine in `knapsack.c` emits an equivalent `quantikz` diagram to `printCircuitFile.txt`, one of several outputs selectable through the integer `QDrawDetail` detail level (ranging from no output, through text and schematic renderings, to a gate-level OpenQASM 3 export).

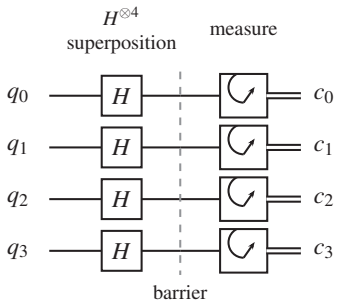


Figure 3. The N=4 quantum brute-force knapsack circuit. A Hadamard layer $H^{\otimes 4}$ places the register in a uniform superposition over all $2^4 = 16$ item-selection subsets (Eq. 3). A barrier (dashed line) separates the superposition layer from the measurement layer. Each qubit $q_i \equiv \texttt{item}[i]$ is measured into classical bit $c_i \equiv \texttt{sel}[i]$. For general N, the circuit extends to N qubit wires with a correspondingly wider barrier.

Table 1 summarizes the circuit parameters.

Table 1. Quantum circuit parameters.

Quantity	Value	Notes
Qubits	N	one per item
Classical bits	N	one per item
Hadamard gates	N	$H^{\otimes N}$ layer
Barriers	1	visual separator
Measurements	N	qubit $i \to$ bit i
Basis states	2^N	all item subsets
Statevector size	$2^N \times 16\,\mathrm{B}$	complex128

7 The Qiskit C API

7.1 API Overview

The Qiskit C API is experimental code made available by IBM under an open-source license bundled in the Qiskit SDK source code.[1] Table 2 lists the API calls used in the prototype `knapsack.c` (Section 8).

7.2 Circuit Construction

Circuit construction in `solve_knapsack()` follows the ownership model of the Qiskit C API, in which each piece of data is "owned" by one specific "owner".

[1] To use the C API, the user must compile the Qiskit SDK source code using Rust, then build the C API separately using a makefile. See `https://quantum.cloud.ibm.com/docs/en/api/qiskit-c`. The underlying implementation is written in Rust, and the C API is a thin bridge over the Rust core.

DOI 10.3384/ECP219 276

Table 2. Qiskit C API functions used in the prototype `knapsack.c`.

Function	Purpose
`qk_circuit_new`	Allocate circuit
`qk_quantum_register_new`	Create q-register
`qk_classical_register_new`	Create c-register
`qk_circuit_add_quantum_register`	Attach q-register
`qk_circuit_add_classical_register`	Attach c-register
`qk_{quantum,classical}_register_free`	Free registers
`qk_circuit_gate`	Append gate
`qk_circuit_barrier`	Append barrier
`qk_circuit_measure`	Append measurement
`qk_circuit_draw`	Text-art drawing
`qk_circuit_num_{qubits,clbits}`	Metadata query
`qk_circuit_num_instructions`	Instruction count
`qk_circuit_instruction_kind`	Instruction type
`qk_circuit_get_instruction`	Read instruction
`qk_circuit_instruction_clear`	Free instruction
`qk_circuit_free`	Deallocate circuit
`qk_param_str / qk_str_free`	Param serialization

The construction shown here is the prototype enumerator's, preserved in the repository alongside the successor solver whose results Section 11 reports. The successor builds its deeper Grover circuit through the same API, sized at creation rather than assembled from register objects. An empty `QkCircuit` is created with `qk_circuit_new(0,0)`, after which named registers are constructed, attached, and immediately released (the circuit retains its own copy):

Listing 1. Qiskit C API circuit construction (abridged).

```
QkCircuit *qc = qk_circuit_new(0, 0);
QkQuantumRegister  *qreg =
    qk_quantum_register_new((uint32_t)N,
        "item");
QkClassicalRegister *creg =
    qk_classical_register_new((uint32_t)N
        , "sel");
qk_circuit_add_quantum_register(qc, qreg)
    ;
qk_circuit_add_classical_register(qc,
    creg);
qk_quantum_register_free(qreg);
qk_classical_register_free(creg);
for (int i = 0; i < N; i++) {
    uint32_t q[1] = {(uint32_t)i};
    qk_circuit_gate(qc, QkGate_H, q, NULL
        );
}
uint32_t all_q[MAX_ITEMS];
for (int i = 0; i < N; i++) all_q[i] = (
    uint32_t)i;
qk_circuit_barrier(qc, all_q, (uint32_t)N
    );
for (int i = 0; i < N; i++)
    qk_circuit_measure(qc, (uint32_t)i, (
        uint32_t)i);
```

8 `knapsack.c` Implementation

The code in this section documents the original brute-force enumerator that is the subject of this paper. The GitHub repository's current `knapsack.c` has since been developed into the realized Grover solver and multi-format circuit exporter (the `QDrawDetail` drawing levels and OpenQASM 3 export). The solver's measured results are reported in Section 11. It therefore differs in structure and in its `solve_knapsack` internals from the listings here, which describe the earlier prototype.

8.1 Module Structure

`knapsack.c` is organized into five sections:

1. **Pure-C statevector simulator** (`sv_alloc`, `sv_h`, `sv_x`, `sv_y`, `sv_rz`, `sv_cx`).
2. **Circuit walker** (`simulate_circuit`).
3. **Quantum knapsack solver** (`solve_knapsack`).
4. **Module to produce a circuit diagram in LaTeX** (`write_latex_circuit`).
5. **OpenModelica entry point** (`callKnapsack`).

All internal functions are declared `static`, confining their linkage to the translation unit. Only `callKnapsack` carries external linkage and is visible to the OpenModelica-generated simulation binary.

8.2 Pure-C Statevector Simulator

Sections 1 and 2 implement a self-contained statevector simulator that requires no Python runtime or simulation back-end. A heap-allocated array of 2^N `double complex` values represents the amplitudes α_x, $x \in \{0,\ldots,2^N-1\}$, initialized to the all-zero state $|0\rangle^{\otimes N}$. Bit i of index x corresponds to qubit i (qubit 0 = LSB).

Each single-qubit gate routine iterates over 2^{N-1} amplitude pairs (i, j) where $j = i\,|\,(1 \ll t)$ for target qubit t, applying the gate matrix to both amplitudes in place. The supported gate set ($H, X, Y, Z, S, T, R_Z(\theta)$, Phase, CX) is more general than the brute-force circuit requires—which emits only the Hadamard layer—keeping the simulator reusable for richer circuits.

`simulate_circuit()` retrieves each instruction from the `QkCircuit` object via `qk_circuit_get_instruction()`, dispatching to the appropriate gate routine. Barrier, measurement, reset, and delay instructions are detected by `qk_circuit_instruction_kind()` and silently skipped. Probabilities are derived from the final amplitudes using the Born rule:

$$P(x) = |\alpha_x|^2 = \mathrm{Re}(\alpha_x)^2 + \mathrm{Im}(\alpha_x)^2. \tag{4}$$

After the $H^{\otimes N}$ layer all 2^N amplitudes equal $1/\sqrt{2^N}$, yielding $P(x) = 1/2^N$ uniformly.

8.3 Quantum Knapsack Post-Processor

With $P(x) = 1/2^N$ for all x, the measurement is a uniform random draw over all subsets. Rather than sampling a single outcome, the prototype's `solve_knapsack()` performs a deterministic classical *exhaustive search* over all 2^N basis states:

DOI 10.3384/ECP219 276

$$x^* = \arg\max_x \sum_{i=0}^{N-1} v_i \cdot b_i(x) \quad \text{s.t.} \quad \sum_{i=0}^{N-1} c_i \cdot b_i(x) \le W, \quad (5)$$

where $b_i(x) = (x \gg i) \,\&\, 1 \in \{0,1\}$ is the i-th bit of x. The optimal basis state x^* is decoded into a binary indicator vector such that bit i of x^* yields `PatentIndices`$[i] \in \{0,1\}$. In the GitHub repository's current `knapsack.c`, this exhaustive search is retained only as an independent ground-truth check. The subset selection itself is performed by the realized Grover / Dürr–Høyer search (Dürr and Høyer 1996), whose results are reported in Section 11.

Single-inclusion guarantee. The bitmask encoding assigns each item to a unique bit position. Because a bit can be either 0 or 1 but is not set twice, no item can appear more than once in the selected subset. This structural property is made explicit by a `seen[]` guard in `solve_knapsack()`, which records each decoded index and permits at most one inclusion per item.

The total cost of the selected subset is computed in Modelica (see Section 9) to avoid the calling-convention issue that arises when C return values and scalar pointer outputs are mixed in OpenModelica's external C interface.

9 The OpenModelica C Interface

9.1 Package and Function Structure

`ModKnapsack.mo` defines a Modelica `package` with two members: the external-function wrapper `ModCallKnapsack` and the simulatable model `KnapsackRun`.

Listing 2. KnapsackRun algorithm section (abridged).

```
when initial() then
  (PatentIndices, ExStatus) :=
    ModCallKnapsack(Npatents, Values, Costs,
                    MaxCost, QDrawDetail);
  ActualCost := 0.0;
  for i in 1:Npatents loop
    if PatentIndices[i] == 1 then
      ActualCost := ActualCost + Costs[i];
    end if;
  end for;
  // ... print inputs and outputs ...
end when;
```

Binding the call to the initialization event is deliberate: the quantum solver acts here as an initialization-time coprocessor, the same pattern by which a deployed model would consult a QPU before or between transients, and the Hardware-in-the-Loop direction of Section 12 extends exactly this call into the running simulation.

9.2 External Function Declaration

`ModCallKnapsack` is declared as a Modelica `function` with an `external "C"` clause that provides the explicit argument list:

Listing 3. ModCallKnapsack external declaration.

```
function ModCallKnapsack
  input  Integer Npatents;
  input  Real    Values[Npatents];
  input  Real    Costs[Npatents];
  input  Real    MaxCost;
  input  Integer QDrawDetail;
  output Integer PatentIndices[Npatents];
  output Integer ExStatus;
  external "C"
    callKnapsack(Npatents, Values, Costs,
        MaxCost,
                 PatentIndices, ExStatus,
                     QDrawDetail)
  annotation( Include = ...,
              IncludeDirectory = ...,
              Library = ...,
              LibraryDirectory = ... );
end ModCallKnapsack;
```

10 Compiler & Linker Specifications

10.1 Annotation Mechanism

OpenModelica resolves build dependencies for external C functions through four standard Modelica annotations attached to the `external "C"` clause. Table 3 lists the annotations used in `ModCallKnapsack` and the corresponding `gcc` flags they generate.

Table 3. Compiler/linker annotations and the `gcc` flags they produce.

Annotation	Example value	Effect
`Include`	`#include ".../knapsack.c"`	Embeds source in compilation unit
`IncludeDirectory`	`.../qiskit/dist/c/include`	Adds `-I` flag for Qiskit headers
`Library`	`{"qiskit","m"}`	Adds `-lqiskit -lm`
`LibraryDirectory`	`.../qiskit/dist/c/lib`	Adds `-L` flag for `libqiskit.so`

10.2 Source-Inclusion Strategy

Using the `Include` annotation to embed a `#include` directive for a full `.c` source file, rather than a pre-built library, is an intentional architectural choice with the following advantages:

1. **No separate build step.** OpenModelica's own `gcc` invocation handles the entire compilation at simulation time, so no Makefile or CMake is required.
2. **Single translation unit.** All `static` functions in `knapsack.c` are compiled with the generated simulation code, allowing the compiler to inline gate routines aggressively and simplifying symbol resolution.
3. **No symbol conflicts.** Only `callKnapsack` carries external linkage, while all helpers use `static` linkage and do not pollute the global symbol namespace.

The principal trade-off is recompilation cost: each `simulate()` call re-compiles `knapsack.c`. For production use, compiling `knapsack.c`

DOI 10.3384/ECP219 276

into a shared library and switching to the `Library`/`LibraryDirectory` pattern would amortize this cost.

10.3 Runtime Library Discovery

The `LibraryDirectory` annotation generates a `-L` linker flag that resolves the location of `libqiskit.so` at *link time*. At *run time*, the Linux dynamic loader searches for `libqiskit.so` through the standard mechanism: `LD_LIBRARY_PATH`, `/etc/ld.so.cache`, and system default paths. Since `libqiskit.so` is installed in a non-standard project directory, the user must extend the search path before simulation:

Listing 4. Runtime library path configuration.

```
export LD_LIBRARY_PATH=\
  <project_path>/qiskit/dist/c/lib:
      $LD_LIBRARY_PATH
```

Alternatively, an `RPATH`/`RUNPATH` entry could be embedded in the executable at link time via a `-Wl,-rpath` linker flag, reducing reliance on `LD_LIBRARY_PATH`. (We use the `LD_LIBRARY_PATH` approach here.)

11 Preliminary Results

The integration ships with a default four-item test case ($\mathbf{v} = [3,4,5,7]$, $\mathbf{c} = [2,3,4,5]$, budget $W = 6$) that a new user can run to confirm a working installation. With `libqiskit.so` on the library path, simulating the model from the project root with `omc run_knapsack.mos` constructs and evaluates the quantum circuit and returns `PatentIndices` $= [1,0,1,0]$ (items 0 and 2 selected) and `ExStatus` $= 0$, with `ActualCost` $= 6.0$ computed in Modelica. The selected subset attains the optimal total value of 8.0 within the budget, matching the classical optimum and confirming that the OpenModelica $\rightarrow$ C $\rightarrow$ Qiskit toolchain is wired correctly end-to-end.

The brute-force Hadamard circuit scales as 2^N, which confines it to small instances (roughly 20–25 items before classical memory is exhausted). Building on this integration, we have also prototyped a structured replacement: a Grover-based amplitude-amplification solver that performs the subset selection *in-circuit* (Figure 4), pairing a reversible cost/value oracle with quantum maximum-finding (Dürr and Høyer 1996; Grover 1996; Nielsen and Chuang 2010). It reproduces the optimal selection across the validation test cases, with the classical exhaustive search kept only as an independent ground-truth check and no claim of quantum advantage. On the four-item test case the realized solver recovers the optimum with 59 oracle calls over a best-of-three schedule (measured, deterministic seed) against $2^4 = 16$ classical evaluations for the exhaustive scan — at this scale the quadratic envelope has, as expected, not yet paid for itself.

To probe robustness beyond the noiseless Aer baseline, we used the OpenQASM 3 export path to sample the structured circuit under an Aer depolarizing noise model: the probability of recovering the optimal subset degrades smoothly as the two-qubit gate error rate rises, and the noise-tolerance threshold scales *inversely* with the number of entangling gates in the oracle, so deeper instances demand proportionally lower error rates.

Reading Figure 4 from left to right, the oracle U_f encodes the solver's selection criterion as six reversible steps on the ancilla registers: (1) accumulate the subset's total cost $\sum_i c_i x_i$ into the `cost` register and (2) its total value $\sum_i v_i x_i$ into the `value` register; (3) test the cost against the budget ($\leq W$) and (4) the value against a running threshold ($\geq T$), setting the feasibility and value flags; (5) *phase-mark* the subsets that are simultaneously feasible and high-value; and (6) *uncompute* the arithmetic so every ancilla returns to $|0\rangle$ and only the phase mark remains. The diffuser D then amplifies the marked amplitudes. One oracle–diffuser pass is a single Grover iteration, repeated under the BBHT schedule (Boyer et al. 1998) and wrapped in Dürr–Høyer maximum-finding (Dürr and Høyer 1996) to ratchet the threshold T toward the optimum. The repository's algorithm companion steps through this pipeline with flowcharts.

These results bound the regime in which the present integration operates, consistent with the scope stated in Section 4. The demonstration establishes that the problem can be posed and solved through the integrated toolchain. Algorithmic scaling to realistic portfolio sizes is deliberately deferred.

12 Conclusion

12.1 Contributions

ModKnapsack demonstrates a complete vertical integration from an equation-based Modelica model, through OpenModelica's external C interface, to a C solver that constructs and inspects a quantum circuit via the Qiskit C API. To the author's knowledge, this is the first published integration of the Qiskit C API into a Modelica platform. The key architectural contributions are:

- A concrete, runnable demonstration of the OpenModelica `Include` / `IncludeDirectory` / `Library` / `LibraryDirectory` annotation quadruple applied to the Qiskit C API. Because that API is a C ABI (Application Binary Interface) bridge over a Rust implementation, the demonstration establishes a build-and-link pattern by which a Rust-backed quantum-software stack can be invoked from OpenModelica.
- A portable calling-convention pattern for OpenModelica external C functions with multiple typed outputs including the `void`-return / all-pointer pattern, with array and scalar outputs both passed by pointer, as demonstrated by `callKnapsack`.
- A self-contained pure-C statevector simulator that evaluates Qiskit circuit objects without requiring a

DOI 10.3384/ECP219 276

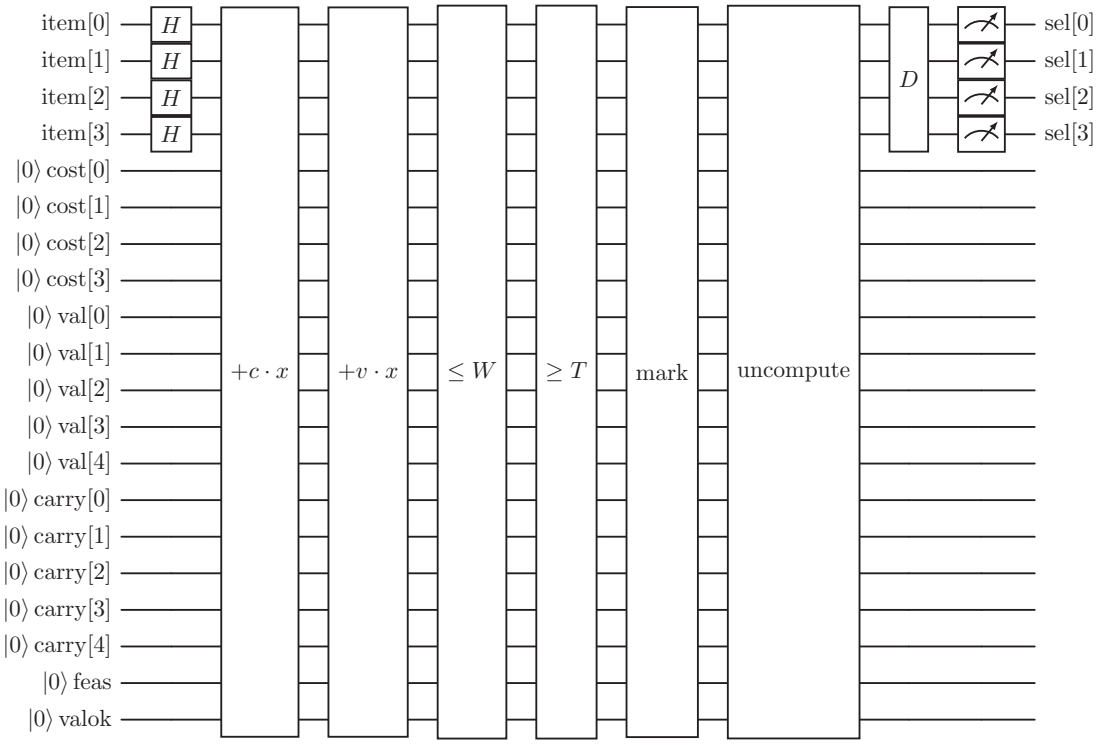


Figure 4. Register-level schematic of the realized Grover knapsack solver for the N=4 test case, emitted by the `QDrawDetail` circuit exporter in `knapsack.c`. The oracle–diffuser pipeline is walked through in the text. Only the `item` register is measured (into `sel`). The cost, value, and carry ancillae are $|0\rangle$-initialized and restored.

Python runtime, enabling the integration to run in any environment where `libqiskit.so` is available.

- A worked end-to-end test case, the patent-portfolio 0/1 Knapsack Problem, demonstrating that an *NP*-hard combinatorial optimization problem can be posed in Modelica syntax and solved through an embedded quantum-circuit subsystem — transparent enumeration with classical selection in the prototype presented here, and full in-circuit Grover selection in the repository's successor solver.

12.2 Future Work

Several directions remain open. Because the present study aimed to establish the viability of the integration between OpenModelica and Qiskit through their respective C-language APIs, it used only small 0/1 knapsack cases chosen for code validation. The most immediate priority is now to exercise the method on realistic patent portfolios from the quantum-computing space, drawing on intellectual-property sources such as the official USPTO repository.

The principal open challenge is scaling the structured solver of Section 11, or alternatives such as QAOA (Farhi et al. 2014), to industrially relevant portfolio sizes.

Execution on real IBM quantum hardware through the Qiskit C API has not yet been demonstrated. The solver keeps that path open by isolating circuit execution behind a single measurement "seam" and exporting any constructed circuit to OpenQASM 3, so the same portable description can be dispatched to the Aer simulator or, later, to a physical IBM QPU backend (e.g., Heron®) through `qiskit-ibm-runtime`, with shot-based sampling replacing the statevector post-processor. Validating that interface on hardware comes next.

On the modeling side, the patent-portfolio formulation currently treats patent values as static scalars. Richer variants, such as time-varying valuations, citation-network synergies (the Quadratic Knapsack variant), or multi-dimensional budget constraints (MDKP), would bring it closer to the realities of commercial IP management.

The OpenModelica integration also invites Hardware-in-the-Loop (HIL) workflows, in which a quantum co-processor is invoked asynchronously during a mechatronic simulation, opening possibilities for quantum-accelerated real-time optimization in cyberphysical systems. Because the correct operation of a hardware quantum circuit can depend sensitively on QPU temperature, such workflows even admit system-level control design in which the cryogenic subsystem itself becomes the hardware in the loop.

The `Include`-based single-translation-unit strategy is

DOI **10.3384/ECP219 276**

convenient for prototyping but recompiles `knapsack.c` at every `simulate()` call. Migrating it to a pre-built shared library and benchmarking the compilation overhead would quantify that cost and ease the path to production deployment.

Incorporating device-calibrated noise models beyond the depolarizing study of Section 11, and folding the resulting stochasticity into the HIL analyses noted above, is a further important direction.

Acknowledgment

The author would like to thank the law firm, *McCormick, Paulding & Huber PLLC*, for their support.

During the preparation of this work the author used Anthropic's Claude (LLM chosen appropriate to task and cost, via Claude Code) as a development and editing assistant: to help resolve build and integration issues with the experimental Qiskit C API, to generate and debug test-harness scripts, and to assist with drafting and editing portions of the manuscript and its LaTeX representation. All algorithmic design, code, mathematical formulations, and text were directed, independently reviewed, and verified by the author, who retains sole intellectual property ownership of, and full responsibility for, the final work.

References

Bellman, Richard (1957). *Dynamic Programming*. Princeton, NJ: Princeton University Press.

Benioff, Paul (1980). "The Computer as a Physical System: A Microscopic Quantum Mechanical Hamiltonian Model of Computers as Represented by Turing Machines". In: *Journal of Statistical Physics* 22.5, pp. 563–591. DOI: 10.1007/BF01011339.

Boyer, Michel, Gilles Brassard, Peter Høyer, and Alain Tapp (1998). "Tight Bounds on Quantum Searching". In: *Fortschritte der Physik* 46.4–5, pp. 493–506.

Chang, T.-J., N. Meade, J. E. Beasley, and Y. M. Sharaiha (2000). "Heuristics for Cardinality Constrained Portfolio Optimisation". In: *Computers & Operations Research* 27.13, pp. 1271–1302. DOI: 10.1016/S0305-0548(99)00074-X.

Deutsch, David (1985). "Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer". In: *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences* 400.1818, pp. 97–117. DOI: 10.1098/rspa.1985.0070.

Dürr, Christoph and Peter Høyer (1996). "A Quantum Algorithm for Finding the Minimum". In: *arXiv preprint arXiv:quant-ph/9607014*. DOI: 10.48550/arXiv.quant-ph/9607014.

Egger, Daniel J., Jakub Mareček, and Stefan Woerner (2021). "Warm-Starting Quantum Optimization". In: *Quantum* 5, p. 479. DOI: 10.22331/q-2021-06-17-479.

Farhi, Edward, Jeffrey Goldstone, and Sam Gutmann (2014). "A Quantum Approximate Optimization Algorithm". In: *arXiv preprint arXiv:1411.4028*. DOI: 10.48550/arXiv.1411.4028.

Feynman, Richard P. (1982). "Simulating Physics with Computers". In: *International Journal of Theoretical Physics* 21.6–7, pp. 467–488. DOI: 10.1007/BF02650179.

Fritzson, Peter (2015). *Principles of Object-Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*. 2nd. Wiley-IEEE Press. ISBN: 978-1-118-85912-4.

Grover, Lov K. (1996). "A Fast Quantum Mechanical Algorithm for Database Search". In: *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, pp. 212–219. DOI: 10.1145/237814.237866.

Javadi-Abhari, Ali et al. (2024). "Quantum Computing with Qiskit". In: *arXiv preprint arXiv:2405.08810*. DOI: 10.48550/arXiv.2405.08810.

Kadowaki, Tadashi and Hidetoshi Nishimori (1998). "Quantum Annealing in the Transverse Ising Model". In: *Physical Review E* 58.5, pp. 5355–5363. DOI: 10.1103/PhysRevE.58.5355.

Kellerer, Hans, Ulrich Pferschy, and David Pisinger (2004). *Knapsack Problems*. Berlin, Heidelberg: Springer. DOI: 10.1007/978-3-540-24777-7.

Lorie, James H. and Leonard J. Savage (1955). "Three Problems in Rationing Capital". In: *The Journal of Business* 28.4, pp. 229–239. DOI: 10.1086/294081.

Lucas, Andrew (2014). "Ising Formulations of Many NP Problems". In: *Frontiers in Physics* 2, p. 5. DOI: 10.3389/fphy.2014.00005.

Mathews, G. B. (1896). "On the Partition of Numbers". In: *Proceedings of the London Mathematical Society* s1-28.1. Read 13 May 1897, pp. 486–490. DOI: 10.1112/plms/s1-28.1.486.

Myers, Mark R. (2026). *A Survey of Quantum Computing for the Working Attorney at the Dawn of the Fault-Tolerant Era*. Preprint. Zenodo. DOI: 10.5281/zenodo.22031924.

Nielsen, Michael A. and Isaac L. Chuang (2010). *Quantum Computation and Quantum Information*. 10th Anniversary. Cambridge: Cambridge University Press.

Preskill, John (2018). "Quantum Computing in the NISQ Era and Beyond". In: *Quantum* 2, p. 79. DOI: 10.22331/q-2018-08-06-79.

Sahu, Himanshu and Hari Prabhat Gupta (2023). "Quantum Computing Toolkit: From Nuts and Bolts to Sack of Tools". In: *arXiv preprint arXiv:2302.08884*. DOI: 10.48550/arXiv.2302.08884.

Shor, Peter W. (1994). "Algorithms for Quantum Computation: Discrete Logarithms and Factoring". In: *Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society, pp. 124–134. DOI: 10.1109/SFCS.1994.365700.

Venturelli, Davide and Alexei Kondratyev (2019). "Reverse Quantum Annealing Approach to Portfolio Optimization Problems". In: *Quantum Machine Intelligence* 1, pp. 17–30. DOI: 10.1007/s42484-019-00001-w.

Wilkening, Sören, Andreea-Iulia Lefterovici, Lennart Binkowski, Marlene Funck, et al. (2025). "A Quantum Search Method for Quadratic and Multidimensional Knapsack Problems". In: *arXiv preprint arXiv:2503.22325*. DOI: 10.48550/arXiv.2503.22325.

Wilkening, Sören, Andreea-Iulia Lefterovici, Lennart Binkowski, Michael Perk, Sándor P. Fekete, and Tobias J. Osborne (2025). "A Quantum Algorithm for Solving 0-1 Knapsack Problems". In: *npj Quantum Information* 11.1, p. 146. DOI: 10.1038/s41534-025-01097-8.

Georgia Institute of Technology
Modelica
Language