ICAS 2020 SHANGHAI
International Council of the Aeronautical Sciences
32nd Congress of the International Council of the Aeronautical Sciences
September 6-10, 2021
Pudong Shangri-La, Shanghai, China

# CONNECTING SYSTEM SIMULATION TO AIRCRAFT CONCEPT DEVELOPMENT

Alexandra Oprea[1], Robert Hällqvist[2], Ludvig Knöös Franzén[3], Magnus Eek[4], Ingo Staack[3] & Hampus Gavel[2]

[1]Aeronautical Technologies, Saab Aeronautics, Linköping, Sweden
[2]Vehicle Systems, Saab Aeronautics, Linköping, Sweden
[3]Fluid and Mechatronic Systems, Linköping University, Linköping, Sweden
[4]Strategy & Business Development, Saab Aeronautics, Linköping, Sweden

## Abstract

This study presents a solution for connecting system simulation and aircraft concept development using solely open standards. An easy-to-use optimisation framework for aircraft concept development is created with the help of the Modelica, Functional Mock-up Interface (FMI), and System Structure and Parameterization (SSP) standards, and the open source tools OpenModelica and OMSimulator. The framework allows for conceptual aircraft design accounting for transient phenomena by means of standardised integration of dynamic simulation models of aircraft subsystems. The framework is applied to an industry-relevant use case concerning the concept development of a generic fighter aircraft. The generality and modularity of the framework and its straightforward implementation enables tailoring of the optimisation goals to the user needs and requirements. The adoption of industry-wide standards allows for the inclusion of system simulation models developed in the modelling tool best suited for each discipline, thus integrating dynamic system simulation already at the aircraft conceptual design stage.

**Keywords:** system simulation, aircraft conceptual design, modelling, FMI, SSP

## 1. Introduction

The aircraft development process spans several years and involves the collaboration of a multitude of disciplines. The evolution towards model-based engineering (MBE) has reduced development time, risks, and costs; therefore, model development is now an established part of the development process. Often, each discipline develops their simulation models in the Modeling & Simulation (M&S) tool best suited for the purpose, both from a technical capability perspective and from a usability one. The results of the in-depth analysis of the system of interest by the subject-matter experts are the foundation of the individual systems development. The knowledge acquired through such analyses is communicated to the interested parties whenever necessary [1].

Interdisciplinary M&S is often restricted by the compatibility between the individual tools used by each respective model developer. The direct exchange of simulation models is hindered by inconsistencies in the modelling techniques, languages and interfaces. As a result, tool- or vendor-specific solutions have to be developed and maintained, which increases the risks of integration errors. In recent years, attention has been paid to the development of standards facilitating the connection of tool-specific models into one, larger, system simulation.

The flexibility conferred by the possibility of connecting simulation models from different model developers enables their collaboration already from the very beginning of the design process. The traditionally static aircraft conceptual design methodologies can now be enhanced by the inclusion of system dynamics. Studying the influence of system transients on an overall design already at an early stage ensures that dynamic phenomena that might be a dimensioning factor for the final aircraft design are not overlooked.

### 1.1 Contributions

The objective of this study is to present an aircraft concept development framework based exclusively on open standards and open source tools. Even though the heart of the framework is implemented in open source software, models originating in any M&S tool, open or proprietary, can be incorporated provided that the tool supports the FMI or SSP standards. A novelty of the conducted research is the established connection between system simulation and concept development. This connection facilitates the incorporation of transient phenomena in the traditional concept development approaches. In comparison, traditional concept development frameworks focus on deriving aircraft concepts achieving the required static point performance. Additionally, the research has resulted in a tool-agnostic workflow for the preliminary aircraft sizing and optimisation process - a process that usually is executed in proprietary software.

## 2. Theoretical background

### 2.1 Aircraft conceptual design

In the past, Aircraft Conceptual Design (ACD), also denoted as sizing, classically relied on low-level, low-fidelity, empirical and semi-empirical Level-0 methods (Raymer [2], Torenbeek [3], Roskam [4], etc.). The availability of higher computational power and the need for enhanced estimation accuracy has triggered the transition to simple physics-based Level-1 and more complex physical-based Level-2 (accurate physics representation; see [5] for the fidelity level definition) analysis methods.

The goal of this early design stage is to perform an as complete as possible design space exploration (with limited effort and incomplete data and knowledge) to shrink down the design space to one or few design solutions only. Therefore, ACD can be interpreted as a Multidisciplinary Design Optimisation problem with the trend of involving more and more topics into the design optimisation. Large collaborative design frameworks, including knowledge-based engineering and automated workflow management, allow nowadays for automated design analysis [6]. However, to enable a fast execution, easy adaptation to specific design or system needs, and make the design analysis understandable for the involved stakeholders, the implementation of Level-0 and Level-1 methods and models is still beneficial for early-stage ACD investigations.

### 2.2 Optimisation

Aircraft sizing is here viewed as a multi-objective optimisation problem on the general form

$$\begin{aligned} \underset{\vec{x}}{\text{minimize}} \quad & f(\vec{x}) \\ \text{subject to} \quad & \vec{x_{low}} \geq \vec{x} \leq \vec{x_{high}} \end{aligned} \tag{1}$$

where the total objective function

$$f(\vec{x}) = \sum_{i=1}^{n} \lambda_i f_i(\vec{x}) + \sum_{j=1}^{k} w_j G_j(\vec{x}) \tag{2}$$

is the sum of objective subfunctions $f_i(\vec{x})$ multiplied by weights $\lambda_i$. The constituent objective subfunctions $f_i$ map to individual simulation model outputs originating from the simulation of an Operational Concept (OpsCon) [7]. The design variables are denoted $\vec{x}$ and their upper and lower bounds are given by $\vec{x_{high}}$ and $\vec{x_{low}}$ respectively. Additional constraints can be accounted for by means of adding $k$ additional weighted penalties, denoted $w_j G_j(\vec{x})$, to the objective function. In Equation 2, $G_j(\vec{x})$ penalises the objective function such that the feasibility constraints, additional to the upper and lower bounds on the design variables, are fulfilled.

The direct search non-gradient based Complex-RF optimisation method [8] is used to solve the optimisation problem presented in Equation 1. Gradient based methods are deemed as inappropriate for use in this framework, because gradients may be expensive to estimate when there is a simulation model in the optimisation loop [9]. This is particularly true for exported black box simulation models, an export option supported by the FMI standard. Population-based optimisation methods are inherently parallel, which is a desirable property in terms of scalability [10]. Such methods perform particularly well when a large number of processors are available. However, direct search methods, such as the Complex-RF, perform well with a smaller number of central processing units and research to parallelise the Complex-RF method shows promising results [10, 11].

### 2.3 Open standards

The FMI standard [12] offers an interface solution for the exchange and execution of dynamic simulation models. At the time of writing, the FMI standard can be considered well established with over 150 tools officially supporting the standard to varying degrees [13]; in comparison, the SSP standard [14, 15] is quite young, as it was first released in 2019, and the current tool support is scarce. The complete set of tools officially supporting the SSP standard is available at [16].

In short, FMI specifies the functions an exported executable, compliant to the standard, should support if it is executed outside of its original M&S environment. Additionally, FMI provides a standardised format for communicating the interface of the exported model in a so called *Model Description* eXtensible Markup Language (XML) file. This interface specification is packaged, together with the executable, in a .zip file format denoted as a Functional Mock-up Unit (FMU). The related SSP standard [14] defines a way to store and apply architecture and parameters to coupled simulation models, for example FMUs. The SSP standard specifies several different XML file formats in order to describe an instance of a set of coupled simulation models, two of which are used in the presented research: the System Structure Description (SSD) format to describe the simulation architecture, and the System Structure Values (SSV) format to store the parameter values of a specific configuration. These SSP artefacts are, just as in the FMI standard, packaged in a .zip file format, denoted as an SSP file, together with the referenced resources. Referenced resources are, for example, other SSP files or one or more FMUs.

Both the FMI and SSP standards are used in the open source tool OMSimulator [17] which is used here as an architecture development and Master Simulation Tool (MST). This tool is available as a plug-in to the open source OpenModelica Connection Editor (OMEdit), used for the creation of dynamic simulation models in the Modelica language [18, 19].

## 3. Sizing methodology

Figure 1 shows an overview of the sizing methodology that is the focus of this paper. The sizing is founded on a parameterised assembly of modelled Modelica components included in the Aircraft Conceptual Design Library (ACDLib). The assembly is exported as a FMU from its original M&S tool so that it can be integrated in the developed external sizing optimisation framework. Simulation entities integrated in the sizing framework can be either a single FMU or entire simulators in the form of SSPs. The aircraft geometry and performance parameters, optimal for the specific design and operational requirements, are then deduced through a sizing step in the sizing framework.

The framework itself is built in Python and uses OMSimulator's Python API. The structure is straightforward: there is a main routine where the mission file and the FMUs are assembled into a model and the optimisation boundaries are set. Then, the optimisation routine is called, setting the optimisation parameters. The optimisation routine itself calls the objective value function routine, where the simulation is run with the inputs decided by the optimisation routine. After the simulation the values of the required parameters are read from the result file using the Modelica Buildings Library *buildingspy* [20]. When the optimum design is found the results are analysed using the *matplotlib* library [21].

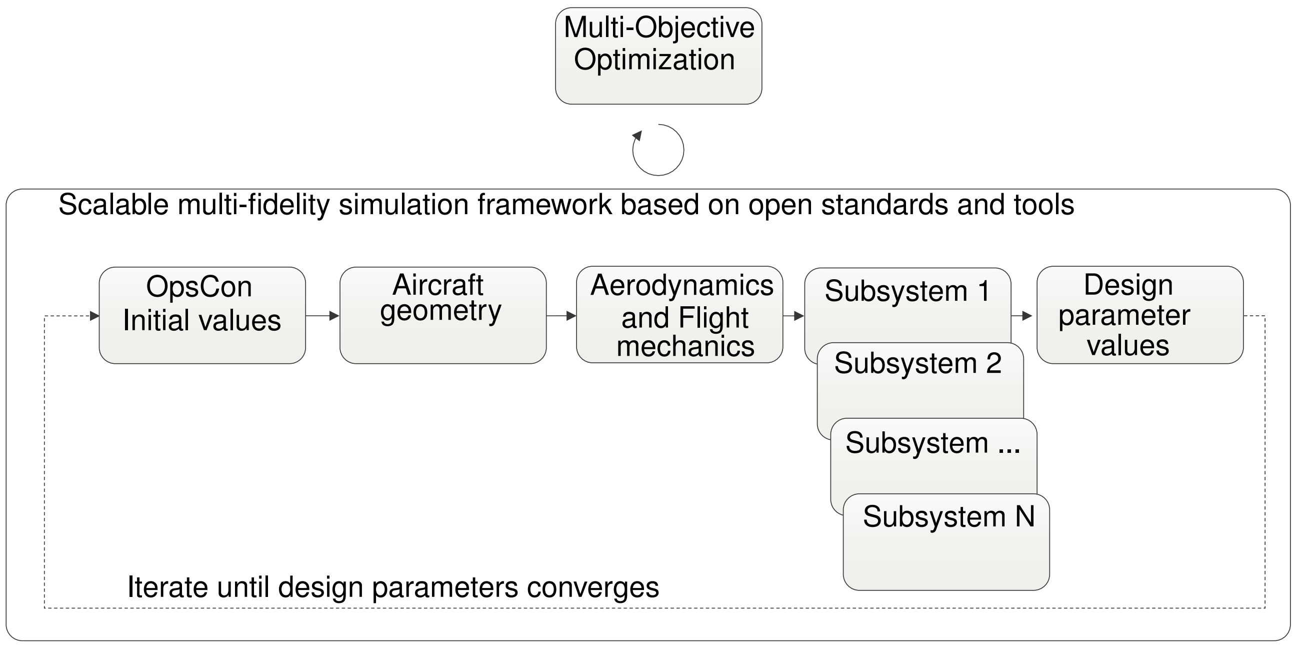

Multi-Objective Optimization
Scalable multi-fidelity simulation framework based on open standards and tools
OpsCon Initial values
Aircraft geometry
Aerodynamics and Flight mechanics
Subsystem 1
Subsystem 2
Subsystem ...
Subsystem N
Design parameter values
Iterate until design parameters converges

Figure 1 – Schematic view of the aircraft sizing optimisation framework. A set of initial design parameter values and a required OpsCon mission are used to initiate the aircraft sizing. The sizing optimisation continues until the design parameter values resulting from the conducted simulation(s) match the design parameter values specified during simulation initialisation

The ACDLib library contains selected parts of the concept development framework presented by Raymer [2] and is developed in the Modelica language [19]. ACDLib does not depend on any external Modelica libraries other than the open source Modelica Standard Library (MSL) maintained by the Modelica Association [22]. The library overall structure is presented in Figure 2a. It is structured into seven different subpackages: *OpsCon*, *Aircraft*, *Propulsion*, *SystemInfo*, *Dynamics*, *Assemblies*, and *Export*. The *Propulsion* subpackage contains components describing both *Electric* and *TurboFan* solutions. Similarily, the *Dynamics* subpackage is partitioned into components concerning *Aero*, *Flight*, and *Vehicle* dynamics. The library components relevant for the case study, see Section 5, are presented in the following paragraphs and subsections.

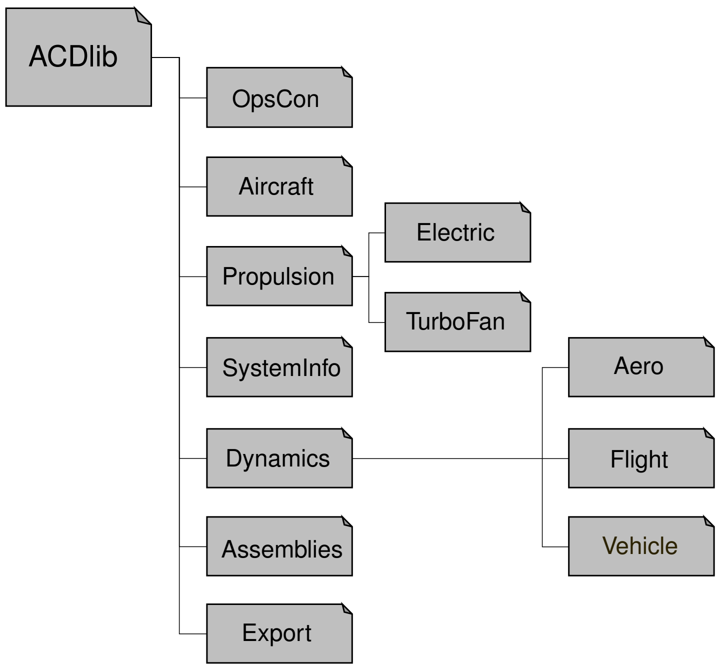
## 3.1 OpsCon

The OpsCon package contains a component that is generic; it can be used to generate missions of any length and character. Additionally, an interpolation-based component is included allowing for the incorporation of externally specified design missions. These design missions can be expressed as either **.csv* or **.mat* files. The generic mission component required inputs are altitude, velocity, and duration of each of the static segments of the modelled mission. Additionally, appropriate climb and descent rates are required to calculate the transition between the static mission segments. These mission parameters are specified in the *SystemInfo.System* component. The OpsCon library components are primarily used for development of assemblies and verification in the Modelica environment.

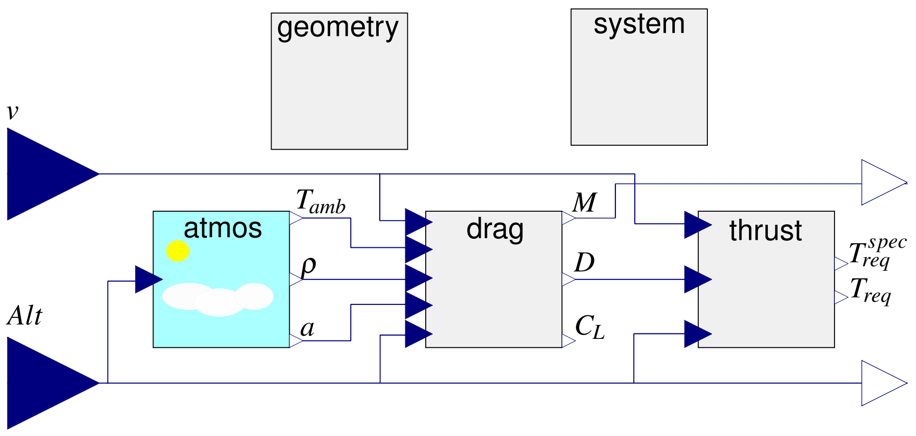
A Python implementation of the general mission component presented by Hällqvist et al. [23] is deployed in the sizing framework. This component is used to generate OpsCon **.csv* or **.mat* files that are used directly during sizing. Components in the *ACDLib.OpsCon* package can then be used, if desired, to incorporate these externally defined missions into the selected Modelica environment.

## 3.2 SystemInfo

All the components in the *SystemInfo* subpackage supply global information via the Modelica *inner* and *outer* concepts. The subpackage includes an atmospheric model and a *System* component.



(a) Aircraft Concept Development library (ACDlib) overall structure. The library is comprised of seven Modelica subpackages containing component models relevant for aircraft concept development in a Modelica, or FMI and SSP, compliant environment



(b) OMEdit screenshot of a model available under the *Export* subpackage, see Figure 2a. This particular export model consists of instances of the *Atmos* and *System* components (available in the *System* subpackage), the *Drag* and *Thrust* components (available in *Dynamics->Aero* subpackage). The instantiated *Geometry* component is positioned in the *Aircraft* subpackage

Figure 2 – Overview of Modelica classes developed for Aircraft Conceptual Design

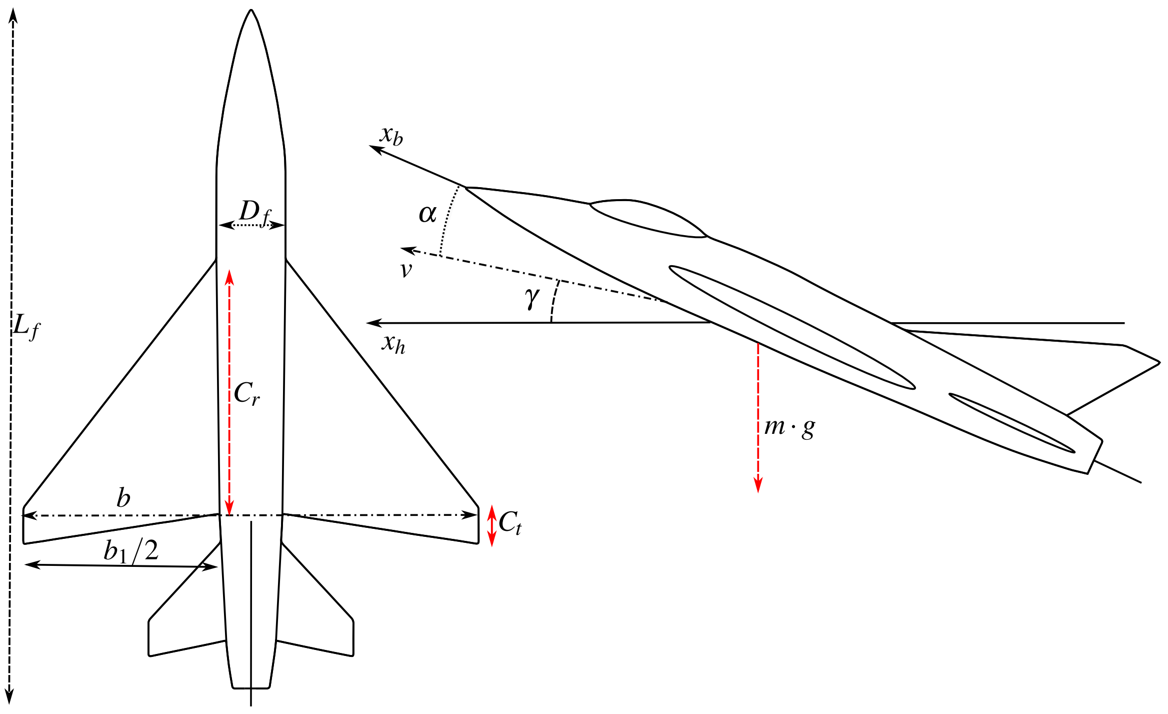


Figure 3 – The aircraft geometrical parameters incorporated in the Modelica library geometry component model are summarised in the figure to the left. The relevant aircraft angles of orientation, the angle of attack $\alpha$ and the flight path angle $\gamma$, are defined in the figure to the right

The atmospheric model is based on the ISOs International Standard Atmosphere from 1976 [24]. The *System* component supplies parametric information concerning the design mission, provided that the user is not using an externally specified mission, as well as physical global quantities not captured in the atmospheric model.

## 3.3 Aircraft.Geometry

The sizing process is dependent on the given aircraft and mission requirements. Some of the parameters can be deduced from the requirement specification (e.g.: type of aircraft, number of crew, payload weight) and others have to be deduced through some design space exploration or via algebraic relations. These parameters are declared in the *Aircraft.Geometry* model.

The fighter aircraft model is based on a simple geometrical model: a cylindrical fuselage of constant diameter, a nose and a tail cone, a swept wing and a conventional tail. A rough sketch of the aircraft and its most important design dimensions is provided in Figure 3. Table 1 presents a subset of the parameters that need to be decided on beforehand in the current version of ACDLib. Appropriate values can be found by looking at similar aircraft designs, or by following methodologies described by [2] or others. Any of the parameters of Table 1 can be specified as design variables. As design variables, the values are no longer fixed, and the parameter value is instead determined by the sizing optimisation. The algebraic relations specified in the *Aircraft.Geometry* model connect the design variables to the geometry of the aircraft. A subset of the implemented equations are described in the following paragraphs.

The most important task for the *Geometry* model is to calculate the wing dimensions in order to accommodate the fuel $m_{fuel}$. It is assumed that half of the total fuel is stored in the wings; the rest is stored somewhere else in the aircraft and does not affect the subsequent calculations. With a given fuel density $\rho_{fuel}$ and an assuming that the fuel takes up $16.7\%$ of the wing, the minimum wing volume

| **Description** | **Notation** | **Value** | **Unit** |
|---|---|---|---|
| Taper ratio | $\lambda$ | 0.21 | [-] |
| Sweep angle | $\Lambda$ | 30.0 | [°] |
| Aspect ratio | $AR$ | 3.0 | [-] |
| Fuselage diameter | $D_f$ | 1.9 | [m] |
| Fuselage length | $L_f$ | 14.0 | [m] |
| Mean aerodynamic chord | $\bar{c}$ | 5.0 | [m] |
| Thickness to chord ratio | $t/c$ | 0.04 | [-] |

Table 1 – Summary of aircraft concept prerequisites parameters along with their default values

$V_{wing}$ is estimated as

$$V_{wing} = 6.0 \cdot 0.5 V_{fuel} \frac{m_{fuel}}{\rho_{fuel}}. \tag{3}$$

The wing area $S_{wing}$ can be calculated as

$$S_{wing} = \frac{V_{wing}}{t/c \cdot \bar{c}} \tag{4}$$

by dividing the wing volume by the wing thickness-to-chord ratio $t/c$ and the mean aerodynamic chord $\bar{c}$. The outer wingspan can be calculated as

$$b_1 = \sqrt{AR \cdot S_{wing}} \tag{5}$$

provided that the wing area $S_{wing}$ and the aspect ratio $AR$ are known. Finally, to get the complete wingspan $b$ the diameter of the fuselage $D_f$ needs to be added to $b_1$. The reference area $S_{ref}$

$$S_{ref} = S_{wing} + c_{root} D_f \tag{6}$$

can be approximated from the wing area and the area between the wing roots $c_{root}$. Having the $\lambda$ fixed allows for the root chord $c_{root}$ in Equation 6 to be obtained from

$$c_{root} = \frac{2S_{wing}}{b_1 * (1+\lambda)}, \tag{7}$$

and the tip chord $c_{tip}$ from

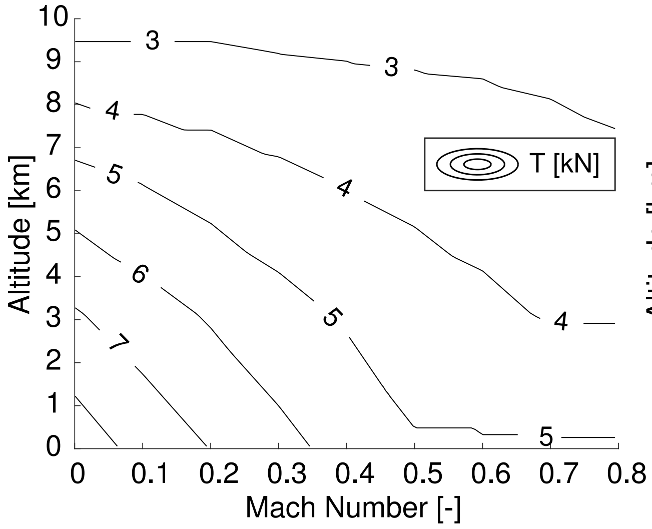
$$c_{tip} = \lambda \cdot c_{root}. \tag{8}$$

The reference area $S_{ref}$, the root chord $c_{root}$ and tip chord $c_{tip}$ are then used in all the subsequent aerodynamics calculations in the components of the *Dynamics.Aero* subpackage.

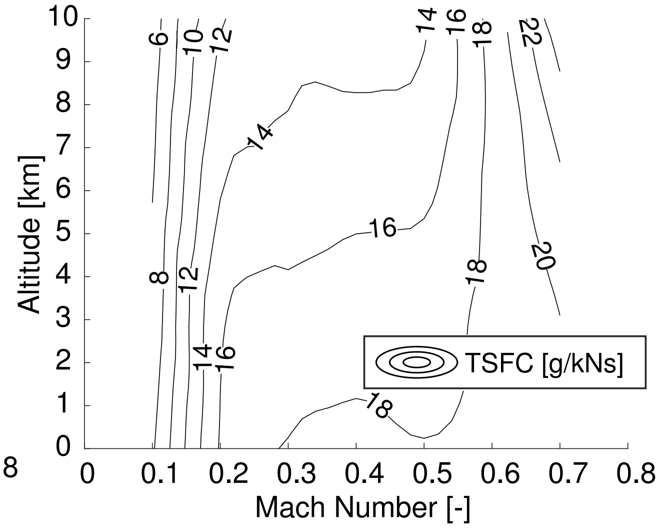
## 3.4 Propulsion

The propulsion subpackage currently contains modelled components related to electric propulsion and turbo fan solutions. A case study incorporating the foremost components was presented at the MODPROD and OpenModelica workshop in February 2021 [25]. These results will therefore not be described in detail here; the focus is instead placed on the propulsion components relevant for the case study of the presented research.

The engine models of the turbofan subpackage are digitalised engine performance maps, including the maximum available thrust and the thrust specific fuel consumption TSFC as functions of altitude and Mach number. The engine used in the case study is the Williams FJ44, with performance data obtained from [26] and open TSFC data made available by Saab Aeronautics [23], see Figure 4. The performance data is incorporated in the subpackage FJ44 component via the MSL model *CombiTable2D* which allows for two dimensional interpolation in, for example, performance maps. The resulting interpolated *CombiTable2D* outputs of TSFC and maximum available thrust $T_{avail}$ are manipulated in a sequence of steps rendering a scalable engine model. The scaling is achieved via a



(a) Maximum available thrust as function of altitude and Mach number.



(b) Estimated and used Thrust Specific Fuel Consumption (TSFC) as function of altitude and Mach number.

Figure 4 – Characteristics of map-based engine model used in the case study. This is one of the turbofan engines incorporated in the *TurboFan* subpackage of the ACDLib

scaling factor $k$ which allows the user to tailor the engine properties to the application needs. This scaling factor acts upon the standard mass $W_{eng_{base}}$ of the engine whose performance map is used, resulting in a scaled engine mass

$$W_{engine} = (k-1)\cdot 100 + W_{eng_{base}} \tag{9}$$

and a scaled output thrust

$$T_{avail} = k \cdot T_{FJ44}. \tag{10}$$

## 3.5 Dynamics

The aerodynamics of the modelled aircraft is calculated in three different modelled components from the *Dynamics.Aero* subpackage: *Drag*, *Thrust* and *Airfoil*.

Based on the considered aircraft, mission, and atmosphere parameters, the lift and drag coefficients $C_L$ and $C_D$, and the lift $L$ and drag $D$ forces are calculated in the *Drag* component. The subsonic parasite drag coefficient is calculated using the component buildup method [2]

$$C_{D_0} = \frac{\sum C_{f_c} FF_c Q_c S_{wet_c}}{S_{ref}} + C_{D_{misc}} + C_{D_{L\&P}} \tag{11}$$

where $C_{f_c}$ is the flat-plate skin friction coefficient, $FF_c$ is the component form factor, $Q_c$ is the component interference factor (here assumed as 1), and $S_{wet_c}$ is the component wetted area. The *component* term refers to any aircraft component to be included: the wing, fuselage, canopy, pylons etc. For the current design, only the wing and the fuselage are included in the calculation.

The total drag coefficient is calculated from

$$C_D = C_{D_0} + \frac{C_L^2}{\pi e AR} \tag{12}$$

where $e$ is an efficiency factor. The resulting total drag is, then

$$D = q C_D S_{ref} \tag{13}$$

where $q$ is the dynamic pressure.

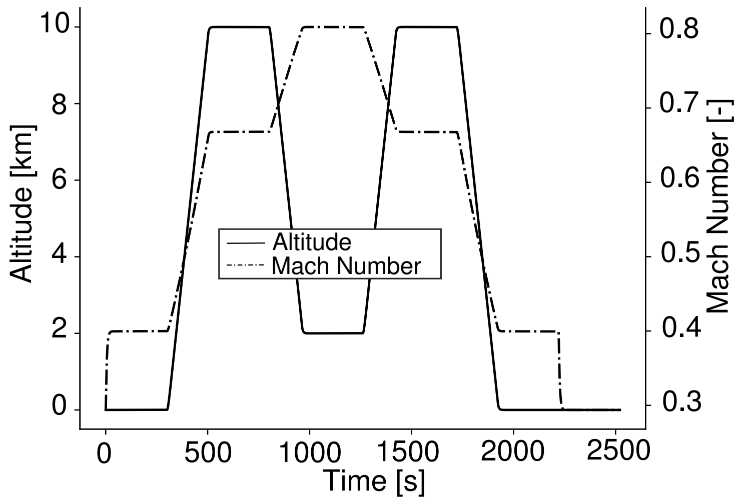


Figure 5 – Altitude and Mach number profiles of the required OpsCon mission.

The drag $D$ is then sent as an input to the *Thrust* component, where from the balance of forces the required thrust $T_{req}$ is calculated according to

$$Ma\sin(\gamma) = T_{req} - D \tag{14}$$

where $M$ is the total aircraft mass, $a$ is the acceleration and the $\gamma$ is the flight path angle. The flight path angle $\gamma$ is obtained from

$$\gamma = \arcsin\frac{\dot{h}}{v}. \tag{15}$$

with $h$ being the altitude and $v$ the velocity of the aircraft.

The required angle of attack $\alpha$ to sustain the calculated lift is estimated in the *Airfoil* component using the lift coefficient $C_L$ and a digitalised version of lift polar diagram for the chosen airfoil.

### 3.6 Assemblies

The connected modelled components of the *ACDLib* library, relevant for each specific conceptual design application, are stored in the *Assemblies* subpackage. Each assembly corresponds to a specific parameterised case study model capturing the aspects relevant for each individual application. The assemblies are designed to be a common denominator between models to be exported and models to be used for experimentation in the selected Modelica environment.

### 3.7 Export

The export subpackage contains assemblies tailored for export and use in non-Modelica tools. These assemblies are typically comprised by models with instantiated models of the *Assemblies* subpackage that inherit predefined export interfaces.

## 4. Case study

The functioning of the proposed framework is demonstrated by means of a case study. A design for a generic single-seat fighter aircraft with a standard configuration (horizontal tail, no canards) is sought. This case study encompasses the derivation of a concept that fulfils a formulated OpsCon, see Section 4.1. Additional to successfully executing the OpsCon mission, the derived concept is specified to fulfil the prerequisites listed in Table 1. The NACA 64-006 airfoil is chosen for its suitability for fighter aircraft and due to the availability of the data [2].

The prerequisites can be seen as case study requirements, and they are here specified as fixed for the sake of simplicity when demonstrating the framework. Either of these parameters could be seen as design variables of the sizing optimisation or incorporated in the model via algebraic equations

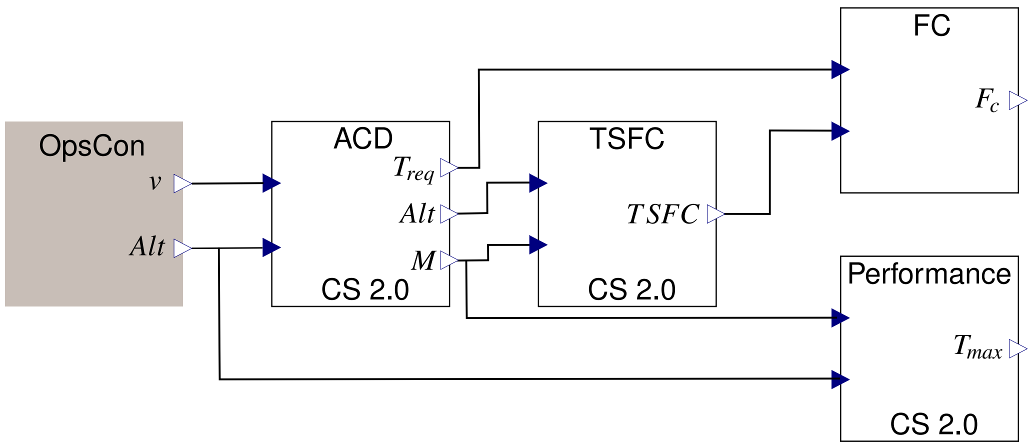


Figure 6 – OMEdit screenshot of the portable SSP of the case study. Each connected component is a individual and replaceable FMU, except for the component denoted OpsCon, which incorporates the requirement boundary conditions. The *TSFC*, *FC*, and *Performance* FMUs constitute a modular modelled engine whereas the *ACD* component is a set of connected ACDlib components relevant for concept development.

related to other design variables.

## 4.1 Operational Concept

To investigate the framework's functionality, a "High-Low-High" mission type for a fighter aircraft is specified as required in OpsCon. The mission includes high altitude flight, low altitude manoeuvres for threat avoidance, and a return-to-base segment at high altitude. The mission profile is presented in Figure 5. The mission is generated given a required climb rate of 50 m/s and a maximum acceleration of 50m/s$^2$. The rate of descent is set as equal to that of the rate of climb for the sake of simplicity.

## 4.2 Sizing

The sizing optimisation schematically shown in Figure 1 is shown in detail in Figure 7. A screenshot of the case study SSP is presented in Figure 6. This SSP is integrated in the optimisation framework with the objective to specify the aircraft design in the form of a populated SSV file. The sizing begins with the specification of the parameter values that are given by the case study requirements or other prerequisites. Additionally, design parameter initial values are specified in this first step of the optimisation workflow. An OMSimulator simulation of the selected SSP is then conducted. At a minimum, this SSP includes an exported assembly of components from the ACDLib library. Once a simulation is conducted, the selected simulated Quantities of Interest are retrieved and stored.

The total objective function value, tailored to the specific conceptual design problem, is computed in the subsequent step. The general objective function presented in Equation 2 is instantiated as

$$f_{casestudy} = \lambda_1 f_1(\vec{x}) + \lambda_2 f_2(\vec{x}) + w_1 G_1(\vec{x}) \tag{16}$$

in the case study. In Equation 16, the two objective subfunctions are

$$\begin{aligned} f_1 &= |m_{tf} - m_{if}|, \\ f_2 &= \max_t |T_{req}(t) - T(t)| \end{aligned} \tag{17}$$

where $\lambda_1 = 1/m_{if}$ and $\lambda_2 = 1/T$ are the selected weights. The weights are chosen such that both objective subfunctions have approximately equal influence on the overall objective. The total fuel mass consumed during a mission, and the initially specified fuel mass, are denoted $m_{tf}$ and $m_{if}$ in Equation 17. The purpose of the second objective subfunction, $f_2$, is to scale the modelled engine such that

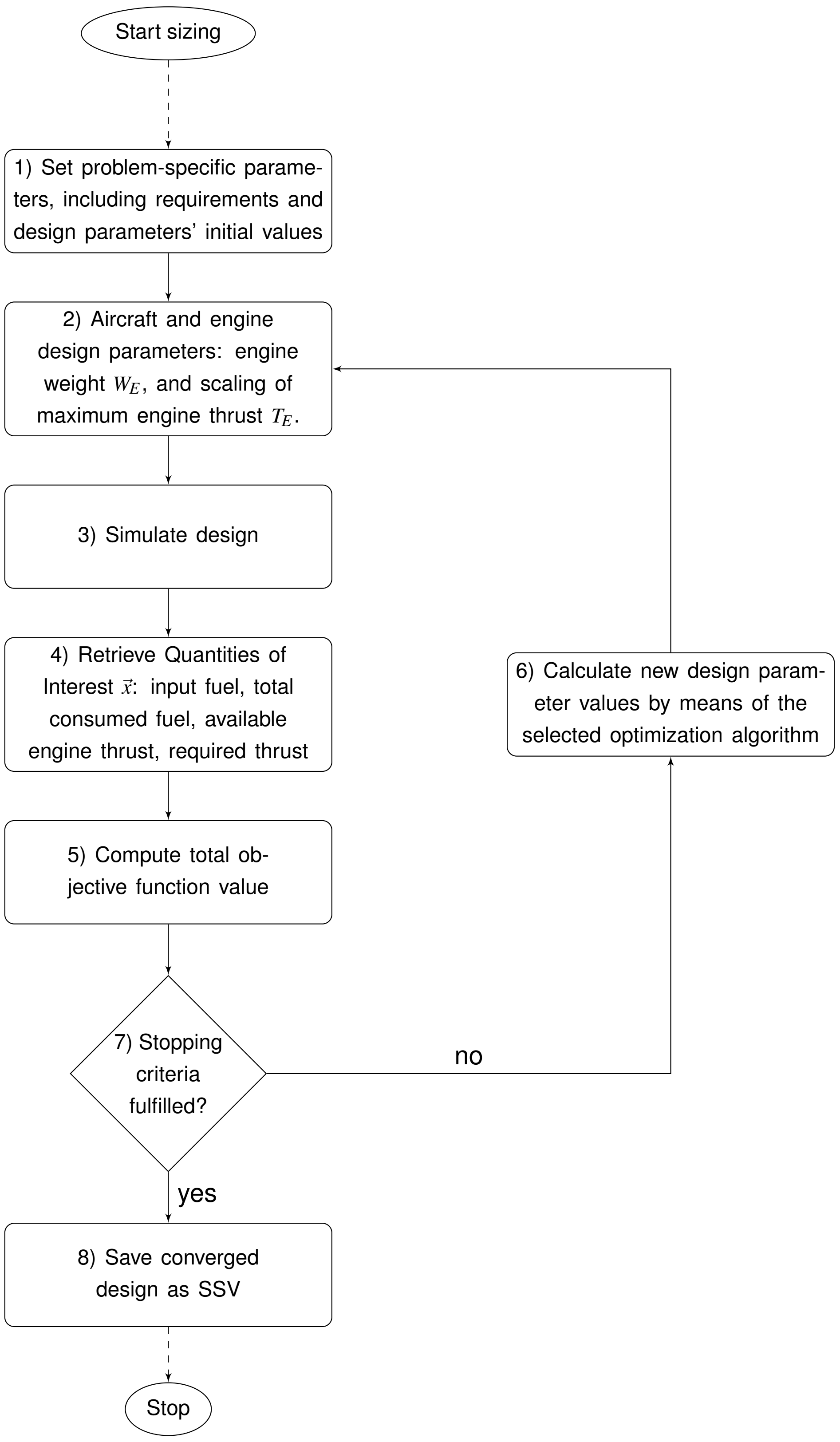

Start sizing
1) Set problem-specific parameters, including requirements and design parameters' initial values
2) Aircraft and engine design parameters: engine weight $W_E$, and scaling of maximum engine thrust $T_E$.
3) Simulate design
4) Retrieve Quantities of Interest $\vec{x}$: input fuel, total consumed fuel, available engine thrust, required thrust
5) Compute total objective function value
6) Calculate new design parameter values by means of the selected optimization algorithm
7) Stopping criteria fulfilled?
no
yes
8) Save converged design as SSV
Stop

Figure 7 – Detailed flow chart of the sizing optimisation presented in Figure 1

the available thrust $T_{avail}$ corresponds to the mission required thrust $T_{req}$ by means of incorporating the identified maximum difference between the two. The third term in Equation 16 is a weighted penalty function

$$w_1 G_1(\vec{x}) = 500 \cdot (1 + \text{sign}(T_{req} - T_{avail})) \tag{18}$$

ensuring that the selected engine can supply the required thrust.

If the objective function reaches a global minimum, then the design has converged and the resulting parameter values are stored in an SSV file. If not, then new design parameter values are calculated, using the selected optimisation algorithm, and a new iteration is commenced.

## 5. Results

### 5.1 Baseline aircraft

The parameters of the derived baseline aircraft concept are listed in Listing 1. A subset of the simulation results, describing the concept behaviour, is presented in Figure 8. The simulated values of drag $D$ and maximum available engine thrust $T_{avail}$ are plotted as functions of time, together with the OpsCon required thrust $T_{req}$, in Figure 8a and Figure 8b respectively. The results presented in Figure 8a contain multiple time segments where the drag and required thrust differ, highlighting the aircraft dynamics. These differences quantify the dynamic characteristics of the aircraft as the required thrust is greater than the drag when the aircraft is accelerating and lower when the aircraft is decelerating.

The comparison between $T_{req}$ and $T_{avail}$ presented in Figure 8b provides information concerning the suitability of the scaled engine to the derived aircraft and the OpsCon. Note that for this particular OpsCon, there are not any transient peaks in $T_{req}$ acting as sizing requirements. The specified low altitude constant speed flight governs the engine scaling. Even though the aircraft acceleration and drag $D$ are far from their peak values in these operating conditions, so is the engine performance, as presented in Figure 4.

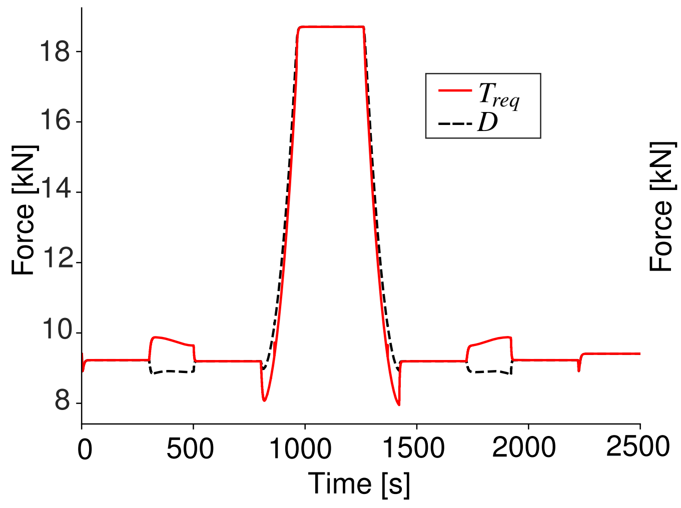
Listing 1: Parameters specifying the aircraft concept derived using the framework described in the presented research. The parameter values are presented in the SSP SSV format

```
<ssv:ParameterSet>
 <ssv:Parameters>
     <ssv:Parameter name="ACD.geometry.b"> <ssv:Real value="9.74"/> </ssv:Parameter>
     <ssv:Parameter name="ACD.geometry.croot"> <ssv:Real value="4.32"/> </ssv:Parameter>
     <ssv:Parameter name="ACD.geometry.ctip"> <ssv:Real value="0.91"/> </ssv:Parameter>
     <ssv:Parameter name="ACD.geometry.FuelMass"> <ssv:Real value="546.00"/> </ssv:Parameter>
     <ssv:Parameter name="ACD.geometry.PropulsionWeight"> <ssv:Real value="837.93"/> </ssv:Parameter>
     <ssv:Parameter name="ACD.geometry.Sref"> <ssv:Real value="28.67"/>  </ssv:Parameter>
     <ssv:Parameter name="ACD.EngineEfficiency"> <ssv:Real value="4.45"/> </ssv:Parameter>
 </ssv:Parameters>
</ssv:ParameterSet>
```

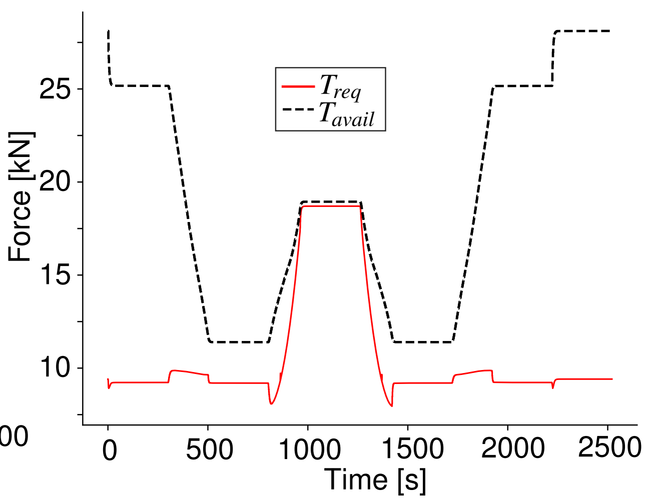
### 5.2 Trade-study: influence of the maximum climb rate

The maximum allowable aircraft climb rate has been varied in order to investigate its influence on the aircraft design. In total, five optimisation runs have been executed, with the climb rate ranging from $30m/s$ to $70m/s$ in steps of $10m/s$.

The climb rate influences the required mission fuel amount. This is shown in Figure 9a. Since the fuel mass is affected so is the total mass of the aircraft, directly changing the size of the engine required to deliver the necessary thrust. Figure 9b shows the size of the engine (expressed as installed engine mass, accounting for the weight of the engine mounts) in all of the five studied cases. The total fuel mass also affects the wingspan of the resulting aircraft. Figure 9c shows the variation of the wingspan with the climb rate. A higher climb rate requires a lower wingspan since the wing needs to accommodate less fuel than in the case of a lower climb rate and a longer mission.

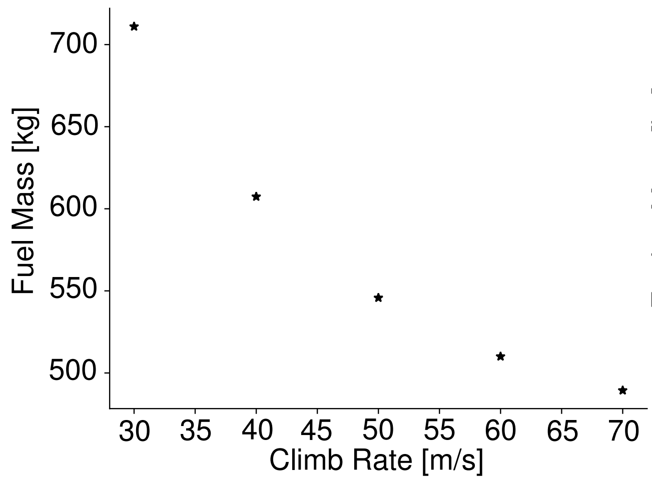


(a) Drag $D$ and required thrust $T_{req}$ as function of time

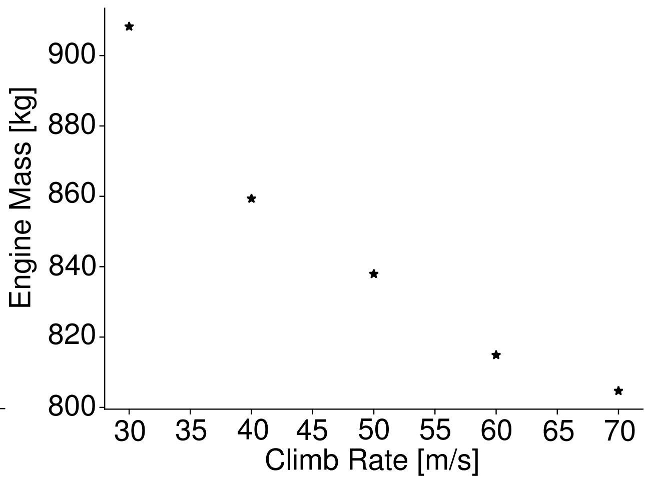


(b) Engine maximum available thrust $T_{avail}$ and required thrust $T_{req}$ as function of time

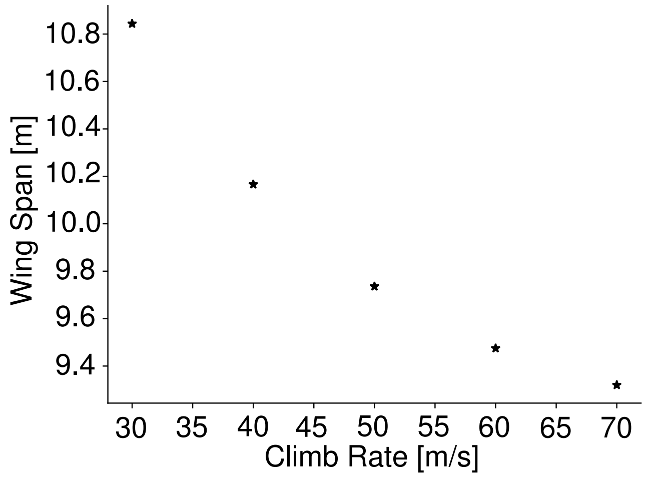
Figure 8 – OpsCon simulation results of the derived baseline aircraft concept. The presented subset of available simulation results provide an overview of the baseline concept characteristics.



(a) The climb rate influence on the total required fuel



(b) The climb rate influence on engine mass



(c) The climb rate influence on the wingspan

Figure 9 – Summary of the trade study results

The parameters for two of the aircraft configurations resulting from the trade study are summarised in two different SSV files: a concept with a maximum climb rate of 30m/s, and a concept with a maximum climb rate of 70m/s. The former is summarised in Listing 2 and the latter in Listing 3. The different derived executable concepts are now in the portable format of a SSP containing the case study executable, see Figure 6, along with three different SSV parameter specifications. These executable configurations can now be deployed and used for analysis in any SSP supporting M&S tool.

The duration of one optimisation round is approx. $900s$, with a set tolerance of $0.001$ for the objective function convergence and a time step of $1.0s$ for the OMSimulator simulations. This is, of course, varying with the mission length, with the shorter missions (higher climb rates) taking as short as $800s$ for the optimisation to converge, and for the longer ones (lower climb rates) about $1000s$. The maximum number of iterations has been set at $500$, but was never reached. All the optimisations converged in 20-50 iterations.

Listing 2: SSV specifying an aircraft with a maximum climb rate of 30m/s

```
<ssv:ParameterSet>
 <ssv:Parameters>
     <ssv:Parameter name="ACD.geometry.b"> <ssv:Real value="10.84"/> </ssv:Parameter>
     <ssv:Parameter name="ACD.geometry.croot"> <ssv:Real value="4.93"/> </ssv:Parameter>
     <ssv:Parameter name="ACD.geometry.ctip"> <ssv:Real value="1.04"/> </ssv:Parameter>
     <ssv:Parameter name="ACD.geometry.FuelMass"> <ssv:Real value="711.00"/> </ssv:Parameter>
     <ssv:Parameter name="ACD.geometry.PropulsionWeight"> <ssv:Real value="908.25"/> </ssv:Parameter>
     <ssv:Parameter name="ACD.geometry.Sref"> <ssv:Real value="36.02"/>  </ssv:Parameter>
     <ssv:Parameter name="ACD.EngineEfficiency"> <ssv:Real value="4.99"/> </ssv:Parameter>
 </ssv:Parameters>
</ssv:ParameterSet>
```

Listing 3: SSV specifying an aircraft with a maximum climb rate of 70m/s

```
<ssv:ParameterSet>
 <ssv:Parameters>
 <ssv:Parameter name="ACD.geometry.b"> <ssv:Real value="9.32"/> </ssv:Parameter>
     <ssv:Parameter name="ACD.geometry.croot"> <ssv:Real value="4.09"/> </ssv:Parameter>
     <ssv:Parameter name="ACD.geometry.ctip"> <ssv:Real value="0.86"/> </ssv:Parameter>
     <ssv:Parameter name="ACD.geometry.FuelMass"> <ssv:Real value="489.38"/> </ssv:Parameter>
     <ssv:Parameter name="ACD.geometry.PropulsionWeight"> <ssv:Real value="804.7"/> </ssv:Parameter>
     <ssv:Parameter name="ACD.geometry.Sref"> <ssv:Real value="26.12"/>  </ssv:Parameter>
     <ssv:Parameter name="ACD.EngineEfficiency"> <ssv:Real value="4.19"/> </ssv:Parameter>
 </ssv:Parameters>
</ssv:ParameterSet>
```

## 6. Discussion & Conclusion

The framework manages to successfully optimise an aircraft design based on the input variables. It should be noted, however, that the presented design is not meant to be further evaluated in terms of feasibility; the input aircraft data and the designed mission are mere proof of concepts. The results are clear enough to illustrate that the design optimisation routine performs as desired: with increasing mission length (decreasing climb rate), the required fuel and maximum take-off weight (MTOW) of the aircraft increase. Likewise, a decreasing MTOW requires a lower wingspan $b$; this, in turn, impacts the lift, affecting the required thrust and ultimately the engine size.

The proposed framework has been successfully used in the conceptual design of a fighter aircraft, starting from a small set of requirements; however, its potential is further-reaching. The possibility of inclusion of system dynamics early in the aircraft design process opens the door to a design space exploration that is not traditionally performed at such an early stage [2].

Instead of performing concept exploration independently of each other, engineers could assess the impact of their own systems on the overall design, or on any other system of interest, earlier in the design process, provided the required standards are in use and the necessary information is available. The benefits are three-fold. First, this possibility allows for the discovery of potential design incompatibilities between the systems earlier in the design process, when fixing them is not as costly. Second, the continuous integration of system simulation models from an early stage encourages collaboration between disciplines. The signal-based FMU and SSP connection process is an easy, unambiguous way of establishing knowledge maturity about the concerned systems: whenever a system SSP, or subsystem FMU, requires more input than there is available, it cannot be integrated with the existing system models. Finally, the framework allows for early concept evaluations with respect to transients, which are overlooked by traditional, static methods. In the case study it turned out to be that the transients were not the primary dimensioning factor. Even so, the resulting total fuel mass is a result of integrated dynamic behaviour.

One of the strengths of the presented framework is its scalability. There are no inherent limits on neither the size of the models included (both physical and information-wise), nor on their number. The FMI and SSP standards confer a high degree of modularity, enabling exploration of any relevant system and its effect on the overall design, if they are standard-compliant. Moreover, since there is no connection between the inner entities of the different FMUs, there is no requirement either on a matching level of fidelity of the individual system models. If the minimum information required is available (highlighted by the FMUs interfaces), any models can be interconnected. It is the intended use of the framework that dictates what level of fidelity is required from each model. However, in a situation where both detailed models are coupled with ones with a lower fidelity level, the interpretation of the result will be a more challenging task.

A second degree of scalability comes from the physical resources themselves. As each optimisation is a Python process, multiprocessing is a possibility, thus decreasing design space exploration time. The Complex-RF optimisation method also offers parallelisation opportunities for more computationally intensive optimisation tasks [11].

Thirdly, as the framework does not use any proprietary software, there are no limitations on the number of users it can be made available to. This can benefit both industry and academia; the former can extend its exploratory activities to other departments than the ones traditionally concerned with concepts, as there are no licensing costs involved. Moreover, since the framework is written in Python, a popular programming language in the engineering field, the learning slope for the individual engineers should be rather steep. Even if the user is not familiar with the language, due to its popularity there are endless resources available online to understand the basics. On the academic side, including the framework in the engineering curriculum could aid in familiarising the engineering students with tools, methods, processes and standards encountered in the industry, thus bridging the infamous gap between industry and academia.

In order to better investigate the effects of dynamic phenomena on an aircraft concept, the framework could benefit from the inclusion of a six degrees of freedom aircraft dynamics model and more detailed system models. This would allow for more in-depth analysis of the resulting aircraft in terms of stability. The challenge lies, as mentioned before, in deciding on what level of detail is good enough in order to generate representative results with the limited amount of information that might be available on the aircraft.

## Acknowledgements

The presented research was conducted within the frame of the ITEA3 project *EMBrACE* and the NFFP7 projects *Flexible flight simulation framework* and *Digital Twin for Automated Model Validation and Flight Test Evaluation*. The research was funded by Vinnova and Saab Aeronautics. The authors would specifically like to thank professor Peter Fritzson for his support during the early stages of the research. Additionally, the authors would like to thank Lennart Ochel and Arunkumar Palanisami for their help in implementing the necessary OMSimulator SSP and FMI functionality.

## 7. Contact Author Email Address

mailto: alexandra.oprea@saabgroup.com

## 8. Copyright Statement

## References

[1] Carlsson, M., Andersson, H., Gavel, H., and Ölvander, J. Methodology for development and validation of multipurpose simulation models. *50th AIAA Aerospace Sciences Meeting*, Nashville, TN, 2012.

[2] Raymer, D. *Aircraft design: a conceptual approach*, Sixth Edition, AIAA, 2018.

[3] Torenbeek, E., Wittenberg, H. *Flight physics: essentials of aeronautical disciplines and technology, with historical notes*. 1st edition, Springer Science & Business Media, 2009.

[4] Roskam, J. *Airplane design*. DARcorporation, 1985.

[5] Moerland, E., Becker, R-G., Nagel, B. Collaborative understanding of disciplinary correlations using a low-fidelity physics based aerospace toolkit. *4:th CEAS Air & Space Conference*, Council of European Aerospace Societies (CEAS), Linköping, Sweden, Vol. 4, 2013.

[6] Ciampa, P. D., Nagel, B., La Rocca, G. A MBSE Approach to MDAO Systems for the Development of Complex Products. *AIAA Aviation Forum*, American Institute of Aeronautics and Astronautics (AIAA), Virtual, Vol. 2020-3150, 2020.

[7] International Council on Systems Engineering. *Systems Engineering Handbook*. 4th edition, John Wiley and Sons, Inc, 2015.

[8] Box, M. J. A New Method of Constrained Optimization and a Comparison With Other Methods. *The Computer Journal*, Vol.8, Issue 1, pp. 42–52, 1965.

[9] Krus, P., Ölvander, J. Performance index and meta-optimization of a direct search optimization method. *Engineering Optimization*, Vol.45, Issue 10, pp.1167-1185, 2013.

[10] Baer K., Ericson L., Krus P. Robustness and performance evaluations for simulation-based control and component parameter optimization for a series hydraulic hybrid vehicle. *Engineering Optimization*. Vol.52, Issue 3, pp.446-464, 2019.

[11] Braun, R., Krus, P. Parallel implementations of the Complex-RF algorithm. *Engineering Optimization* Vol. 49, Issue 9, pp. 1558–1572, 2017. doi: 10.1080/0305215X.2016.1260712.

[12] Modelica Association. *Functional Mock-up Interface for model exchange and co-simulation*, 2014.

[13] FMI development group. *Tools*. *Online* `http://fmi-standard.org/tools/`. Accessed: 2021-05-31

[14] Modelica Association. *System Structure and Parametrization, Report 1.0*, 2019-03-05.

[15] Köler, J., Heinkel, H-M., Mai, P., Krasser, J., Deppe, M., Nagasawa, M. "System Structure and Parametrization" - Early Insights. *Proceedings of the 1st Japanese Modelica Conference*, Tokyo, Japan, 2016.

[16] Modelica Association Project System Structure and Parameterization. *SSP Related Tools*. *Online*. `https://ssp-standard.org/tools/`. Accessed: 2021-05-25

[17] Ochel, L, *et al.*. Integrated FMI and TLM-based co-simulation with composite model editing and SSP. *Proceedings of the 13th International Modelica Conference*, Regensburg, Germany, pp. 69-78, 2019.

[18] Open Source Modelica Consortium. *OpenModelica user's guide*, *Online*, `https://www.openmodelica.org/doc/OpenModelicaUsersGuide/latest/`. Accessed 2021-05-31.

[19] Fritzson, P. *Principles of Object Oriented Modeling and Simulation with Modelica 2.1*. 1st edition, Wiley-IEEE Press, 2004.

[20] Wetter, M., U.S. Department Of Energy. BuildingsPy. `https://www.osti.gov//servlets/purl/1569219`. Accessed on 2021-05-31.

[21] Hunter, J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, Vol.9, Issue 3, pp. 90-95, 2007.

[22] Modelica Association. *Modelica Standard Library*. `https://doc.modelica.org/om/Modelica.html`. Accessed on 2021-05-31.

[23] Hällqvist, R., Braun, R., Eek, M., and Krus, P. Optimal Selection of Model Validation Experiments: Guided by Coverage. Accepted paper, revision under review at *The Journal of Verification, Validation and Uncertainty Quantification*, 2021.

[24] International Organization for Standardization. *Standard Atmosphere, ISO 2533:1975*. 1975.

[25] Knöös Franzén, L., Hällqvist, R. Hydrogen Aircraft Development Applying Co-simulation with Modelica and FMI. *Center for Model-Based Cyber-Physical Product Development*. `https://conference.ep.liu.se/index.php/MODPROD/article/view/740`, 2021.

[26] Saarlas, M. *Aircraft Performance*. 1st edition, John Wiley and Sons, Inc, 2006.