ENHANCED STEADY-STATE IN MODELON JET PROPULSION LIBRARY

AN ENABLER FOR INDUSTRIAL DESIGN WORKFLOWS

#OneModelon

Modelon

Moritz Hübel

Matthis Thorade

Clément Coïc

AUTHORS

Modelon

CONTENT

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

- Industrial Design Workflows

CONTENT

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

- Industrial Design Workflows
- Dynamic v/s Steady-state simulation

CONTENT

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

- Industrial Design Workflows
- Dynamic v/s Steady-state simulation
- Enhanced steady-state in Modelon Jet Propulsion library

CONTENT

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

- Industrial Design Workflows
- Dynamic v/s Steady-state simulation
- Enhanced steady-state in Modelon Jet Propulsion library
- Component-level examples
- System-level example

CONTENT

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

- Industrial Design Workflows
- Dynamic v/s Steady-state simulation
- Enhanced steady-state in Modelon Jet Propulsion library
- Component-level examples
- System-level example
- Conclusion

CONTENT

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

INDUSTRIAL DESIGN WORKFLOWS

A basic example:

V = R * I

"Typical" simulation

Off-Design Simulations

- ✓R is a known parameter
- ✓V (or I) is a known variable
- ➢I (or V) is computed

On-Design Simulations

- ✓Both V and I are known variables <u>at design point</u>
- ➢R is computed based on this point

INDUSTRIAL DESIGN WORKFLOWS

A basic example:

V = R * I

"Typical" simulation

Off-Design Simulations

- ✓R is a known parameter
- ✓V (or I) is a known variable
- ➢I (or V) is computed

On-Design Simulations

- ✓Both V and I are known variables <u>at design point</u>
- ➢R is computed based on this point

Wait… but there is no need to rearrange the equation?

Nice!
Let's discuss more.

Indeed, that is a-causality! And the best part… the Modelica compiler takes care of that!

INDUSTRIAL DESIGN WORKFLOWS

So, why do we need both on- and off-design modes?

On-Design answers "what should be my system design?"
Off-Design answers "how does my designed system behave?"

So on-design should happen first?

And off-design simulation should use the sizing from on-design

Exactly! And both modes are done with a single model.

If I understand correctly, the compiler will re-order the equations to solve a problem or the other… even if it's a single model?

Yes, the compiler will prepare both cases so we can switch mode without re-doing all the equation processing!

That can be quite a mess, no?

We have a great compiler team, and we can even write insights in our models to guide the compiler with our knowledge of physics

Amazing! You should write a paper on the topic!

Guess what…

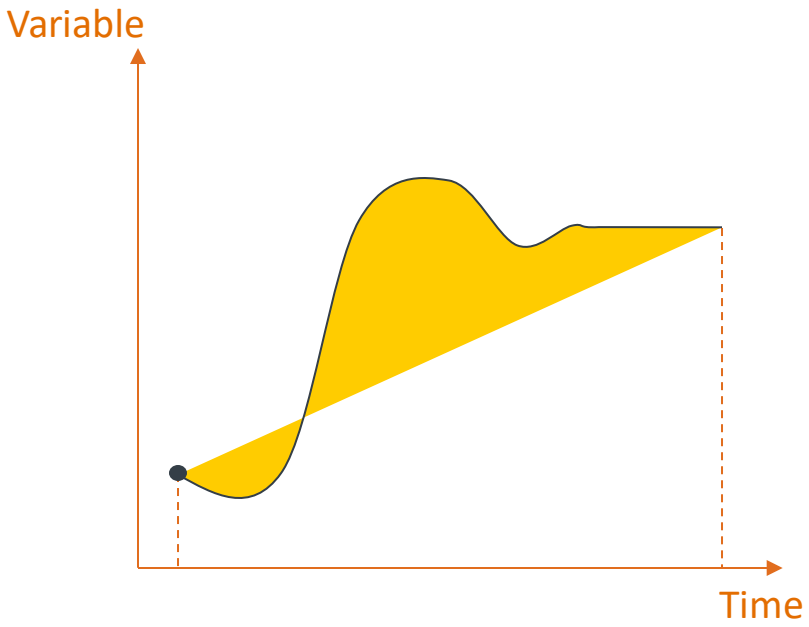
Modelon

CONTENT

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

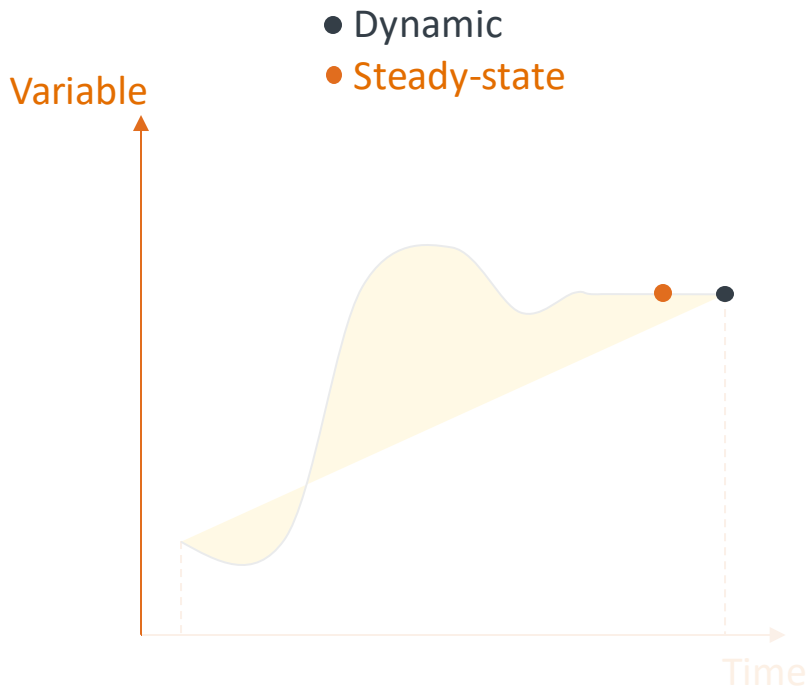
STEADY-STATE SIMULATION: WHAT IS IT?

- Dynamic



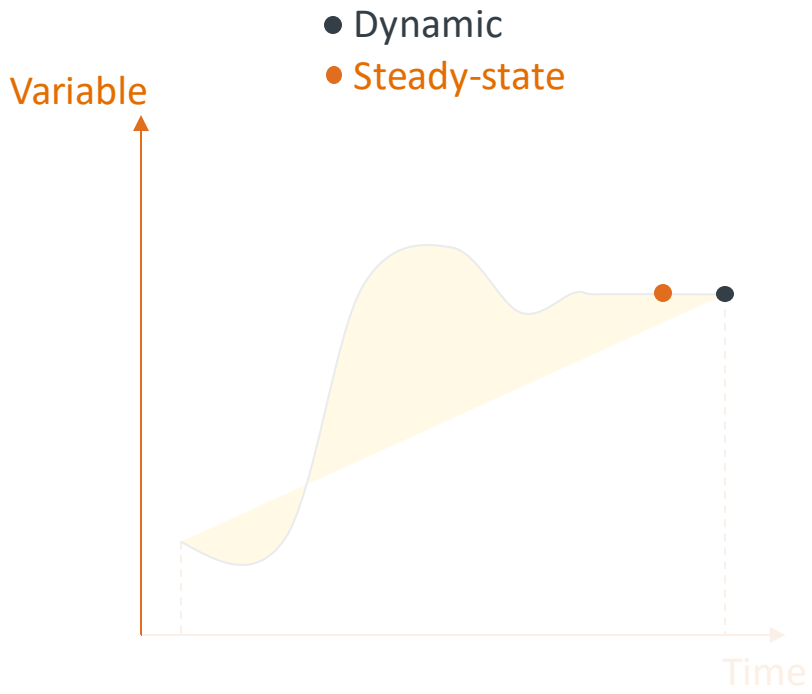
Modelon

STEADY-STATE SIMULATION: WHAT IS IT?



1. Define operating conditions of the modeled system
2. Solve the model to have all variables steady

STEADY-STATE SIMULATION: WHAT IS IT?



1. Define operating conditions of the modeled system
2. Solve the model to have all variables steady

- No notion of time, no history
- Direct results of interest
- Several magnitudes faster
- Easier results processing
- Facilitates sequential simulation

Modelon

CONTENT

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

- Industrial Design Workflows
- Dynamic v/s Steady-state simulation
- Enhanced steady-state in Modelon Jet Propulsion library
- Component-level examples
- System-level example
- Conclusion

ENHANCED STEADY-STATE: OCT CAPABILITIES

ENHANCED STEADY-STATE: OCT CAPABILITIES

Compiler

Rearrange the acausal equations into causal algorithms
Automatically derive the sensitivities for optimization

ENHANCED STEADY-STATE: OCT CAPABILITIES

Compiler

Rearrange the acausal equations into causal algorithms
Automatically derive the sensitivities for optimization

Capability to **hide residual equations and iteration variables** from the solver in a compiled model.

ENHANCED STEADY-STATE: OCT CAPABILITIES

Compiler

Rearrange the acausal equations into causal algorithms
Automatically derive the sensitivities for optimization

Capability to **hide residual equations and iteration variables** from the solver in a compiled model.

➡ Key to switch between simulation modes

ENHANCED STEADY-STATE: OCT CAPABILITIES

Compiler

Rearrange the acausal equations into causal algorithms
Automatically derive the sensitivities for optimization

Capability to **hide residual equations and iteration variables** from the solver in a compiled model.

➡ Key to switch between simulation modes

Solver

Scaling of residual equations and iteration variables
Combination of residual and step norm as exit criterion

Modelon

ENHANCED STEADY-STATE: OCT CAPABILITIES

Compiler

Rearrange the acausal equations into causal algorithms
Automatically derive the sensitivities for optimization

Capability to **hide residual equations and iteration variables** from the solver in a compiled model. ➡ Key to switch between simulation modes

Solver

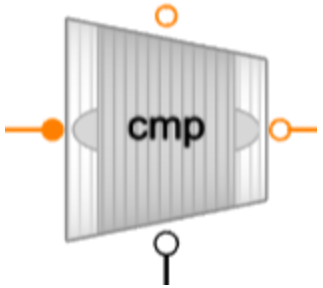
Scaling of residual equations and iteration variables
Combination of residual and step norm as exit criterion

Debugging

Log messages in XML format
Python package for parsing the log and data extraction, and user interaction with the equation system

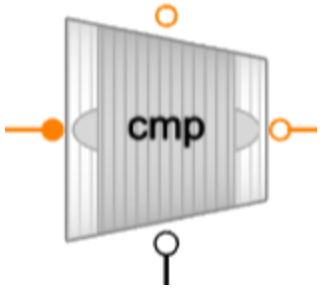
Modelon

ENHANCED STEADY-STATE IN MODELON JPL



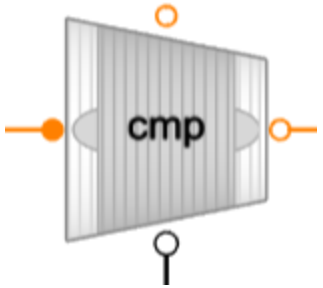
Physics-based Solving:
Vendor-specific language construct supported by Modelon’s compiler
Specify which equation should be a residual associated to which iteration variable
Holding of residuals or iteration variables is possible through parameter conditions

ENHANCED STEADY-STATE IN MODELON JPL



Physics-based Solving:
Vendor-specific language construct supported by Modelon’s compiler
Specify which equation should be a residual associated to which iteration variable
Holding of residuals or iteration variables is possible through parameter conditions

ENHANCED STEADY-STATE IN MODELON JPL



Physics-based Solving:
Vendor-specific language construct supported by Modelon's compiler
Specify which equation should be a residual associated to which iteration variable
Holding of residuals or iteration variables is possible through parameter conditions

Physics-based Solving instructions at component level enables system assembly and solving is deduced from topology
Use of generally accepted choices for iteration variables and residual equations from gas turbine community

Modelon

CONTENT

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

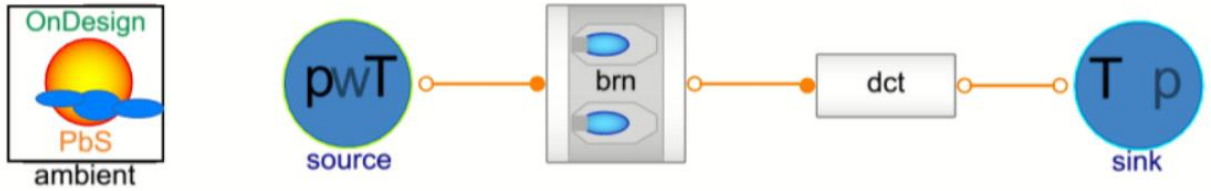
- Industrial Design Workflows
- Dynamic v/s Steady-state simulation
- Enhanced steady-state in Modelon Jet Propulsion library
- Component-level examples
- System-level example
- Conclusion

COMPONENT EXAMPLE: BURNER

Physics-based Solving with Jet Propulsion Library & OCT

1. FMU generation

In this experiment, we will illustrate how it is possible to change the type of input on the burner from Fuel-to-Air-Ratio (FAR) to fuel mass flow rate (wFuel) after recompilation, on a burner instance:



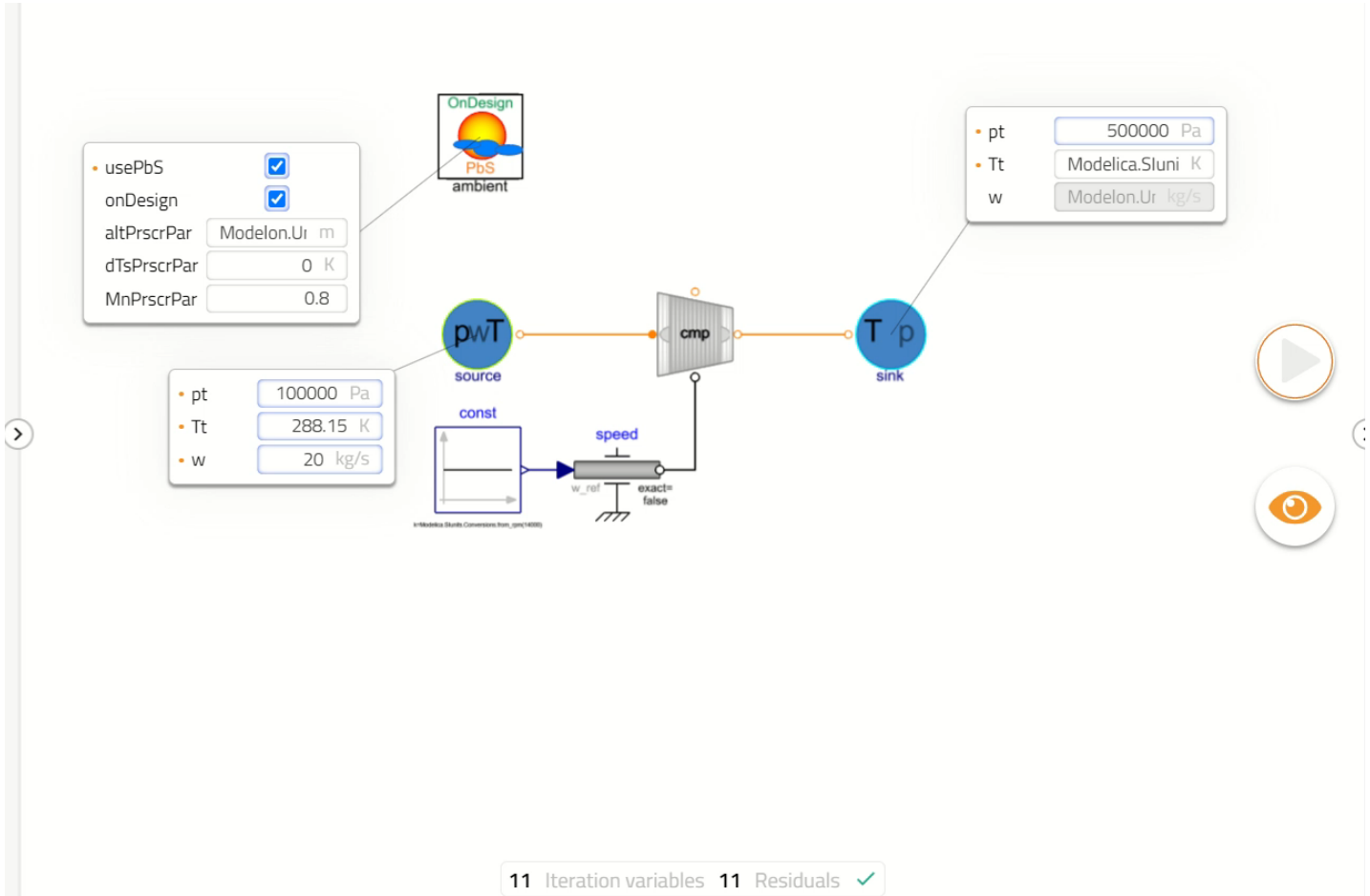
In [1]:
```
modelName = "JPL_PbS_Tests.Burner.PbS_far"
```

Run script that defines the library paths and compiler options:

In [2]:
```
import sys
sys.path.insert(1, 'C:/Users/ClementCoic/Documents/0_Developement/P545-JPL/JetPropulsion/Resources/Python')
import jplUtilities
import importlib
importlib.reload(jplUtilities)
from pyfmi import load_fmu
from oct.steadystate.nlesol import FMUProblem, Solver
```

Speed x1

COMPONENT EXAMPLE: COMPRESSOR

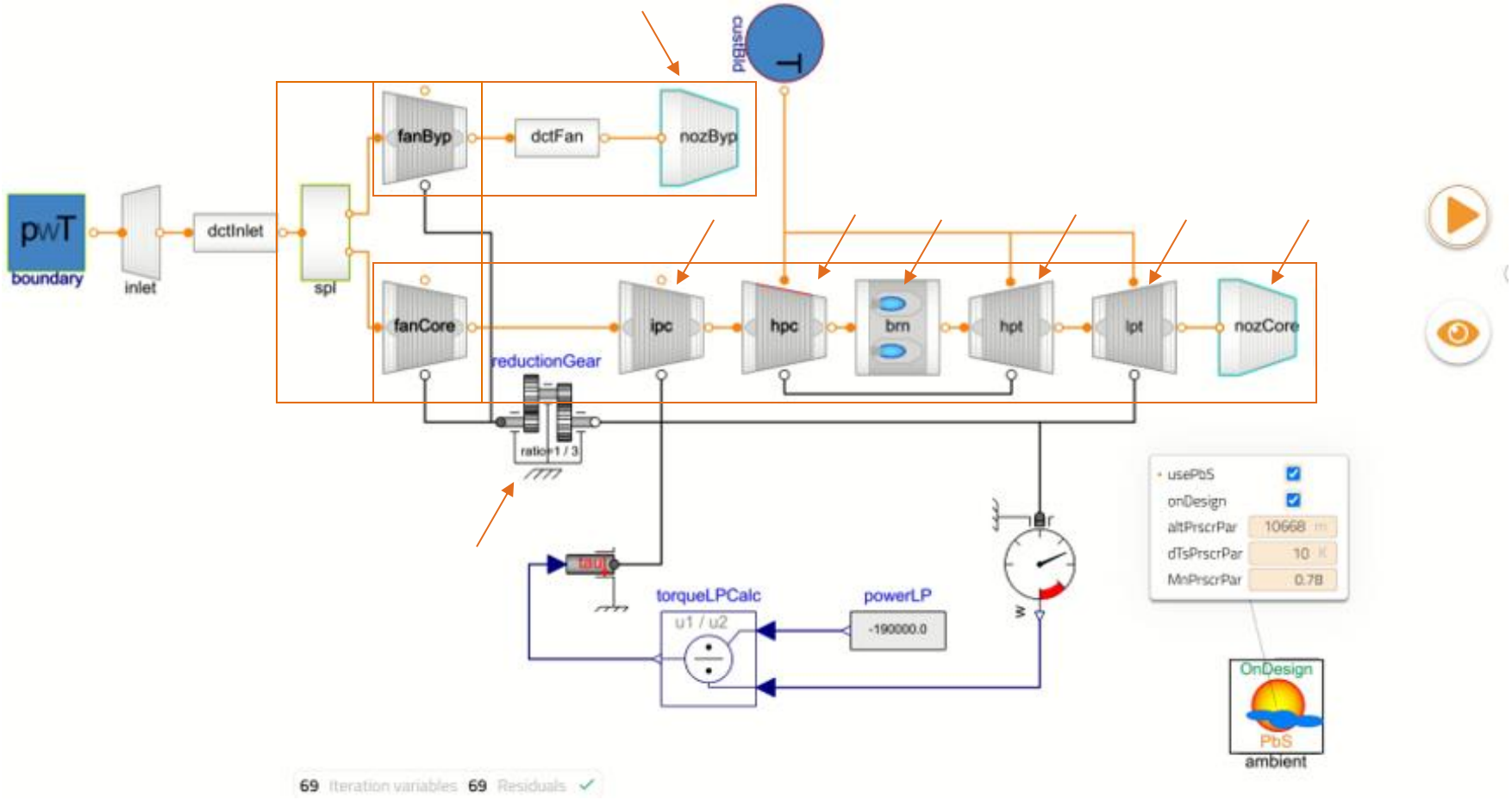


Speed x1

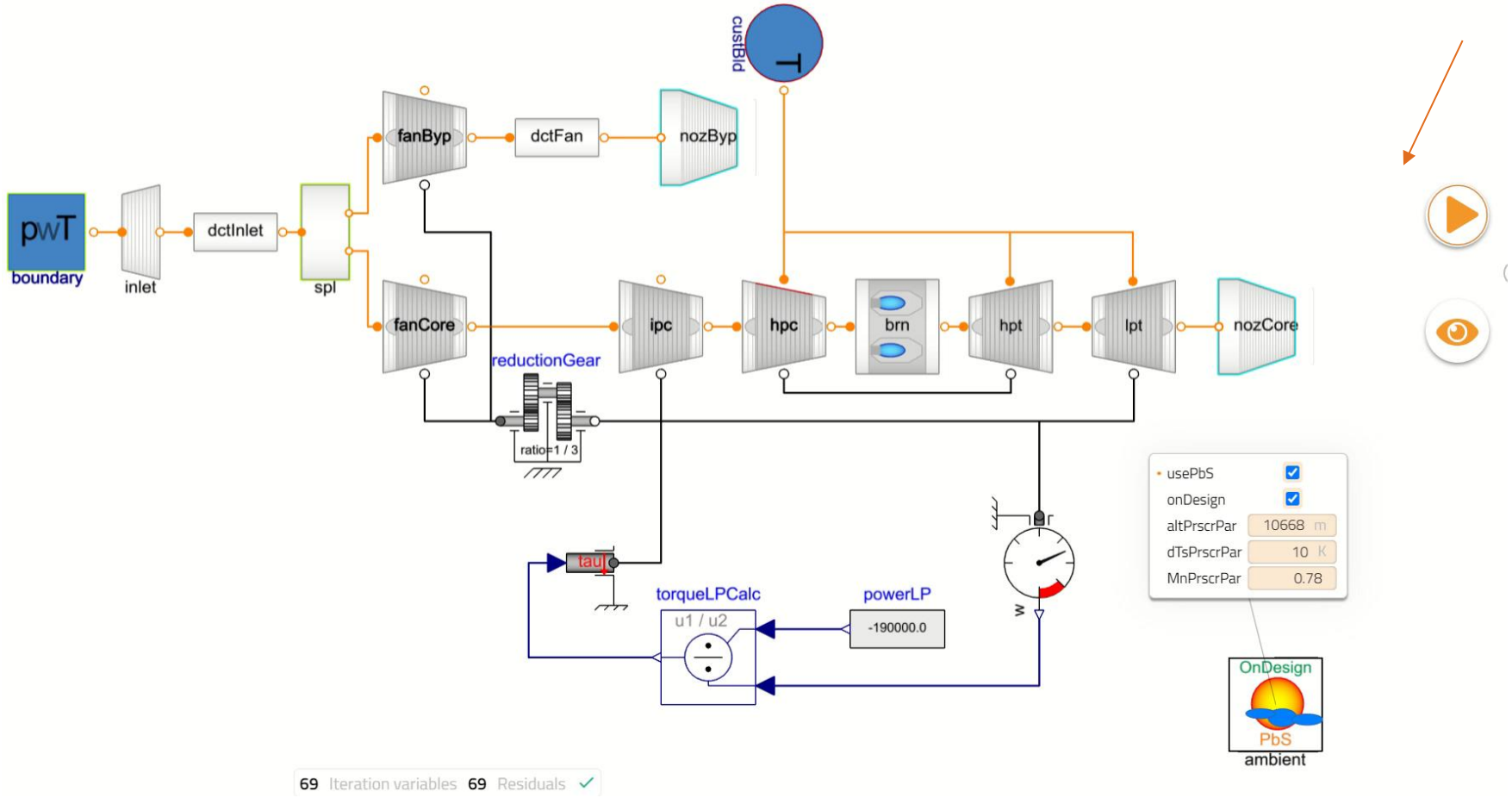
CONTENT

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

GEARED TURBOFAN



GEARED TURBOFAN



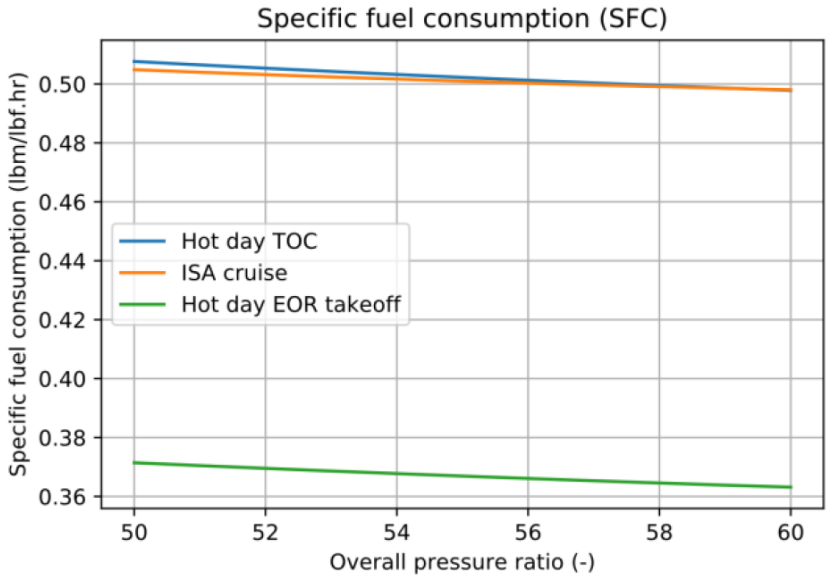
Speed x1

GEARED TURBOFAN: RESULTS & DISCUSSIONS

	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

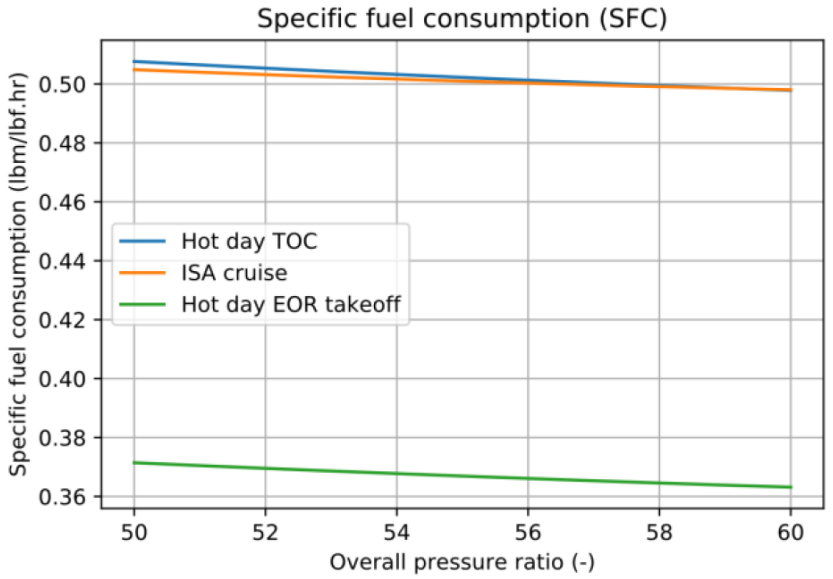
GEARED TURBOFAN: RESULTS & DISCUSSIONS

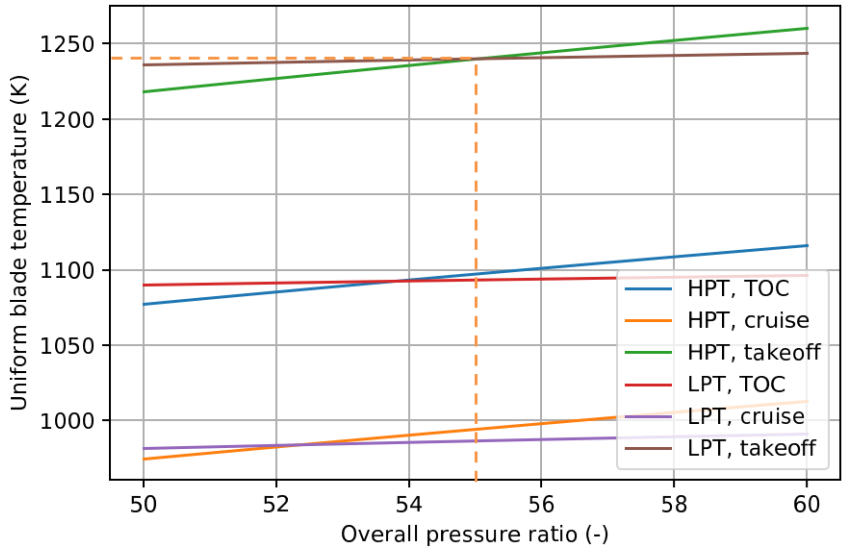
	Top of climb	Cruise	Take-off
Thrust	24 kN	18 kN	92.5 kN
Day type	ISA	ISA	Hot day (ISA+15 K)
Altitude	35000 ft	35000 ft	0 ft
Mach Number	0.78	0.78	0.25
Type	On-Design	Off-Design	Off-Design



GEARED TURBOFAN: RESULTS & DISCUSSIONS

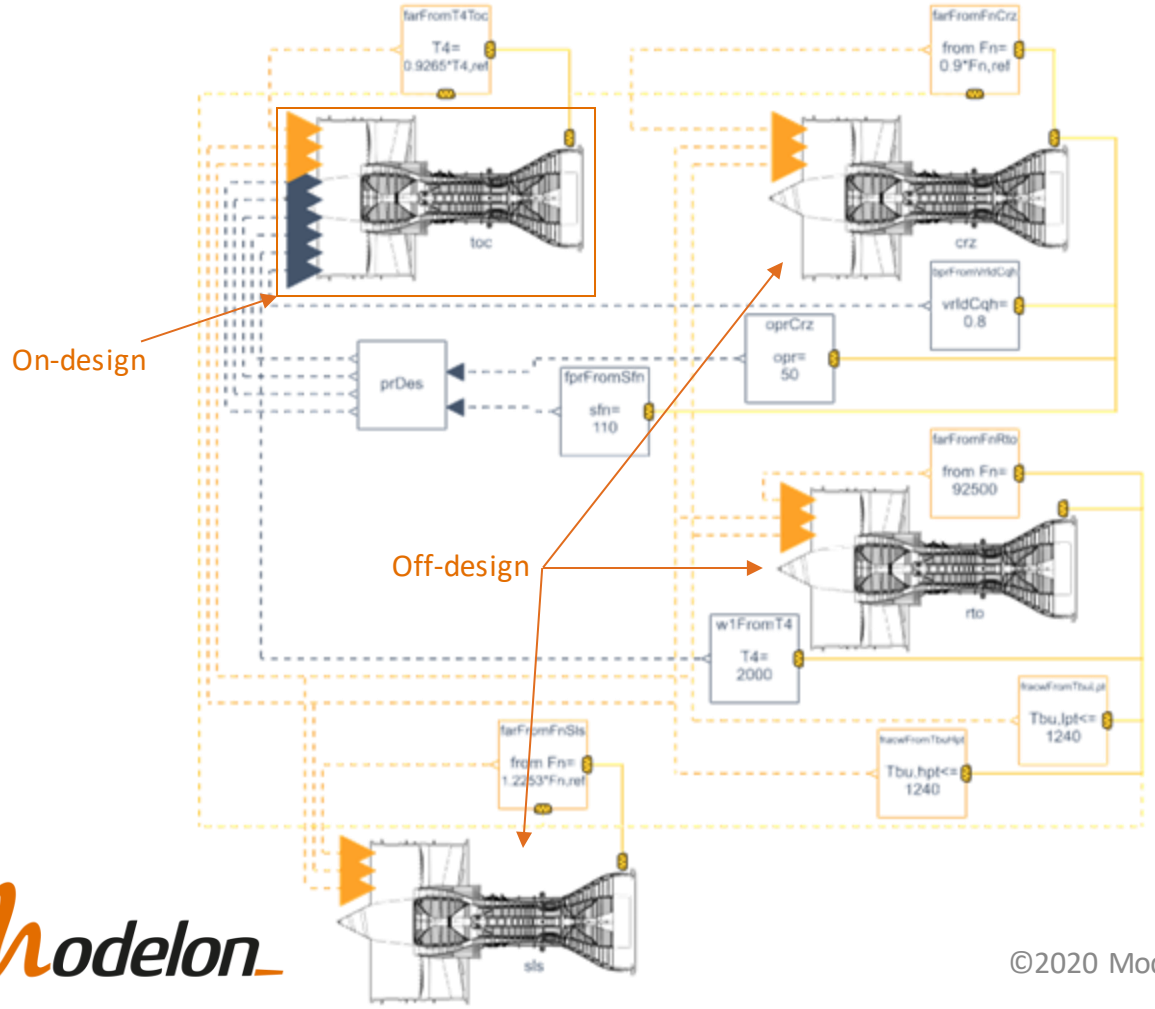
	Top of climb	Cruise	Take-off
Thrust	24 kN	18 kN	92.5 kN
Day type	ISA	ISA	Hot day (ISA+15 K)
Altitude	35000 ft	35000 ft	0 ft
Mach Number	0.78	0.78	0.25
Type	On-Design	Off-Design	Off-Design





There are constrain requirements that could be violated on other mission profile points.

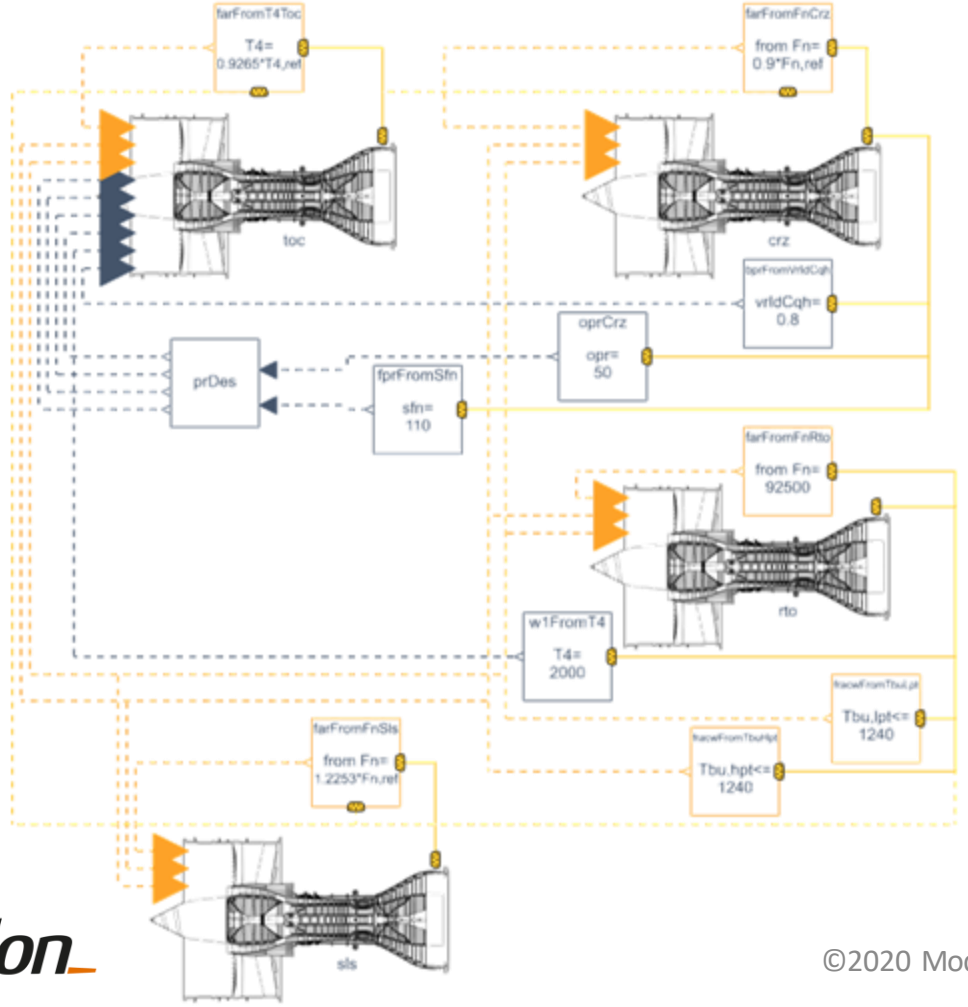
GEARED TURBOFAN: MULTIPOINT DESIGN



On-design and Off-Design simulations are modeled in a single model
We ensure that the design meets the constrains in all off-design points
If not, the design is automatically modified

(At each step of the convergence, the design is propagated to all off-design models)

GEARED TURBOFAN: MULTIPOINT DESIGN



On-design and Off-Design simulations are modeled in a single model
We ensure that the design meets the constrains in all off-design points
If not, the design is automatically modified

(At each step of the convergence, the design is propagated to all off-design models)

More about multi-point design at
https://www.youtube.com/user/modelonweb

Modelon

CONCLUSION

- Introduced terminology
  - Simulation modes: on- and off-design
  - Steady-state
- Industrial design workflow
- Robust and efficient steady-state simulation of JPL with OCT or Modelon Impact
- Demos at component and system levels
- Graphical multi-point design

QUESTIONS?

Thank you for your time

Modelon