Enhanced Steady-State in Modelon Jet Propulsion Library, an Enabler for Industrial Design Workflows

Clément Coïc[1] Moritz Hübel[1] Matthis Thorade[1]

[1]Modelon Deutschland GmbH, Germany,
{clement.coic,moritz.hubel,matthis.thorade}@modelon.com

Abstract

This paper communicates on the implementation of Physics-based Solving in the Modelon Jet Propulsion library, driven by requirements from industrial jet engine design workflows. On- and off-design simulation modes are typically sequential and iterative steps in a model-based design process of jet engines. The solution Modelon provides – based on the Jet Propulsion Library, Optimica Compiler Toolkit, FMI Toolbox and pyFMI – enables performing a robust design of a gas turbine for a design point satisfying relevant constraints of typical off-design scenarios. This paper illustrates this workflow with component and system level examples.

Keywords: Jet Propulsion, Physics-based Solving, Steady-State, On-Design, Off-Design, Optimization

1 Introduction

The sizing of a gas turbine is typically performed over a set of different scenarios. The design point would be the first step in identifying basic parameters as it would be the most constraining scenario for most components. Computing the component parameters from the boundary conditions on this point will be qualified as on-design simulation. For a jet engine, this would typically be the cruise mode at the top of climb. Other scenarios will be run to validate the design for conditions that are relevant for the expected operation such as takeoff or landing. In addition to the validations, iterative tuning of variables (e.g., cooling flow fraction) can be included – these would be named off-design simulations.

Running these scenarios in a disconnected fashion would be tedious and error prone. Model Based System Engineering applied to the design of gas turbines provides a relevant workflow that is addressed in this paper and serves as source of requirements for augmenting the Jet Propulsion library with additional features. Section 2 of this paper discusses such workflows, the resulting requirements and the associated implementation within Modelon Jet Propulsion Library. Section 3 provides some key benefits of using Modelica language and Modelon Optimica Compiler Toolkit to address this problem. Section 4 presents two component examples and discusses their specific on- and off-design behaviors. Section 5 illustrates all these topics within a system level example: the cycle model design of a jet engine.

2 Discussion on Model-Based System Engineering

2.1 Simulation Modes in a MBSE Design Cycle

In an industrial Model-Based System Engineering (MBSE) development, the customer needs, system goal, and purpose would define the system requirements (R from the RFLP acronym). The system would be split into several subsystems – most likely through a function breakdown structure process (F from the RFLP acronym) – and requirements would be propagated and incremented, with traceability and rationale, to the subsystems. This step would be reproduced as many times as necessary to reach a level of subsystems that can be assigned to a given company department – typically organized by physical domains or main product functions.

Based on a requirement specification and the functions and scenarios it shall fulfill – assuming correctness, completeness and consistency –a subsystem can be designed. In a model-based design approach, lumped parameter modeling and simulation (L for logical from the RFLP) can be used to define a viewpoint of the subsystem architecture, select component technologies (assisting trade-off studies) and size each of them. At this stage, it is relevant to highlight that some efforts exist in verifying that models satisfy the requirements using the Modelica language (see for example [OTT15] or [BOU18]). Finally, the detail design would be performed using 3D drawings, Finite Element Analysis, Computational Fluid Dynamics, etc. – this would be the P for physical from the RFLP.

One of the main added values of using model-based development is the possibility to run virtual tests during the design cycle, prior to manufacturing any physical prototype. This enables iterating on the design in a cost- and time-effective manner.

This statement enables deriving two different simulation modes, based on the different steps of the design process:

1. On-design: the boundary conditions (design point scenarios) are known and the aim of the simulation is to compute the parameters of the system (sizing).
2. Off-design: the parameters are all known at this stage and the system and experiments can be run based on some boundary conditions. The outcomes will be the system behavior.

It is quite intuitive based on the above description to see these two modes as sequential and probably iterative. A user would first run the on-design simulation on a design point and then run the simulation and validate his design. The iteration process would come from the fact that the design point might not be unique – e.g., it might be different for each component or subsystem of the system – and thus convergence of the system design can only be achieved by iterating on the on-design simulations.

Figure 1 presents the double-V-model from VDI 2206 [VDI04] standard, displaying the RFLP steps, on which the on- (1) and off-design (2) simulations and potential design iterations (3) have been drawn onto in orange.

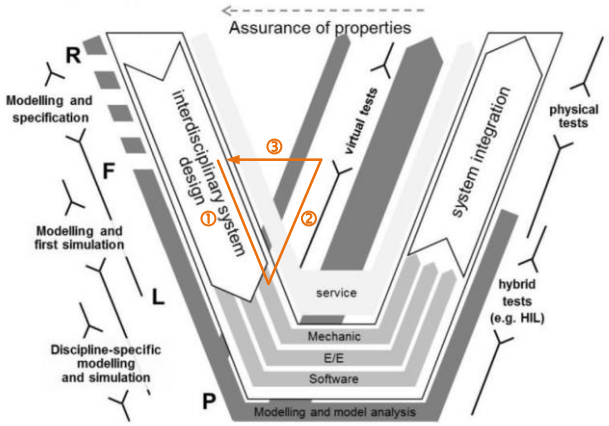

Figure 1. Simulation modes on a design cycle, based on [EIG12]

2.2 Requirements for models

The notion of system and subsystem is a question of perspective. When changing viewpoint, one system might become a subsystem and vice-versa – e.g., when sizing an aircraft, the jet engine is a subsystem; however, when sizing the jet engine, the engine is the system while the turbine might be a subsystem. Models in a MBSE development are a "key tool" for the design. For a model developer, the model shall be considered as a system. A good practice is thus to have requirements for the model development, in the same way that we have requirements driving the physical system itself.

The authors already contributed in:

- adding requirements when it comes to model development [COI16], quoting:
  - R1 Realism. The model architecture shall enable incremental modelling to progressively increase realism regarding the key physical effects that impact performance.
  - R2 Genericity. As far as possible, the model shall be made of a combination of generic sub-models that can be re-used for other modelling purposes.
  - R3 Interfacing. The model shall have standard interfaces that are conserved throughout modelling levels in order to ensure models' replaceability.
  - R4 Balancing. The model shall be balanced (mechanically, energetically, etc.).
  - R5 Ageing and faults. The model shall enable ageing effects or faults to be simulated.
  - R6 Causality. The model shall be developed to admit various causalities, including for inverse simulation.
- developing model architecture that fits the main functions to fulfill [COI18].

It is relevant here to note that the Modelica language enables developing models that fulfill all these requirements. The Modelon Jet Propulsion library is a great example of that. In [SIE17], Sielemann et al. introduced this library dedicated to modeling and simulation of jet engines. This paper shows how the model architecting enables selecting different levels of realism of the subsystems (R1), based on the use of generic models (R2) and interfaces (R3) – see Figure 2. Balancing (R4) is ensured in the library in Modelon implementation of physical laws and following Modelica specification for model numerical balancing [OLS08]). The a-causality of Modelica enables satisfying the last requirement (R6). While ageing and faults (R5) are perfectly feasible with Modelica language, this topic is out of the scope of this library and communication, for now.

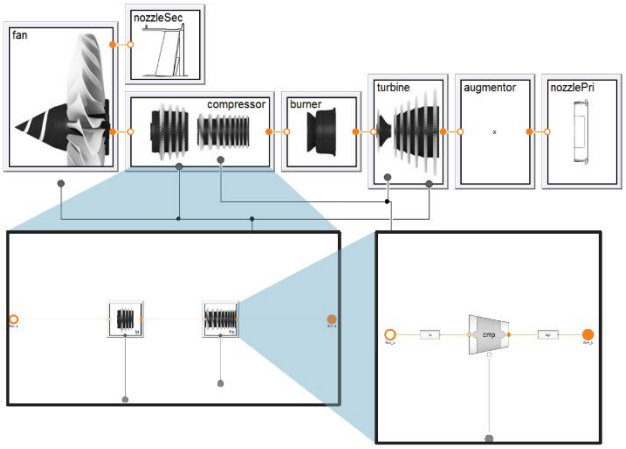

Figure 2. Top-level turbofan model breakdown shown on the top, compressor break-down on the lower left, high pressure compression section break-down on the lower right – from [SIE17]

2.3 Additional requirements for the simulation modes in Steady-State

Unfortunately, there is still a lack of standards when it comes to requirements for model development. In this paper, the authors would like to contribute on this topic by adding model requirements to fit the workflows of model-based design and more specifically for the simulation modes discussed in §1.

Typically, the technical requirements needed to follow the design workflow are:

- In terms of processing:
  - R7. The user shall be able to switch between on- and off-design without changing model structure or realism.
  - R8. The user shall be able to re-use the outputs from the on-design simulation as input to the off-design simulations.
  - R9. The user shall access the model execution through convenient user interface and scripting tools.
- In terms of user friendliness:
  - R10. Switching between simulation modes should not create execution overhead.
  - R11. Switching between simulation modes should be performed by changing a single parameter.
  - R12. The model shall have a robust steady-state embedded capability.

2.4 Implementation of Steady-State simulation modes into Modelon Jet Propulsion Library

To meet these requirements, Modelon developed a vendor-specific language construct, supported by the Modelon Optimica Compiler Toolkit (OCT), to support Physics-based Solving of systems. The Physics-based Solving implemented in our libraries relies on Modelon insights about the physical properties of components and systems to achieve a structure of the system of equations that yields vastly superior numerical properties as compared to traditional tearing algorithms.

Tearing is a symbolic substitution technique well-established in general system simulation. For an introduction in the context of Modelica, see [ELM94]. In most Modelica tools, the selection is fully automatic and based on heuristics. These heuristics are based on the structure of the equation system and code implementation, but not on physical insight. This typically works well for nonlinear algebraic equation systems with a small to medium size of iteration variables per block. However, when using automatic tearing, iteration variables can change unexpectedly for the user with small model changes (e.g., adding more components to the system, changing the type of a model, or with a structural parameter change). Then, hand-tuned start values are lost. Additionally, automatic tearing may choose iteration variables for which the user has less intuition concerning suitable start values or bounds based on engineering insight.

With the Physics-based Solving, the above challenges were solved, and the authors were able to implement this engineering in the library. Instructions embedded in component models guide the compiler and solver on which variables and equations should be selected respectively as iteration variables and residuals for the steady-state simulation. The Physics-based Solving enables a robust steady-state simulation (R12). This language construct enables changing the iteration variables and residuals based on Boolean parameters, without the need for recompilation. The information is stored in an object-oriented fashion, such that modelers can assemble systems graphically, and the desired solving can be deduced from the model topology (model instances and connections). In the discussed application, changing the simulation mode parameter from on- to off-design would change the set of iteration variables and residuals the solver would use. This feature was used to meet requirements R7, R10 and R11.

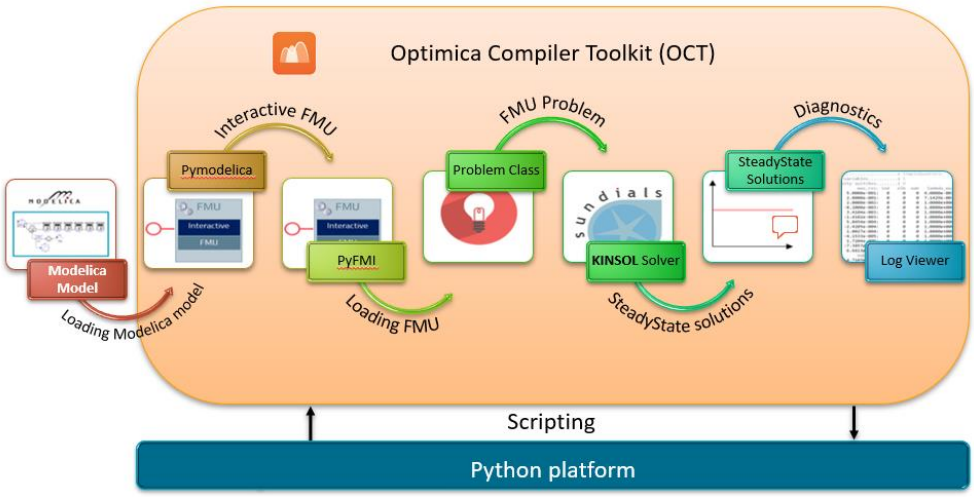

Figure 3. Typical workflow using OCT with Python interface

Generally accepted choices for iteration variables and residual equations are documented in many textbooks and scientific papers from the gas turbine community (see Walsh and Fletcher [WAL04], Oates [OAT97], for instance). They are commonly referred to as "thermodynamic matching" by this community. All that was required for this work was encoding the information in the Modelica language through the above-mentioned language construct, and then testing and validating physical and numerical behavior.

The Modelica compiler OCT generates Functional Mock-up Units (FMUs) – following the FMI standard. Using pyFMI or the FMI Toolbox from Modelon, it is easy to interact with the FMU, and thus requirements R8 and R9 are both met using Python or Matlab scripting into Jupyter notebooks. Figure 3 shows the typical workflow using OCT with Python interface for steady-state simulation.

3 Steady-state simulation using OCT – a robust solution

3.1 Modelica Compiler capabilities

The Modelica language, being equation-based, enables the compiler to have more degrees of flexibility. All Modelica compilers are able to rearrange the acausal equations into causal algorithms, suitable for simulation. Modelica compilers are also usually able to automatically derive the sensitivities relevant for solving optimization problems.

While these features are generic to most Modelica compilers, Modelon compiler (OCT) has the additional capability to hide residual equations and iteration variables from the solver in a compiled model. This feature is key to enable switching between simulation modes without recompiling the model.

3.2 Solver Improvements

Modelica models and FMUs usually use variables expressed in SI units. Variable values may therefore differ by several orders in magnitude [SIE12]. This can yield to ill-conditioned systems which are difficult to converge. In order to handle ill-conditioned systems, the solver supports scaling of both the residual equations and of the iteration variables, which is used in the criteria for a successful solution. Note that while for the Newton method the step is scaling-independent (i.e., direction and length do not change with scaling), for the steepest descent search direction as well as solver termination criteria, results are dependent on the scaling factors applied.

Iteration variables are normally scaled based on the nominal values provided by the modeler. This is done using the nominal attribute x_{nom_j}, according to $D_x^j = 1/|x_{nom_j}|$. If the nominal attribute for a particular iteration variable is not set, then the corresponding scaling factor D_x^j is set to one.

Residual scaling factors utilized by the solver are evaluated based on the Jacobian and follow ideas similar to Jacobian equilibration. The automatic scaling is chosen as $D_x^j = 1/\left|\left|\bar{J}^k(x_0)\right|\right|_\infty$, where $\bar{J}$ is a scaled Jacobian calculated as $\bar{J}(x) = \nabla_x F(x) D_x^{-1}$ and where x_0 is the initial guess.

OCT steady-state solver relies on a combination of residual and step norm as exit criterion. Newton step norm-based criterion is typically used as the primary convergence indicator. If iterations fail to converge for that criterion an additional check on residual criterion is done and the solution is accepted if residuals are sufficiently small.

3.3 Debugging capability

The OCT solver generates log messages in XML format, enabling automated post-processing and debugging of non-convergence. OCT provides a dedicated Python package that facilitates parsing of the log and extraction of the most relevant information.

To further facilitate interactive non-convergence debugging, the framework includes a Python package that enables user interaction with the equation system from the Python console. The interactive features include temporary elimination of some of the iteration variables and residuals from the equation system, local residual and Jacobian analysis, etc. The interactive framework facilitates equation debugging and enables localization of the problematic residual equation (e.g., with local extrema or discontinuity).

4 Component level examples

4.1 Combustor

The combustor or burner is a key component for gas turbine simulation. It models the injection of a fuel stream into a gas stream, and its partial or complete combustion. It is usually modeled as a two-port or three-port component, based on whether the fuel supply shall be modeled in a physical way (e.g., solving mass flow rates from pressure differences and correlations of wall friction) or simplified. In case of the latter, no physical fluid connector is used for the fuel supply. Instead, parameters or signal inputs are used to specify fuel supply information. In the case of the former, a physical fuel connector with complete information on convective transport quantities such as pressure, mass flow, composition, and specific enthalpy is included, and physical pipe or boundary condition models are connected to the burner. Independently of the modeling abstraction of the fuel supply, the gas inlet and outlet of the combustor are always normally modeled with physical connectors in the Jet Propulsion Library.

Engineering activities call for some flexibility in the modeling of such combustors. One aspect relates to the prescription of fuel flow. The most basic ways of prescribing the latter are to prescribe either the dimensional fuel flow (in units kg/s or a non-SI counterpart) directly, or the non-dimensional fuel-to-air ratio ("FAR"). Following the "thermodynamic matching" principle, upon evaluation of the combustor model equations, a guessed or correct air flow rate entering the combustor will be known. Thus, it is straightforward to compute the dimensional mass flow rate of fuel from FAR in case of the latter, and both are equivalent from a computational order principle. "Thermodynamic matching," then, prefers the usage of non-dimensional FAR over dimensional fuel flow rate because of its higher robustness against variations in gas turbine requirements/size. (As the gas turbine becomes larger, the air flow rate increases. To compensate, an increase in fuel flow is required to maintain otherwise unchanged conditions, which is already accounted for when using FAR as iteration variable and not accounted for when using dimensional fuel flow as iteration variable.).

However, the user typically wants to prescribe fuel flow or FAR indirectly via combustor outlet temperature (so-called "T4"), net thrust, etc. With this implementation, it is possible to switch between these prescriptions while iterating on commonly preferred FAR by switching out residual equations (that enforce equality of T4, net thrust, etc.).

To enable switching between different equation systems for on-and off-design, similar functionality was implemented in the compressor.

4.2 Compressor

The compressor usually receives mechanical power from a shaft and converts it into thermofluidic power by compressing the entering gas – as its name indicates. Following the air flow path, it is typically located before the burner and the turbine and is aimed at bringing the gas to a higher pressure. It is modeled in the Jet Propulsion Library as a two-port component – it can also include vectorized bleed ports if desired – and uses maps to define the compressor performance. While the compressor is modeled in a physical way, its map is purely an algebraic abstraction of the performance that enables solving the compressor models in a more robust way.

For the compressor, the library includes a model of the R-line map. R-lines are sets of curves that can be parallel to the surge line and evenly spaced among each other. The use of such an artificial interpolation to coordinate R-lines (sometimes also called argument lines or beta lines) ensures unique results in the regions of low corrected air flow, where pressure ratio is almost a constant and regions of constant air flow towards the highest air flow region for a given speed line (avoiding table look-up along vertical or horizonal tangents).

The performance map is a good example of how on- and off-design sets of equations are different and how the Physics-based Solving is well suited to this workflow:

- For on-design simulation, the scaling factors of the map are computed based on the design point performances.
- For off-design simulation:
  - These scaling factors shall now be fixed parameters and therefore are held constant during the simulation, and the equations that defined them are hidden to the solver.
  - The off-design operating points and surge margin are an output of the map and are computed based on the off-design point definition.

For both simulation modes, the R-line is an iteration variable that is associated to a residual equation that enforces the corrected air flow from the map to be equal to the physical airflow computed in the compressor.

5 Optimization of a jet engine – a system example

5.1 System under study

In this part, the optimization of a jet engine (also described as cycle model in the literature – due to the importance of thermodynamics effects) is discussed. The architecture selected and objectives for the parameter values are based on [SIE19] in which the technology investigated corresponds to an Entry-into-Service in year 2035 for geared turbofan engine.

A geared turbofan is shown in Figure 4. It usually includes one inlet fan that can be divided into a center part that supplies the flow for the core region (fanCore) and a coaxial outer section that supplies air for the bypass (fanByp). The center part of the engine includes two compressor stages (ipc and hpc), the combustion chamber or burner (brn), and the turbine consisting of a high-pressure section (hpt) and a low-pressure section (lpt). There are two nozzles at the outlet, one for the bypass (nozByp) and one for the center part (nozCore). A gearbox (gear) between the turbine and the fan allow different rotational speed (e.g., for optimized efficiency of each section). It enables a design in which the speed of the inlet fan with its large blades is reduced and the compressor and turbine can be operated at higher speeds.

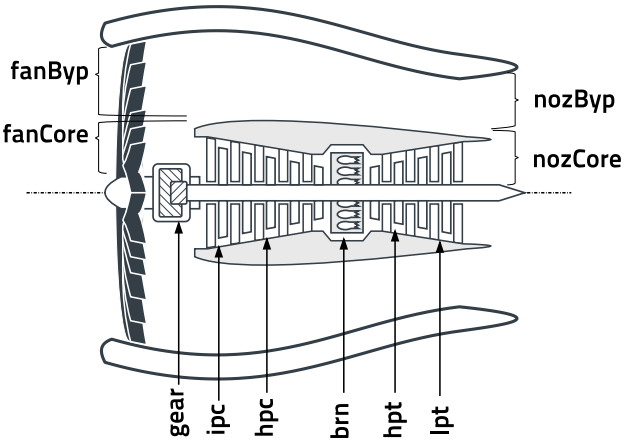

Figure 4. Cross-section view of a geared turbofan design

Figure 5 presents the associated model of this geared turbofan developed using Modelon Jet Propulsion library.

On the left of the schematic, the air flow enters the gas turbine via an inlet. Eventually, it is split into the core flow (through combustor and turbines) and bypass flow. The bypass flow terminates, after the bypass side of the fan and a duct model, with the bypass nozzle. The core flow passes through the usual set of compressors, the burner, and the usual set of turbines. The connections on the top of the compressor and turbine components represent the bleed air flow. Air is extracted from the high-pressure compressor both for customer uses (routed to boundary custBld) and to cool the high-pressure and low-pressure turbines. At the bottom of the compressors and turbines, the mechanical shaft connections are made. The high-pressure spool and the low-pressure spool with fan gear are shown. Eventually, the core flow is disposed of via the core nozzle. Analysis overview

In order to achieve a proper design of the cycle, a set of simulations on several operating points has been performed. The scheme involves the following steps:

- Converging a simulation for the design point
- Extracting the gas turbine sizing and initialization data for use in off-design points (see [BEC15] for more details on this)
- Creating a simulator instance for all off-design points (e.g., cruise and take-off)

We exemplify the need for iteration described in Figure 1 via the definition of cooling flows to be extracted from the high pressure compressor to the high pressure turbine and low pressure turbine (see above cycle description). Typically, they must be set to some approximate value on the design point, and then the resulting turbine blade metal temperatures must be computed in all off-design cases (so-called “uniform blade temperatures” in gas turbine parlance). In order to enforce a maximum temperature across all cases, the gas turbine designer must iterate on the cooling flow fractions on the design case until all temperature constraints are met.

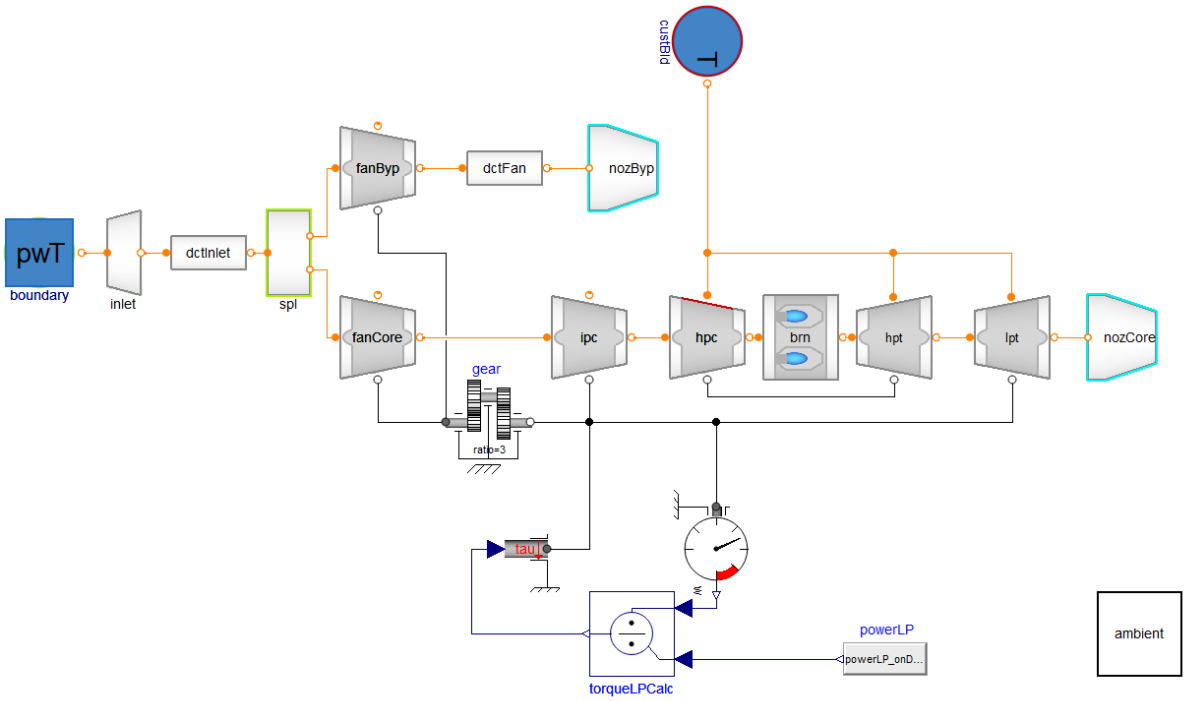

Figure 5. Architecture of the geared turbofan to optimize

5.2 Cycle missions and simulation modes

In [ZHA19], a slight variant of the above-shown geared turbofan model was investigated for a A320-200 type aircraft. The model setup was replicated for this effort. [SIE19] provides data on the accuracy of the resulting data match. The general missions presented in this communication are listed in Table 1, below.

	Top of climb	Cruise	Take-off
Thrust	24 kN	18 kN	92.5 kN
Day type	ISA	ISA	Hot day (ISA+15 K)
Altitude	35000 ft	35000 ft	0 ft
Mach Number	0.78	0.78	0.25
Type	On-Design	Off-Design	Off-Design

Table 1. Cycle design missions

The design point in the missions under study is the "Top of Climb" – i.e., it is aerodynamically the most constraining mission point for the sizing of the gas turbine, optimizing its specific fuel consumption (SFC), while keeping the uniform blade temperature (UBT) below 1240°C. Note that the SFC corresponds to the ratio between the fuel consumption and the thrust of the jet engine. Nevertheless, the other missions are relevant to include as they represent the points with highest thrust requirement (take-off) and longest duration (cruise, thus driving overall mission block fuel consumption).

5.3 Cycle design, results and discussions

For the geared turbofan under study and modeled with Modelon Jet Propulsion library, the overall pressure ratio (OPR) technology variable was set to 55, which is a relevant order of magnitude for a 2035 technology. Overall pressure ratio is the product of all fan and compressor pressure ratios, and a key technology variable as it increases the thermal efficiency of the cycle. A manual convergence was achieved by varying the above-mentioned cooling flows on the design point, and all temperature criteria were eventually met for the different mission profiles.

In a second step, in a small design space exploration exercise, the OPR was varied down to 50 and up to 60 (however, the manual convergence of cooling flows and temperatures is not included in the analysis). Figure 6 shows a key cycle design result: the specific fuel consumption (SFC) versus the OPR.

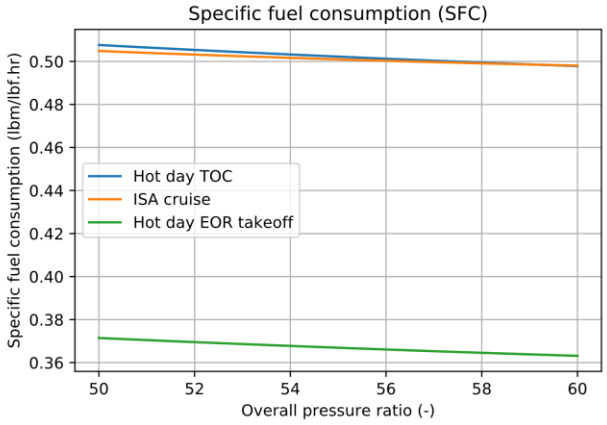

Figure 6. Design exploration, SFC v/s OPR

Figure 6 shows how the specific fuel consumption improves with increasing OPR. The most relevant line is the orange one, which is weighed highly in the total mission block fuel. From this plot alone, we would be tempted to directly move to even higher OPR values to leverage the fuel consumption improvement. However, upon reviewing results on other key variables (the above-described temperatures), we see that further manual convergence work is required to yield the complete picture.

In Figure 7 we see the corresponding results. Based on material and cooling technology constraints, the uniform blade temperature (UBT) shall remain below 1240 K. The manual convergence was applied at OPR 55 and yields exactly these or lower values for all operating conditions and turbines. Figure 7 shows how the UBT varies within the design exploration.

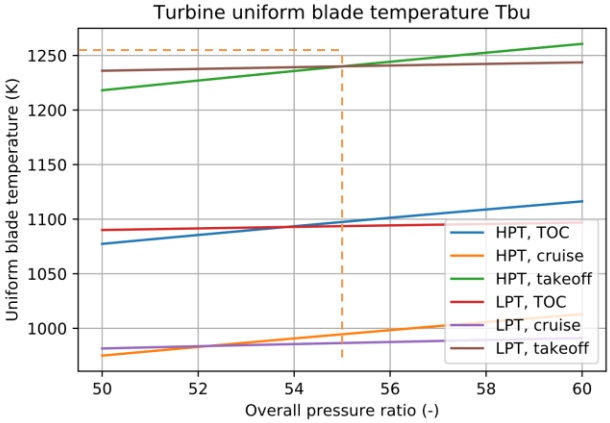

Figure 7. Design exploration, UBT v/s OPR

Keeping the cooling flow fractions constant results in violation of temperature constraints on turbine blade metal temperatures. This convergence would have to be achieved not only for an OPR of 55 but for all the OPR values to avoid dropping with UBT below acceptable temperatures at lower OPR (leaving unused efficiency potential on the table) and going beyond acceptable UBT at higher OPR (and thus resulting in an infeasible design). This plot thus illustrates the need for iterations on the flow fractions when sizing a cycle model.

To illustrate the challenge further, the variation in velocity ratio and specific thrust are plotted respectively in Figure 8 and Figure 9. To yield a balanced and efficient design, the gas turbine designer intends to maintain these at specific values, which lead to high efficiency [KYP17]. These values would have to be included in the manual convergence, too, and illustrates the complexity of the required analysis work.

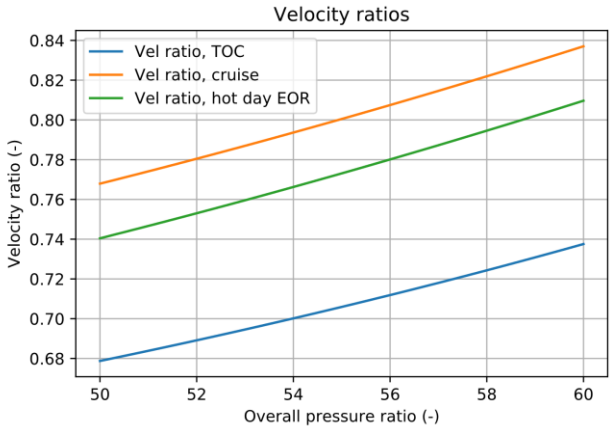

Figure 8. Design exploration, VR v/s OPR

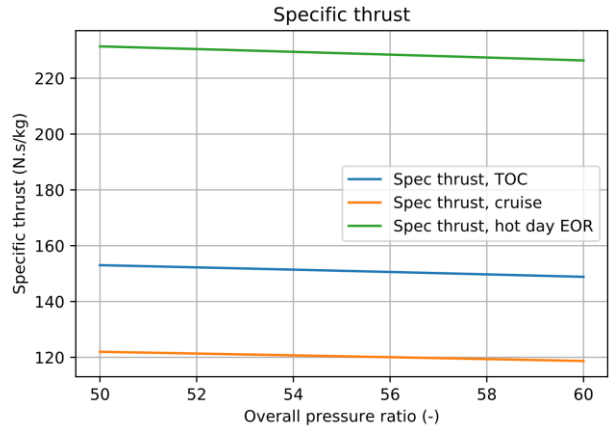

Figure 9. Design exploration, ST v/s OPR

Figures 5 to 8 illustrate how robustly the steady-state cycle analysis problems can be solved with the new Physics-based Solving functionality. This analysis is enabled by the functionality introduced above, such as for the extraction of solutions from one simulation for the initialization of another. At the same time, the figures also illustrate how a full optimization of the cycle design for given objectives requires observing and satisfying constraints, which, based on problem complexity, can be tedious. This observation indicates a potential next step in our work – to implement automatic multi-point design techniques to solve the above-described consistency via the already-used equation solver across the design point and all the off-design points. This is an ongoing development that is beyond the scope of this paper, and it will be described in a future publication.

6 Conclusion and perspectives

This paper discussed how the Modelon products enable suiting the design workflow of a typical cycle model. This simulation was made possible by properly addressing the technical requirements for following such a workflow and developing a dedicated Physics-based Solving design in the library to meet these requirements. Additionally, Modelon Optimica Compiler Toolkit as well as pyFMI and FMIT products are enablers for this workflow. Illustrations of this statement were performed in component and system level examples.

While this paper addresses the complexity of gas turbine design and illustrates the benefit of Modelon dedicated products, it also touches the multi-point design problem without covering it in the examples.

7 References

[BEC15] Becker, R.-G., Bolemant, M., Krause, D, Peitsch, D., *An automated process to create start values for gas turbine performance simulations using neural networks and evolutionary algorithms*, https://ieeexplore.ieee.org/xpl/conhome/8365669/proceeding International Gas Turbine Congress, Tokyo, Japan, 2015.

[BOU18] Bouskela, D. & Jardin, A., *ETL: A New Temporal Language for the Verification of Cyber-Physical Systems*, 2018 Annual IEEE International Systems Conference (SysCon), Vancouver, Canada, 2016.

[COI16] Coïc, C., Fu, J. & Maré, J.-C., *Bond Graphs Aided development of Mechanical Power Transmission for Aerospace Electromechanical Actuators*, International Conference on Bond Graph Modelling, Montréal, Canada, 2016.

[COI18] Coïc, C. & Biéron, M., *An Evolutive Bond Graph Modeling of Aerospace Hydraulic Reservoirs and its Modelica Implementation*, International Conference on Bond Graph Modelling, Bordeaux, France, 2018.

[ELM94] Elmqvist, H., Otter., M., *Methods For Tearing Systems Of Equations In Object-Oriented Modeling. European Simulation Multiconference*, Barcelona, Spain, 1994.

[EIG12] Eigner, M., Gilz, T. & Zafirov, R., *Proposal for functional product description as part of a PLM solution in interdisciplinary product development.* International Design Conference, Dubrovnik, Croatia, 2012.

[KYP17] Kyprianidis, G. K., Dahlqvist, E., *On the trade-off between aviation NOx and energy efficiency. Applied Energy, volume 185, pages 1506-1516*, Elsevier, 2017.

[OAT97] Oates, G. C., *Aerothermodynamics of Gas Turbine and Rocket Propulsion.* AIAA Education Series, 1997.

[OLS08] Olsson, H., Otter, M., Mattsson, S. E. & Elmqvist, H., *Balanced models in Modelica 3.0 for increased model quality.* Proceedings of the 6th International Modelica Conference, Bielefeld, Germany, 2008.

[OTT15] Otter, M., Nguyen, T., Bouskela, D., Buffoni, L., Elmqvist, H., Fritzson, P., Garro, A., Jardin, A., Olsson, H., Payelleville, M., Schamai, W., Thomas, E. & Tundis, A., *Formal Requirements Modeling for Simulation-*

Based Verification, Proceedings of the 11th International Modelica Conference, Versailles, France, 2015.

[SIE12] Sielemann, M., *Device-Oriented Modeling and Simulation in Aircraft Energy Systems Design*. PhD Dissertation, Munich, Germany, 2012.

[SIE17] Sielemann, M., Pitchaikani, A., Selvan, N. & Sammak, N., *The Jet Propulsion Library: Modeling and simulation of aircraft engines.* Proceedings of the 12th International Modelica Conference, Praha, Czech Republic, 2017.

[SIE19] Sielemann, M., Thorade, M., Nguyen, A., Zhao, X., Sahoo, S. & Kypriandis, K., *Modelica and Functional Mock-Up Interface: Open Standards for Gas Turbine Simulation.* ASME TurboExpo, Phoenix, USA, 2019.

[WAL04] Walsh, P. P. & Fletcher, P., *Gas Turbine Performance.* John Wiley & Sons, 2004.

[VDI04] VDI GUIDELINE 2206, *Entwicklungsmethodik für mechatronische Systeme – Design methodology for mechatronic systems*, Beuth, 2004. (in German).

[ZHA19] Zhao, X., Sahoo, S., Kyprianidis, K., Sumsurooah, S., Valente, G., Rashed, M., Vakil, G., Hill, C. I., Jacob, C., Gobbin, A., Bardenhagen, A., Prössl, K., Sielemann, M. Rantzer, J. & Ekstedt, E., *A Framework for Optimization of Hybrid Aircraft.* ASME TurboExpo, Phoenix, USA, 2019.