Working Paper

Aerospace Concept Exploration System

Architecture and Methods for an Air Vehicle Design Tool

Jia Xu, David Merrell, John P. Godges and James S. Chow

RAND Project AIR FORCE/IR&D

WR-1122-RC
October 2015

RAND CORPORATION

Published by the RAND Corporation, Santa Monica, Calif.
© Copyright 2016 RAND Corporation
RAND® is a registered trademark

Preface

The capability to assess aerospace vehicle designs independently, consistently and rapidly can help RAND address complex technological and operational questions with agility, objectivity and precision. We develop the Aerospace Concept Exploration System (ACES) to fulfill this need. In the RAND context, the ACES design tool can serve as the vehicle design component of large-scale end-to-end concept development and system evaluations. This Working Paper details the motivation, theory, applicability and architecture of the ACES design tool.

This Working Paper is a product of the RAND Corporation’s continuing program of self-initiated independent research. Support for such research is provided, in part, by donors and by the independent research and development provisions of RAND's contracts for the operation of its U.S. Department of Defense federally funded research and development centers. The research was conducted within Project AIR Force (PAF) of the RAND Corporation. PAF provides the Air Force with independent analyses of policy alternatives affecting the development, employment, combat readiness, and support of current and future air, space, and cyber forces.

For more information on PAF, see http://www.rand.org/paf/ or contact the director (contact information is provided on the web page).

Comments or questions on this Working Paper should be addressed to the project leader, Jia Xu, at jxu@rand.org.

Table of Contents

Figures

Tables

Abbreviations

$\mathrm{C_l}$	Section lift coefficient
$\mathrm{C_L}$	Lift coefficient
$\mathrm{C_{m\ ac}}$	Pitch coefficient about the aerodynamic center
$\mathrm{c_{wing}}$	Wing chord
$\mathrm{d_{engine}}$	Engine diameter
$\mathrm{h_{fuse}}$	Fuselage height
$\mathrm{l_{fuse}}$	Fuselage length
$\mathrm{l_{to}}$	Balanced takeoff field length
$\mathrm{M_{DD}}$	Drag divergence Mach number
$\mathrm{N_z}$	Load factor in z
$\mathrm{r_{climb}}$	Climb rate
$\mathrm{S_{ref}}$	Wing Area
$\mathrm{t_{wing}}$	Wing thickness
$C_{l\ max}$	Maximum section lift coefficient
T_{sls}	Sea-level static thrust
W_{MTOW}	Maximum takeoff weight
W_{MZFW}	Maximum zero-fuel weight
W_{ZFW}	Zero-fuel weight
W_{engine}	Engine (dry) weight
w_{fuse}	Fuselage width
x_t	Non-dimensional x-location of flow transition
$\Lambda_{1/4}$	Wing sweep at the quarter chord
ACES	Aerospace Concept Exploration System
AAA	Advanced Aircraft Analysis
AR	Aspect Ratio
BWB	Blending Wing Bodies

CEASIOM	Computerised Environment for Aircraft Synthesis and Integrated Optimization Methods
CFD	Computational Fluid Dynamics
CG	Center of Gravity
CL	Coefficient Lift (?)
FLOPS	Flight Optimization System
JASSM	Joint Air-to-Surface Standoff Missile
JSON	JavaScript Serialization Object Notation
M	Mach number
MALD	Miniature Air-Launched Decoy
MATLAB	sequential quadratic programming
MDAO	Multidisciplinary Analysis and Optimization
MDO	Multidisciplinary Design Optimization
MSES	Multi-element Airfoil Design Analysis Software
MVC	Model-View-Controller
NASA	National Aeronautics and Space Administration
OBD	Optimizer-Based Decomposition
OOP	Object-Oriented Programming
PASS	(a design tool?)
R	Range
RAM	Radar Absorbent Material
Re	Reynolds number
SQP	Sequential Quadratic Programming
SUAVE	Standard and Unconventional Aerospace Vehicle Environment
t/c	Thickness to chord ratio
T/W	Thrust to weight ratio
TASOPT	Transport Aircraft System OPTimization
TSFC	Thrust Specific Fuel Consumption
UAV	Unmanned Aerial Vehicle
V&V	Verification and Validation

VLM	Vortex Lattice Method
VSP	Vehicle Sketch Pad
W/S	Wing loading
λ	Taper ratio
Λ	Wing sweep
T	Time
v	Volume
α	Angle of attack
θ	Wing twist

Summary

The capability to assess aerospace vehicle designs independently, consistently and rapidly can help RAND address complex technological and operational questions with agility, objectivity and precision. The Aerospace Concept Exploration System (ACES) is a computational conceptual air vehicle design tool that can answer first-order questions about air vehicle configuration, size and weight. ACES accepts design requirements and constraints as inputs and produces optimized designs as outputs. ACES computes vehicle performance figures using a suite of physics-based and empirically derived models. An efficient gradient-based optimizer drives design convergence with respect to constraints. Ongoing work on the ACES software architecture aims to contain complexity and support future code extensions. Currently, the tool can be used to design vehicles operating in the subsonic and transonic regimes. The intended users of ACES are engineers and analysts who have a basic understanding of air vehicle design and of modeling and simulation. In the RAND context, the ACES design tool can serve as the vehicle design component of large-scale end-to-end concept development and system evaluations.

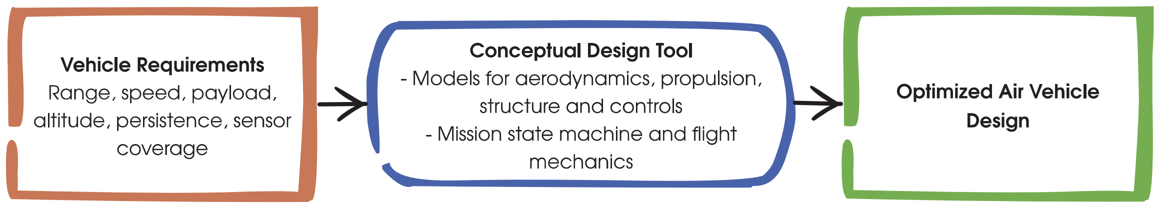
Outline

This paper contains seven sections. First, we describe the motivation, requirements and limitations of the ACES design tool. Second, we discuss the ACES architecture and its constituent models in substantial detail. Third, we address software design issues that arise specifically in the context of aircraft design, but also more generally in multi-physics and multi-fidelity analysis. Fourth, we describe the ACES visualization tools. Fifth, we validate the ACES tool against an established aircraft design. Sixth, we illustrate the basic workings of the ACES design tool through a case study, applying ACES in the development of a new UAV design. Finally, we summarize our conclusions.

1. Motivation, Requirements and Limitations

Conceptual design constitutes the first stage of aerospace design, in which basic questions of vehicle configuration, size, weight and cost are answered. Because of the decisive influence of these questions on requirement and program evolution, conceptual design has been termed the 5% of investment that determines the success of the next 80% of development. Figure 1.1 illustrates the role of a conceptual design tool: to synthesize converged vehicle designs based on mission requirements.

Figure 1.1 The inputs, outputs and logic of an conceptual design tool.


Vehicle Requirements
Range, speed, payload, altitude, persistence, sensor coverage
Conceptual Design Tool
- Models for aerodynamics, propulsion, structure and controls
- Mission state machine and flight mechanics
Optimized Air Vehicle Design

The objective of conceptual design is often as much about the technical design as it is about understanding the performance, cost and schedule impact of varying requirements and constraints. At RAND, ACES can be used to support force planning and recapitalization under changing global circumstances by providing the crucial link between vehicle design, performance assessment and cost estimation. Specifically, it can be used to credibly map requirements to designs, whose operational effectiveness and costs can then be analyzed in broader systemic studies. Example applications could include analyses of alternatives for next-generation tankers, long-range standoff weapons and future unmanned surveillance platforms. More specifically, the design tool can help RAND to:

1. Rapidly and consistently estimate air vehicle performance
2. Design vehicles to mission objectives and constraints
3. Assess the impact of new technologies and operational paradigms
4. Bridge the gaps between vehicle, mission and campaign-level analysis.

An independent design tool, for which we have source level access and control, has both the bandwidth and flexibility to support highly integrated, end-to-end studies. But independence should not require reinventing the wheel (or wing). We build our design tool on a judiciously selected set of models from the aerospace literature. Where we seek to innovate is to apply contemporary software design and development practices to improve the extensibility and maintainability of the tool. A second, associated methodological thrust of ours is to integrate contemporary open source scientific computing tools into the RAND workflow.

Requirements

The ACES tool is built with four top-level requirements in mind. The requirements are aligned with the tool's anticipated use case at RAND.

First, the tool should be agile. This means it should be fast, flexible and scalable to deal with problems of diverse size and scope. Speed is crucial for providing interactivity to engage the user and to adequately support large-scale parametric sweeps. Planners are interested in questions of capabilities, costs and technologies: Can a vehicle be designed to do *x*? If not, then what technology or combination of technologies will be needed to enable *x* in the future? This calls for a tool that is fast, flexible and scalable enough to support iterations between operational and platform concept development.

Second, the tool should be integrated. Aerospace design is inherently multidisciplinary. The non-linear nature of flight physics and its compounding impact on weight and fuel consumption means that converged intermediate-fidelity designs can be more valuable than individual high fidelity, but mutually uncoordinated discipline analyses. Converged designs that respect mission requirements are particularly relevant at RAND, where vehicle design is typically but one component of large-scale "end-to-end" system analyses.

Third, the tool should be rational. The analysis should, where possible, be grounded in physics. Empirical models that are distilled from test results and the performances of existing vehicles are often faster to run and easier to implement, which is why many conceptual design tools are based heavily on empirically derived fits. Empirical models also have the advantage of real-world validation at the specific design points from which they are derived. However, this also means that if the design points move away from well-trotted territory – as they often do in unconventional designs and parametric studies – then the empirical models could rapidly lose accuracy. In this sense, empirical models are biased toward what has been built and do not clearly separate the physics of "what can be built" from the reality of "what has been built". Physics-based models, while often more complex to implement and subject to errors due to model reduction, have the potential to lend generality and sensitivity over larger portions of the design space. The modeling strategy in ACES is therefore one of physics where possible, empiricism where necessary.

Finally, the tool should be tractable. The only invariance in the development of computational tools is the constant need to change and adapt to deal with new problems. The design tool must be built for extensibility and adaptability. Furthermore, the aforementioned multidisciplinary and coupled approach to aerospace design means that the complexity of the design tool will, by definition, always grow much faster than that of an uncoupled and discipline-specific analysis. This has two implications: 1) It is imperative to contain the complexity of the design process and software and 2) it may be necessary to limit the number and complexity of the physics models and, by extension, the design configurations and flight regimes assessed.

The Need for a New Design Solution

Prior to committing to building an in-house solution, we reviewed the existing computational aircraft design tools (in Table 1.1) to see if any satisfied our requirements.

Table 1.1 Comparison of contemporary computational conceptual air vehicle design tools.

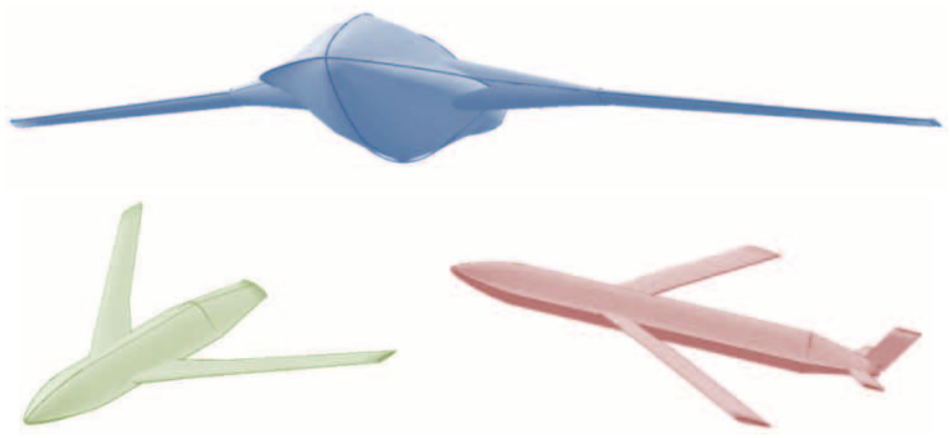
	Developer	Physics Content	Source Access	Language	Actively Developed
FLOPS	NASA	Low	Yes	Fortran	Yes
TASOPT	MIT	Medium	Unknown	Fortran	Unknown
CEASIOM	EU FP7 Program	Medium/High	Limited	Matlab	Unknown
PASS	Desktop Aeronautics	Low/Medium	No	Java	No
RDS	Conceptual Research	Low	No	Object Pascal	Yes
AAA	DarCorp	Low	No	Object Pascal	Yes
VAMPZero	DLR	Low	Yes	Python	Yes
SUAVE	Stanford	Low/Medium	Yes	Python	Yes

Many of the tools listed in the table share similar limitations. First, the source codes are generally not available. While most of the tools listed in Table 1 have been built with parametric design and, in some cases, optimization in mind, access to the source code is still necessary to expand the modeling capacity of the tools and hence, the types of problems that they can address. VAMPZero (Böhnke et al., 2013) and SUAVE (Lukaczyk, et al., 2015) are notable open source exceptions. NASA also distributes FLOPS, including the source code. All other tools are closed source. Furthermore, of the tools whose source codes are available, only VAMPZero and SUAVE are known to conform to modern software design principles and utilize modern programming languages. Given the varied nature of the problems that RAND is often called on to address, the design tool must be made rapidly reconfigurable and extensible. This makes source level access and sound software design essential attributes.

Second, reliance on empiricism renders many of the existing tools insensitive to certain design parameters and constraints, particularly in the design of unconventional vehicles. Only CEASIOM (Böhnke et al., 2013), TASOPT (Drela, 2010 and Greitzer, et al., 2010) and, to a lesser extent, SUAVE contain substantial physics-based analysis. The development of physics-based design tools is challenging and requires careful verification and validation. Parts of ACES, such as the weight and engine models, still rely substantially on empirical models, while other components such as the aero-structural calculations are physics-based, and of comparable sophistication to leading design tools. More importantly, the architecture of the design tool is built with extension in mind to accommodate higher-fidelity, physics based methods in the future.

The absence of a generally acceptable solution thus motivates the development of our tool. Still, our design tool does not exist in a vacuum. We strive to integrate the best practices and architectures from existing tools. We also recognize that open-source developments have the potential to overcome the structural limitations of "boutique" aerospace design tools: small user bases and proprietary, unverified methods. We do not therefore preclude the possibility of relying on open-source developments, or even merging our effort into such a project in the future. Indeed, as a preview of what lay ahead, Figure 1.2 portrays some of the outcomes generated by ACES using a Python wrapper for the open-source NASA OpenVSP (Waddington and McDonald, 2015) aerospace visualization environment.

Figure 1.2 Example UAV visualizations generated using the ACES tool, specifically the Python-based link to the NASA OpenVSP geometry and visualization engine.



Limitations

Currently, the models in ACES are applicable to the design of air vehicles operating in the subsonic to low transonic flight regimes (Mach 0.9 or less). This reflects the transonic, commercial aircraft design heritage of many of the published methods employed in ACES.

The general non-linear partial differential equations that govern fluid flow require sophisticated numerical methods to solve. Flight regime–specific simplifications of the governing equations are generally necessary to obtain solutions at reasonable computational costs. These simplifications, such as the potential flow reductions of the Navier-Stokes equations, render our models applicable mostly to the subsonic and low to intermediate transonic flight regimes.

A simplified supersonics aerodynamic module does exist, but it assumes optimal volume and lift wave shaping. Investments in higher fidelity methods, from equivalent body methods to computational fluid dynamics (CFD), are needed to achieve sufficient design confidence in the supersonic regime. The relative complexity and computational cost of some of these solutions make them hard to justify for the present application. Similarly, the aerodynamic models in

ACES do not yet account for three-dimensional-boundary layer effects, or detached vortices at high angles of attack. The practical implication is that supersonic fighters with delta or cranked delta wings cannot yet be analyzed with great confidence.

Analogous simplifications exist in the structure and propulsion models. The equivalent-beam wing structural model, while physics-based, is geared for the analysis of high-aspect-ratio configurations.

As for engine types, only turbofan and turbojet engine performances are currently modeled (in the form of semi-empirical "rubber engines"). However, an analysis of comparable levels of fidelity can be readily extended to open rotor and variable cycle engines. Provisions have also been made to support future expansion of the models to include electric and hybrid engines.

The absence of detailed component information and geometries in conceptual designs necessitates a pragmatic approach to analytic fidelity. It must be emphasized that the goal is not to conduct high-fidelity design or shape optimization but rather to map out the potential *boundaries* of what can be achieved with (presumably) well-executed designs.

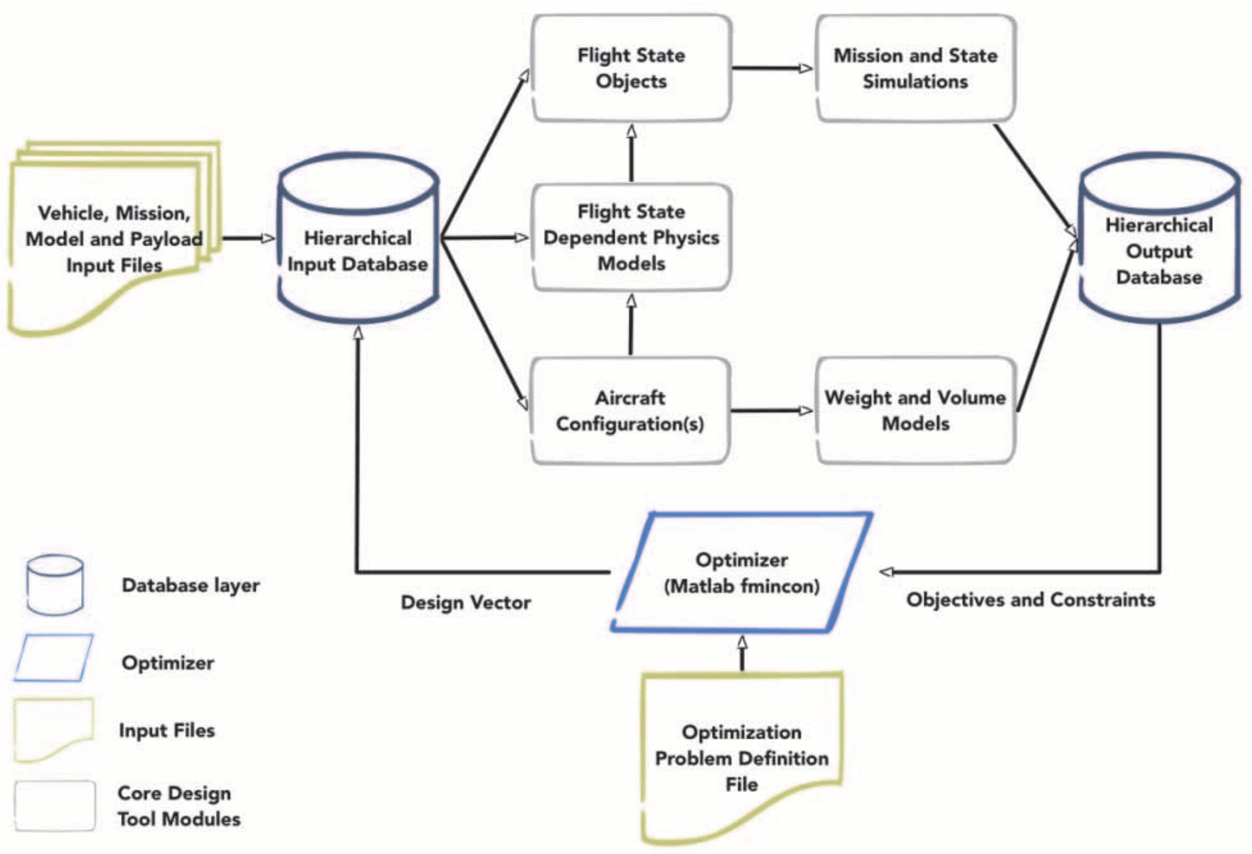
Finally, ACES currently does not contain any general flyaway or lifecycle cost models. This is a significant omission given the tool's likely use case at RAND, but one that can be addressed on a case-by-case bases depending on the system in question.

2. ACES Architecture and Models

Formal optimization drives the design process in ACES. The user is expected to (1) define the vehicle, mission, modeling assumptions and payloads in a series of input files, (2) flag which inputs are design variables in the optimization and (3) define the objective and constraints in terms of both the inputs and outputs of the analysis. Some of the parameters defined in the input files will be fixed. The user can mark any of the input parameters as optimization variables, whose ultimate, converged values will be determined by the optimization process. The optimizer evaluates the fitness and constraint violations, changes the design vector values and starts another run. Typical objectives might be to "successfully execute the design mission with the lowest possible vehicle empty weight." The details of this optimization process and the layer architecture to enable it are discussed in the text.

The ACES design tool is constructed of three “layers” of individually usable components. Working from the center outward in Figure 2.1, they are the core analysis modules (gray rectangles), database layer (blue cylinders) and optimization (blue parallelogram) layers. The user and other program interact with ACES through the input files (olive folder).

Figure 2.1 ACES architecture diagram


Flight State Objects
Mission and State Simulations
Vehicle, Mission, Model and Payload Input Files
Hierarchical Input Database
Flight State Dependent Physics Models
Hierarchical Output Database
Aircraft Configuration(s)
Weight and Volume Models
Database layer
Optimizer (Matlab fmincon)
Design Vector
Objectives and Constraints
Optimizer
Input Files
Optimization Problem Definition File
Core Design Tool Modules

The core modules (gray boxes) combine aircraft configurations, physics models, and mission and state definitions to produce performance results. Here the analysis steps are "open loop" in that no attempt is made to iterate and reach converged, self-consistent designs. We simply define the aircraft and fly it, noting any performance and configuration constraint violations along the way. The optimization later, discussed subsequently, is used to solve the full nonlinear constraint satisfaction problem. In the core modules we maintain strong separation among the aircraft configurations, physics models and mission simulations. This can be considered an application of the Model-View-Controller (MVC) architectural pattern (Fowler, 2002), with the following one-to-one correspondence:

- **Model**: The aircraft configuration, made up of a collection of physical components: wing, tail, fuselage, engine, payload etc.
- **View**: Different physics models, which compute forces, moments and weights based on the aircraft and its state. The models are not ontological: Multiple physics models, with varying degrees of fidelity and locality, and different input spaces, can be used to compute the same performance parameters. Ideally, the user should be able to switch models with little or no code modifications.
- **Controller**: The state machine, which mediates the interaction between the model and views. It mediates between the models (aircraft configurations) and views (physics models) to produce performance results.

By explicitly separating the configurations, physics and simulations, we aim to add flexibility and extensibility to the problem formulation. In principle, this type of modularity can also help enable multi-fidelity modeling. That said, the authors acknowledge that the ability to fully leverage multi-fidelity models, particularly with different parameter spaces, remains a challenging research problem. While the ACES tool can certainly benefit from sound software design patterns and practices, the integration of multi-fidelity and multi-physics models is also distinct from the extensibility problem typically encountered in software design, and is unlikely to be resolved through good software design alone.

The hierarchical input and output files (olive folders) define major components: wings, fuselages and engines. Other important modeling assumptions and payload configurations are defined in their own input files. The user interacts with the tool is primarily through the various input files. For example, the user can define the configuration of the vehicle, the different states that compose the mission, the various modeling assumptions that underpin the physics, the corner states that help to size the aircraft and the payload configuration of the aircraft. In the optimization definition file the user can further flag any of the parameters defined in the input files as optimization variables. They can also flag any expected output of the analysis as optimization constraints. This way the user has complete control over the formal definition of the optimization problem.

The database layer mediates between the optimization and simulation. This layer acts as a shared, hierarchical namespace that allows all vehicle and mission parameters to be designated, without intrusions into the code, as optimization constraints or objectives.[1]

Finally, the optimization layer (blue parallelogram) operates on the parameters in the database layer to reach the best objective and satisfy the constraints. The optimizer manipulates the parameters linked in the database and treats the aircraft performance and simulation layer as a black box. After each run of the core modules the optimizer evaluates the results in terms of the goals of the optimization, changes the variable parameters and starts another run. Typical goals might be, "successfully execute the design mission with the lowest possible vehicle empty weight." The details of this process are discussed in the text.

Communications among the three layers are conducted through well-defined APIs and standard-compliant JSON (JavaScript Serialization Object Notation) files. The input files fully parameterize the design and mission states, and render designs and outcomes reproducible.

The remainder of this section will discuss the ACES component models individually.

Aircraft Configuration

The first step in the analysis is to parametrically describe the vehicle, as indicated by the “aircraft configuration” rectangle in Figure 3. This includes the geometric properties of the wing, fuselage, empennages, control surfaces, engines and other physical components. The user can specify any or all of the parameters via input files. They can also designate, without intrusions into the code, any of the parameters as design variables in the optimization.

The lifting surface (wing and empennage) geometric properties of chord, sweep, twist, t/c and wing box are defined at arbitrary numbers of spanwise stations. The intermediate spanwise geometries are linearly interpolated. The geometric parameterization is flexible enough to represent leading and trailing edge-wing extensions, multi-cranked wings and flying wing configurations.

The fuselage class currently represents tubular configurations, using a composition of interchangeable nose, tail, constant section and cross section objects. The presentation is sufficiently general to accommodate unconventional configurations such as flying wings and blending wing bodies (BWB). The geometric objects in the composition also contain functions to compute useful volumetric and surface geometry properties.

[1] The use of an intermediate abstraction layer between the optimizer and the simulation is similar to the implementation of the PASS design tool within the Caffe framework (Antoine and Kroo, 2005). It can also be considered as a simple, lightweight version of the common parameter and namespace management practices of more sophisticated multi-disciplinary design environments like OpenMDAO (Gray, Moore and Naylor, 2010). The intent is to define a minimalist interface for the user to define optimization and other types of batch problems, not to duplicate the capabilities of dedicated multi-disciplinary design optimization (MDO) integration tools.

Physics Models

The physics models in ACES draw heavily from established low-to-moderate fidelity methods detailed by Kroo (2001), Drela (2010), Shevell (1989) and Nicolai and Carichner (2010). The methodological discussions in this report are high-level. We encourage readers to refer to the original works for details on specific methods.

There are two classes of physics models in ACES: those that depend on the instantaneous flight condition (altitude, Mach number and atmospheric properties) and those that do not. The flight-condition-dependent models include the aerodynamics, load and propulsion elements. These models are injected into the state machine and updated at run time. The second group of models, invariant with the flight condition, includes the structure, weight and volume models. The division between flight-condition-dependent and independent models is subject to change as the design tool continues to evolve. We will now discuss each physics model, in turn.

Aerodynamics

The aerodynamics model provides smooth, low to intermediate-fidelity estimates of lift, drag and longitudinal stability derivatives for mission simulations. The lift balance in equilibrium flight can be solved at either a prescribed lift coefficient CL or an angle of attack. The drag force at a given CL is built up using the methods described in Kroo (2001) and Shevell (1989).

We use a modified Weissinger lifting-line method to compute the wing spanwise lift distribution. Integration of the Trefftz plane normal-wash yields the induced drag. The Weissinger method is a simplified vortex lattice method (VLM) in which spanwise stations are represented using single chordwise panels (Wakayama, 1995). Omission of chord-wise panels limits the types of forces and moments that the method can resolve but allows for faster computation. Our implementation is based on the method outlined in Bertin and Smith (1998) and includes the Prandtl-Glauert compressibility corrections in the aerodynamic influence coefficient matrix to improve lift estimates at moderate transonic conditions.

The Weissinger method is fast, physics-based and, in addition to being able to estimate induced drag for arbitrary planforms, can also resolve the wing spanload. The spanload, in turn, allows for more physical estimates of the wing maximum lift limit (via the so-called method of critical sections), the aerodynamic center, and the spanwise wing shear force and bending moments. All of these sensitivities strengthen the links among the structural, aerodynamic and control aspects of the design problem.

The parasite and compressibility drag of the wing and tail aerodynamic surfaces are integrated from 2-D drag estimates at a number of spanwise stations, using the local and local-normal component of the Mach number and wing geometric properties. This is an application of the so-called “strip” or “blade-element” theory of applied aerodynamics (Sequeira, Willis and Peraire, 2006, and Mariens, 2010).

The sectional profile drag is estimated based on flat plate skin friction coefficients linked to a user-prescribed transition location, corrected by an empirical form factor to account for super-velocity induced by the surface geometry:

$$C_{d_p}(y) = f(\mathrm{h}, \Lambda, c, \frac{t}{c}, M, Re, x_t)$$

Which can be integrated along the span to obtain the wing profile drag:

$$C_{D_p} = \frac{2}{S_{ref}} \int_0^{b/2} C_{d_p} c \, dy$$

The wing component of the compressibility drag is assumed to dominate. This assumption works well below Mach 0.9, beyond which significant pockets of supersonic flow could develop over the fuselage (particularly near corners and cockpit windows) and produce meaningful compressibility drag increments. The wing sectional transonic compressibility drag is estimated on the basis of a modified Korn relationship, adjusted by Torenbeek to reflect the performance of more recent supercritical sections (Torenbeek, 2013):

$$C_{d_c}(y) = f(\Lambda, C_l, \frac{t}{c}, M, \kappa_t)$$

The exposed section compressibility drag estimates are integrated to obtain the wing compressibility drag. The modified Korn equation is simple to use but highly empirical. Implicit is our assumption that a properly designed aircraft would not experience an excessive amount of compressibility drag. Imprecise accounting should therefore not significantly bias the optimal design.

We include maximum lift constraints using the so-called "method of critical sections": The wing is assumed to stall if any wing station along the span reaches its local maximum lift coefficient. Here, the station lift coefficient comes from the Weissinger formulation. The station maximum lift coefficient is estimated based on the flight condition (specifically, the Mach and Reynolds numbers), the sectional thickness and the station flap and slats deflections (Wakayama, 1995):

$$C_{l_{max}}(y) = f(Re, \frac{t}{c}, Mach, \delta_f, \delta_s)$$

A design margin (typically 0.2) can be imposed over the outboard wing stations to preserve control authority at high lift conditions. The estimated sectional maximum lift, like the compressibility drag estimates, are highly empirical and cannot therefore be expected to predict arbitrary airfoil section performance with great confidence. The relationship does, however, capture the basic physical variation of maximum lift with Mach and Reynolds numbers, with significant implications for wing sizing.

One advantage of the quasi-3D strip theory decomposition is that airfoil data of arbitrary fidelity can be used – from panel methods and Euler solutions matched with interactive boundary layers (e.g. XFoil and MSES) (Drela, 1989, 2007) to CFD – so long as the sectional force coefficients can be related, via intermediate surrogate models, to the instantaneous operating conditions. Indeed, Drela introduces CFD-based airfoil surrogate models into the TASOPT design tool (Drela, 2010) as a way to bring more physics into the wing design problem. ACES, while adopting a more empirical approach, likewise makes provisions for the integration of higher-fidelity airfoil data, if they are available.

However, it should be said that as the sophistication and, often, modality of the airfoil surrogate model increases, so, too, do the number of potential local minimums. Examples could include transitional airfoils with significant laminar drag buckets. Furthermore, the sectional surrogate models themselves must be built against a potentially large number of flow and geometry variables. All of this diminishes the relative computational efficiency of the pseudo-3D strip-theory approach to fully-fledged 3D flow solvers. Care must then be taken to balance the complexity and modality of the sectional aerodynamic models and the cost of building sectional surrogate models against the consistency of converged air vehicle solutions. Convex models that are created from sparse sampling strategies and that manage to capture the essential physics of specific airfoils (Hoburg and Abbeel, 2015), but that still work well with gradient-based optimization schemes, may offer a balanced way forward.

The fuselage profile drag is estimated in an analogues fashion using flat plate skin friction coefficients and empirical form factors for axis-symmetric bodies (Ohad, Mason, and Schetz, 2010). The form factor from Shevell (1989) is sensitive to the fineness (length to diameter) ratio of the body and the Mach number. An equivalent diameter and fineness is used to represent non axis-symmetric bodies. The profile drag of nacelles and other faired bodies are computed in a similar fashion. In the case of nacelles, the form factor is modified to account for flow through jet wakes, which can increase the apparent fineness ratio.

Finally, the accounting of the aircraft viscous lift-dependent drag follows the method outlined by Kroo (2001). The viscous component of the lift-dependent drag depends on an empirical (and constant in the case of turbulent transonic vehicles) viscous lift-dependent drag parameter κ_v and is proportional to the profile drag:

$$C_{D_i}(viscous) = \kappa_v C_{D_p} C_L^2$$

Propulsion

In conceptual design, we may not have access to the full flight condition-sensitive engine model – colloquially called an engine “deck”. This is certainly true if no engine has been selected or designed for the vehicle application at hand. Moreover, existing engine models supplied by the manufacturer may not be smooth or complete enough to support design optimization. Such detailed engine decks typically include aircraft-specific operational and maintenance optimizations and constraints (such as engine flat rating) that make them less representative of the underlying physics. A reduced fidelity “rubber engine” that can smoothly approximate existing engine performance across a larger portion of the design space is useful for conceptual design.

The engine weight scaling in the propulsion model interfaces with the aircraft weight model to be discussed below. The engine is scaled based on a required mass flow rate or a required thrust at a reference condition (often the sea level static thrust condition: t_0). The propulsion model produces numerically smooth fuel flow and thrusts for a given bypass ratio. The engine weight and volume then scale with the required sea level static thrust and the bypass ratio.

This model can account for the effects of engine bleed and direct power generation. It can also account empirically for the losses from propulsive integration and inlets. It should be acknowledged that aero-propulsive integration and thrust-drag bookkeeping remain challenging problems, particularly for conceptual design.

Structure, Weight and Volume

ACES uses a simple structure and load model from Kroo (2001) to estimate the wing and fuselage weight.

The wing and empennage structural models assume a fully stressed, metal-based, semi-monocoque wing structure. The load-bearing material is computed to take the dominant bending load at limit conditions. An empirical factor from Gallman (1993) is then used to account for secondary structures to distribute and carry torsional loads, prevent buckling and account for non-load bearing weight. Technology factors can be used to model the weight-reduction effects of advanced composites.

One limitation of the simple wing structure model discussed above is its assumption of elliptic lift distribution over the lifting surface. ACES also contains more sophisticated model that uses the load resolved by the Weissinger vortex lattice method to compute spanwise stress constraints, which in turn size the wing. This model renders the wing weight sensitive to loads and planform configuration and therefore better integrate aerodynamic with structural design. It can also more directly model the averaged material property of composite structures.

The fuselage weight is estimated on the basis of ultimate load factors and an assumed distribution of fuselage and internal system weight. The fuselage can also be sized to take pressurization loads.

The engine weight scale behaves according to empirically derived thrust-to-dry-weight. Other aircraft component and system weights are also estimated based on empirical models found in Kroo (2001), Torenbeek (2013) and Nicolai and Carichner (2010). These include such items as the landing gears, hydraulic systems, electrical and power systems, and surface treatment for survivability. Owing to the overriding importance of weight in air vehicle design, the empirical component weight models represent perhaps the greatest source of uncertainty and error.

The volume model tracks the usable internal volume of the wing and fuselage components. Fuel volume plays an important role in vehicle sizing. The user can specify if the mission and reserve fuel have to be contained within the fuselage, the wing box or a combination of both. An interface exists to represent internal fuel volume and integrate it with the weight and CG calculations. Component volumes, including that of the engine, inlet and ducts, are accounted for using simplified modes. The optimizer ensures internal payload, systems and fuel volume compatibility.

Mission and State Simulations

The mission simulation component of the ACES architecture follows a state machine pattern. A mission is defined in the design tool as a collection of states (takeoff, climb, weight, fuel, altitude continuity). A “state,” which represents an instantaneous force balance, is the fundamental unit of the pseudo-static performance analysis. A “segment” is a computational construction that accepts two or more states and an integrator to compute performance over time.

The mission is solved implicitly as a collocation problem with respect to weight. The control points of the collocation problem are a series of user-defined flight states that define the mission in sequence. The optimizer treats the aircraft fuel fraction (and therefore the instantaneous weight) at each state as design variables while simultaneously constraining the change in fuel weight in between any two successive states to be sufficient to cover the distance and/or flight time of the said segment. This can be considered an application of optimizer-based decomposition.

The aircraft is assumed to transit linearly between the flight states (with linear variations in weight and aerodynamic properties). This simple model is accurate in cruise and loiter situations, where the altitude change is moderate. The assumption of linear variations in aerodynamic properties becomes increasingly tenuous, however, when there are significant altitude changes (such as climb). In such situations, the accuracy can be improved by increasing the number of collocation points, with an associated increase in the numbers of variables and constraints and, hence, the computational cost.

In addition to the collocation-based mission analysis, the user can also specify other flight states at which to evaluate aircraft performance. This approach is useful for specifying “corner” conditions in the flight envelope that are not associated with specific missions, but can nonetheless size the vehicle. Examples of such conditions include high-speed dash, maneuvers, takeoff and recovery and ceiling conditions.

Since each flight state can be explicitly specified, all mission and flight state parameters can be treated optimization parameters. The user can choose to optimize cruise speed, lift coefficient and/or altitude concurrently for the mission, as well as required flight states.

Optimization-Driven Convergence

In discussing aerospace design, Torenbeek draws the important distinction between design “analysis” and “synthesis” (2013). The former refers to the open-loop evaluation of the performance; the latter describes the closed-loop generation of self-consistent, or “converged” designs that respect a set of constraints. We are here concerned with the latter priority of synthesis, which is achieved in ACES by means of formal optimization.

For example, the iterative solution for the aircraft weight in traditional design tools is absent in ACES. Instead, we employ optimized-based decomposition (OBD) (Martins and Lambe, 2013) to pose the iterative weight convergence problem as a design variable (the target weight) and an optimization constraint (the target weight has to be greater than the computed weight). The weight is converged at once with all other aircraft design parameters and mission states. Analogous decompositions are used throughout to allow complex, interdependent physical relationships to be posed explicitly and without local iterations.

ACES uses a gradient-based, sequential quadratic programming (SQP) optimizer (MATLAB fmincon) to drive the design toward optimality and constraint satisfaction. In principle, any continuous domain optimization tool that can handle non-linear constraints can be deployed, including global and population-based methods like genetic algorithms and particle swarms. However, like all search problems, there is an implicit coupling between the problem formulation and the best optimizer for the job (Wolpert and Macready, 1997). Practically speaking, the more restrictive a problem formulation, the more it stands to gain from specialized optimizers.

Gradient-based optimization requires the objective and constraint functions to be numerically "smooth". This requirement naturally constrains the types of physics that can be modeled within the design code. Yet this cost in flexibility is justified because gradient-based methods can scale efficiently against problem size (e.g. the number of variables and constraints) and can achieve fast, provable local convergence even when dealing with complex, high-dimensional problems. A typical ACES design problem, with 30 design variables and 800 constraints can be solved in 2-3 minutes on a contemporary laptop.

While more specialized convex optimizers that can be faster still, they further restrict the physics models to both smooth and convex functions. The requirement for convexity greatly complicates important aspects of aerodynamic and structural modeling, particularly in the case of discretized solutions of differential equations like the Weissinger panel method used in ACES.

The smooth but non-convex problem formulation in ACES is therefore an attempt to strike the right balance between computational efficiency and modeling flexibility. The combination of smooth models and gradient-based optimization means that converged aircraft will take orders of magnitude less functional evaluations than global optimization schemes such as genetic algorithms. This is especially true as the problem grows to include more design variables and constraints. One important caveat is that the non-convex problems solved in ACES will generally converge to local, rather than global, minimums. Optimizations restart are therefore needed to ensure convergence to the global optimum. In principle the need for restarts would add to the effective computational time. In practice, it was found that most problem formulations converge quite reliably to the same design over multiple restarts, which suggests some level of convexity in the underlying models.

3. Software Design to Contain Complexity

The ACES design tool is written largely in MATLAB. Other, more general programming languages like Python and Julia were also considered. We write the core application in MATLAB primarily to take advantage of its mature non-linear optimization package (fmincon). The open-source scientific Python stack is used extensively for post-processing and visualization, where its wealth of libraries, more expressive syntax and advanced language features can add value. Regardless of the initial language choice, investments in building sound software architectures, data structures and automated tests will carry over to different languages.

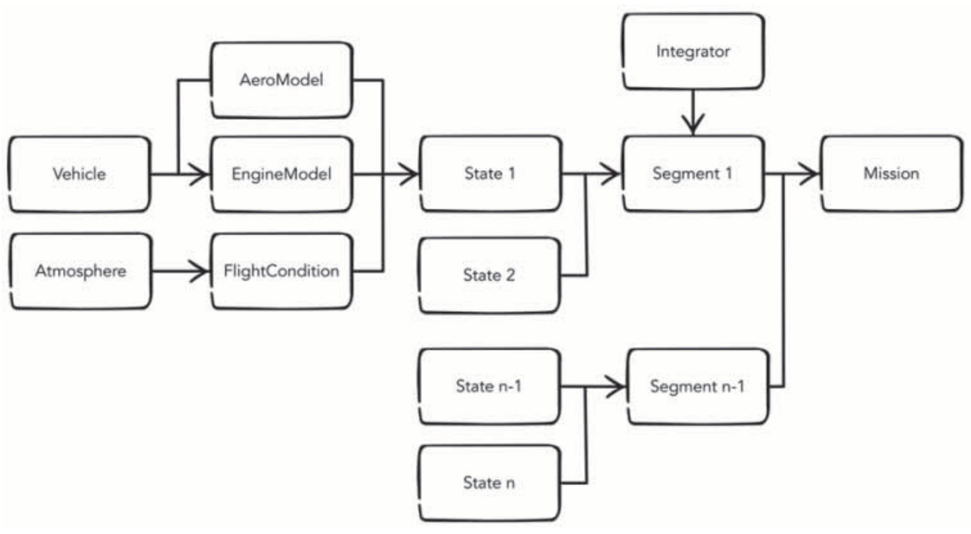
The design tool is written in a multi-paradigm form. High-level interfaces leverage object-oriented programming (OOP) practices. Back-end computational modules are more procedural and functional. This is because the high-level description and manipulation of flight conditions and aerodynamic model are highly stateful and share many states; they stand to benefit significantly from OOP abstractions. Back-end computations of forces are more compartmentalized, rely on fewer shared states, and can be more succinctly posed as functions.

In the OOP implementations, we follow the contemporary pattern of composition over inheritance (Freeman, Eric, et al., 2014). Many hierarchical relationships in the descriptions of the aircraft configuration can, in principle, be efficiently represented using inheritance. However, deep inheritance poses problems for long-term extensibility: upstream changes can impact downstream, derived classes. To cope, we use inheritance relationships (a *is* b) sparingly and rely on composition (a *has* b) where possible. While this leads to a more verbose codebase with a larger number of pass-through functions, it simplifies the architecture and potentially improves extensibility.

The architecture also makes heavy use of dependency injection, in conjunction with composition, to improve modularity and separates, via interfaces, the logic of aircraft design from its detailed implementations. This approach, motivated by software engineering principles, is also consistent with the goal of seamlessly supporting physics of varying complexity.

As a concrete example, dependency injection is used extensively in the composition of aerodynamic and engine models and onwards to the construction of the state and mission objects. Figure 4 illustrates some of the layers of dependency injection needed to construct the executable mission object.

Figure 3.1 State and composition mission



In Figure 3.1, the State objects are the fundamental units of simulation. In keeping with the variable complexity and variable fidelity design goals, each state can be injected with unique configurations and physics models of the user's choosing.

The physics models consume the Vehicle objects in Figure 4 so that they can estimates forces and other dynamic, flight-condition-dependent parameters. The vehicles injected into each model and, by extension, each state within a mission can be different. This architecture readily supports configuration and geometry changes at different points in the mission. The canonical use case may be changes in control surface deflections. Other use cases can include morphing aircraft concepts like the one shown in Figure 3.2 or variable geometry designs with distinct high- and low-speed wing sweeps.

Figure 3.2 NASA morphing wing concept.

Source: NASA

The FlightCondition objects in Figure 9 hold the instantaneous dynamic conditions under which the vehicle is operating, such as speed, load factor and altitude. Each FlightCondition can accept its own Atmosphere object, which produce atmospheric properties at the specified altitudes. This composition allows us to support different atmospheric models in the same mission simulation. Possible use cases for this level of flexibility are for modeling vehicles that take off in tropical atmospheres at sea level and then cruise at standard daytime atmospheres at altitude. Flight-condition-dependent aerodynamics and engine models are then updated according to the flight condition at runtime.

Finally, the authors acknowledge that all software architecture and design patterns are themselves a form of complexity, and carry costs. The construction and application of design patterns are also heuristic by nature – as such, it is generally not possible to prove a "best" architecture for a given problem. Consequently, we do not make any claims that the ACES architecture is somehow "best" – only that it applies certain "best practices" to try to cope with complexity and change.

4. Post-Processing and Visualization

ACES includes a suite of visualization tools, enabling users to understand, communicate, and potentially diagnose designs. The visualization tools are standalone and operate only on the ACES input and output JSON files. The visualization tools are bundled together as a Python package and include interactive Jupyter (iPython notebook) interfaces, which operates as a literate programming environment and a thin graphical user interface. The remainder of this discussion covers four key aspects of post-processing and visualization: the state inspector, the constraint plotter, the design vector plotter, and the python link to NASA's design visualization tool.

State Inspector

Once ACES has produced an optimal design, it may be useful to know how that design performs over its set of missions. Since a mission can be thought of as a sequence of states experienced by the aircraft, this information can be presented on a state-by-state basis. The state inspector reads a design's summary output file and generates plots from the computed aerodynamic, engine and other flight physics data. Examples of this data include components of drag, the spanwise coefficient of lift, and the spanwise divergence Mach number distribution. For each of these data, an array of plots is generated, with each plot corresponding to a flight state.

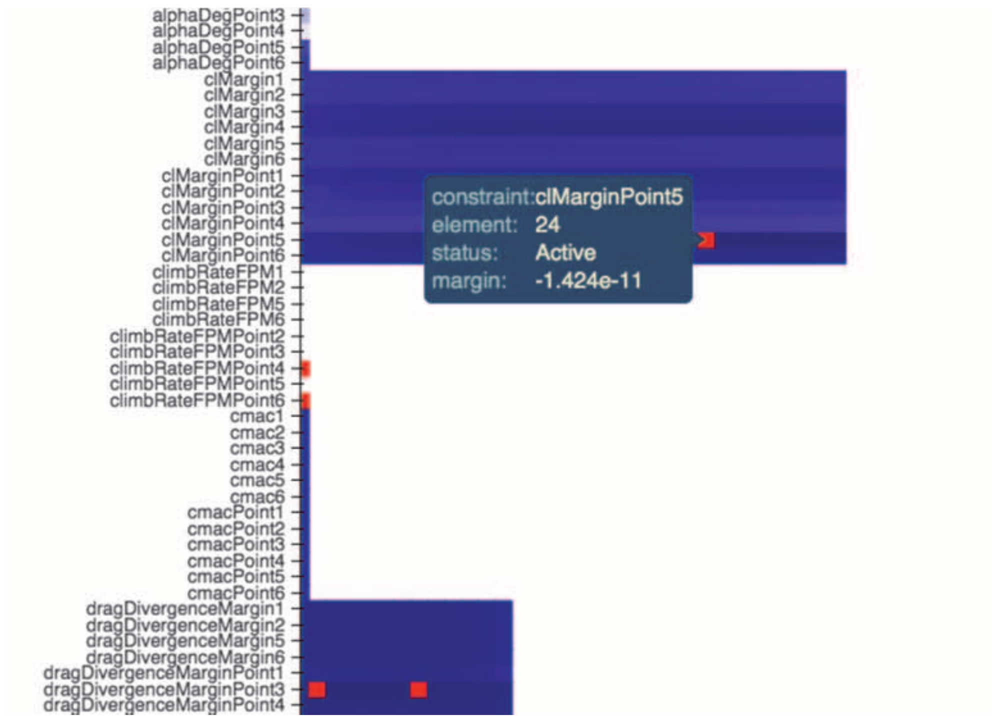
Constraint Plotter

Vehicle design is often subject to a multitude of constraints. It is important to understand which constraints are active and, therefore, driving the design. The constraint plotter provides such a function.

The constraint plotter reads the ACES optimization output file, which contains information about the constraints at play in the optimization process. It then plots a grid of boxes corresponding to the constraints. The boxes' colors are determined by whether or not their corresponding constraints are active; active constraints will have red boxes, while inactive constraints have blue boxes. The blue boxes are further shaded by their proximity to being active, with a darker shade implying that the constraint is closer to being active.

An additional interactive layer allows the user to hover over different constraint boxes with a cursor and bring up additional metadata. This way a user can take a quick glance at the plot, hover over any red boxes, and learn more about the active constraint in question. An example output from this tool is shown in Figure 4.1. The example shows that both scalar and vector constraints can be visualized in rows. Horizontal blocks represent vectors of related constraints. In the figure, the vectors relate spanwise constraint margins, such as maximum lift and drag divergence, at different flight states. The combination of high density and interactive hover provides global perspectives both on what's driving the design and on the ability to examine them in depth and in context.

Figure 4.1 Example constraint plotter visualization. As the user hovers the curser over a box, the interactive layer brings up information about the constraint and its activation status. The box is red since the constraint is "Active".

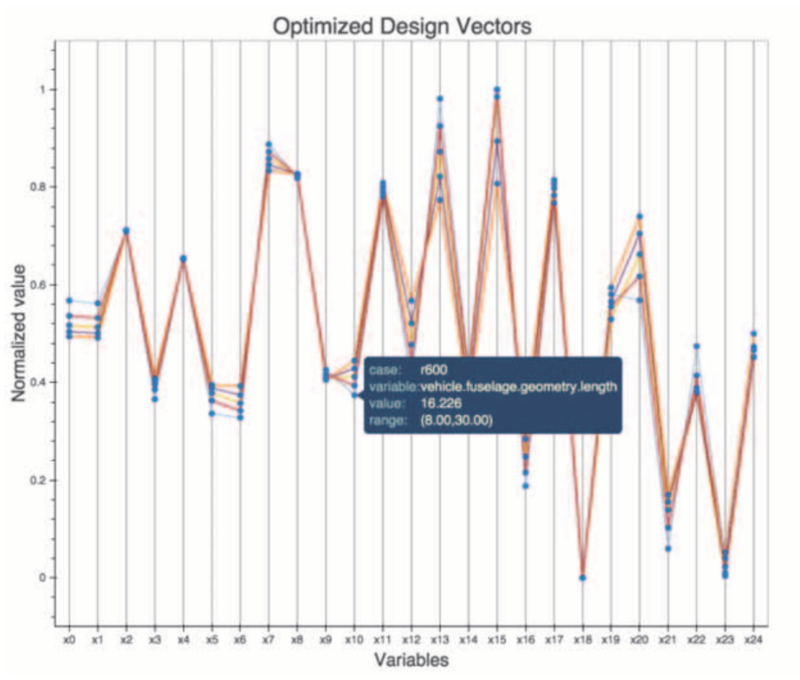


constraint: clMarginPoint5
element: 24
status: Active
margin: -1.424e-11

Design Vector Plotter

In posing the design problem as an optimization problem, we represent the air vehicle as a vector in a high-dimensional design space. One way to visualize the design is to plot the vector components. This enables very high-level comparisons of different designs. An interactive layer, implemented via the Python-based Bokeh library (Bokeh Development Team, 2014), allows the user to extract more detailed information from the plot. An example Multi-Line Plot is given in Figure 4.2. The figure shows that the user can also hover over individual dimensions of each design vector to access meta-data associated with the variable. The interactivity improves the accessibility of a potentially dense plot.

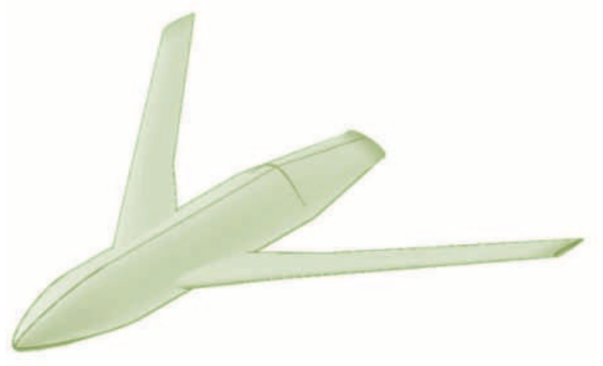
Figure 4.2 Example multi-Line Plot. This plot compares the design vectors of five different optimized vehicles. The cursor is over the fuselage length variable for the "r600" design.



Python Link to NASA's Design Visualization Tool

It is often useful to inspect the design visually. We build a Python-based interface between the ACES output files and the NASA Open Vehicle Sketch Pad (OpenVSP) (Waddington and McDonald, 2015) – an open source tool designed for aerospace vehicle visualization and geometry manipulation. The interface combines aircraft definitions stored in ACES output files with pre-defined OpenVSP templates, which can then be rendered into three-dimensional models like the one shown in Figure 4.3.

Figure 4.3 A tailless aircraft rendered by the OpenVSP design viewer.



ACES outputs gross aircraft parameters—such as fuselage length, wingspan and wing twist—but leaves other design details undefined. Users can fill in the gap in the visualization template using the OpenVSP graphical user interface. It follows that while the visual representations of the vehicles may not be definitive (if the geometry specification is sparse), they do provide a basic idea of what the vehicle would look like.

The combination of the backend ACES design tool (imperative programming) and the frontend OpenVSP interface (decorative programming) represents an efficient division of labor. Leveraging an open-source rendering and visualization environment further improves the quality and maintainability of our tool.

Finally, OpenVSP provides a potential geometric bridge to higher fidelity analyses. OpenVSP can export triangulations and meshes for CFD, panel methods, finite element analysis and, potentially, thermal and electromagnetic analyses. Of course, the usefulness of these tools is a strong function of the quality of the geometry, which can be underspecified in early conceptual design. Using OpenVSP as a link to high-fidelity analysis should be recognized as a longer-term, case-by-case proposition.

5. Verification and Validation

Verification and validation (V&V) is a critical step in the development of real systems and design tools. Ideally, we should validate the design tool against the performance of established designs to ensure that our models, and our integration of those models, are correct.

In the case of highly coupled, multidisciplinary design, we are confronted by the difficulties of isolating cause and effects and of attributing inconsistencies and errors to component models. Furthermore, it can often be difficult to make fair comparisons between optimized designs and real-world systems, whose designs are driven by a multitude of complex objectives, constraints and even non-technical considerations. Design considerations can also be proprietary or poorly documented). Omitted objectives and constraints can lead to design differences.

The coupled nature of the analysis and the complexity of system-level validation motivates a two-pronged approach to verification and validation: (1) Isolate and verify individual model components using unit tests, and (2) apply system-level validations, or vehicle tests, against existing vehicles. The subsequent discussions will cover these two V&V approaches in sequence.

Unit Tests

The first component of the ACES V&V process consists of hundreds of unit test functions that cover the core physics, geometry, and performance models. The tests verify the functionality and correctness of small code units against known true values.

For example, we write unit tests to verify the Weissinger lifting line implementation against known analytic solutions, such as the elliptic wing, and against known numerical solutions (Bertin and Smith, 1998). Likewise, the skin friction and aerodynamic component form factor computations are verified against experimental and analytic solutions at varying Mach and Reynolds numbers. The results of geometric calculations, such as wing and tail exposed areas, and mean aerodynamic chords are also tested against known values of existing designs and wind tunnel models. Basic area and volume calculations of geometry primitives that are used to build up wings and fuselages are likewise tested against known solutions.

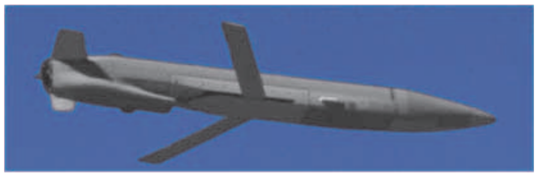
Unit testing pays dividends in several ways. First, while many of the component tests may seem trivial in isolation, their combined effect is to bisect the highly coupled physics, provide assurances of component code quality, and help isolate problems and bugs where and when they occur. Second, consistent with contemporary software development practice, the unit tests are run after every major feature addition and code change to quickly detect code regressions. This continuous testing approach helps to control the growth and propagation of errors in the code base and helps to ensure that a working codebase always exists for production use. This has the effect of accelerating both code development and design production. Finally, successful application of unit testing requires the program to be structured such that atomistic code functionalities must be testable in isolation. This testability requirement provides an important additional imperative to modular design, with its attendant positive effects on software maintainability and extensibility.

Vehicle Tests

After affirming that individual code modules are working in isolation, we perform vehicle-level validations to ensure that, when given analogous design requirements and constraints based on known designs, ACES can produce matching designs. We performed preliminary validation against three vehicles: the RQ-4 Global Hawk UAV, the Joint Air-to-Surface Standoff Missile (JASSM) and the ADM-160B Miniature Air-Launched Decoy (MALD). In all cases, the key weight and configuration results agreed well with the existing designs – in many cases to within 10-15%. Of course, any discussion of the accuracy of the preliminary validation is subject to the aforementioned uncertainties in the reported data and design intent of the existing designs. It should also be mentioned that the vehicle tests are not complete, additional validations using different configurations, including tactical aircraft and transports are needed to more fully validate the tool.

In this section, we discuss the details of the ADM-160B MALD design study as an example of the validation process. The MALD is a low-cost air-launched decoy. The turbojet-powered vehicle, which is shown in Figure 5.1, has folding wings and resembles a cruise missile in configuration. The ADM-160B MALD is in service with the U.S. Air Force. It is developed from the earlier, smaller Northrop ADM-160A MALD-A.

Figure 5.1 Picture of ADM-160B MALD in flight



Source: Raytheon

The MALD is a modular design and can host different mission payloads. The baseline decoy can mimic the flight profile and sensor signature of larger tactical aircraft to help improve tactical aircraft survivability and defeat integrated air defense systems (IADS).

To help validate ACES, we design a MALD-like air vehicle to the same published performance characteristics and requirements of the real MALD vehicle. The validation vehicle is optimized for minimum launch weight with respect to a range of configuration variables (outlined below). We then compare the optimized design parameters—such as the gross vehicle weight, wing area and engine thrust—to the published values for the MALD.

We chose the MALD as a validation case because (1) it is a relatively modern design, and (2) there is a wealth of published data about it, including the vehicle configuration and gross dimensional data. Cruise missiles are also simpler than inhabited tactical aircraft: with fewer components and, therefore, fewer sources of potential system and weight uncertainty.

The authors are cognizant that minimizing the launch weight is, in general, not a sufficiently descriptive design goal. The MALD is likely designed to a multitude of mission-effectiveness considerations as well as acquisition and life-cycle-cost goals. High on our list of priorities is the incorporation of more sophisticated cost and effectiveness models in ACES. That said, we are helped in our simplification of the objective function by two factors: (1) System costs (our principal concern) tend to scale with weight, and (2) the weight of air-launched vehicles like the MALD has an obvious, negative impact on host-platform (e.g., a fighter) performance. The carriage requirement adds to the impetus to minimize the missile weight. This renders the minimum weight solution more representative of the true solution.

Design Requirements

The top-level mission requirement in the validation calls for a Mach 0.75 cruise mission at a constant altitude of 30,000 feet out to a range of 500 nautical miles. The range requirement is consistent with published MALD performance quotes (Raytheon Missile Systems, 2010), while the cruise Mach numbers and altitude are conjectural. They are meant to approximate a representative MALD mission. In practice, we would expect the vehicle to be designed against a multitude of different missions.

Additionally, the vehicle is designed for the dash, maneuver and ceiling states listed in Table 5.1 Each flight state is defined by a speed, altitude and load factor. In this analysis, we assume full fuel weight for each condition. At each flight state, the vehicle is sized to satisfy thrust, maximum lift, angle of attack limit and over-wing shock intensity constraints. The imposition of a range of constraints at each of the corner flight conditions ensures that the aircraft can operate not just at the cruise point but also over potentially more stressful portions of the flight envelope.

Table 5.1 Notional MALD mission and point flight states. Values in bold are known to match published MALD figures.

States	Mach	Altitude (ft)	Load Factor (g)
Mission			
Ingress	0.75	30000	1.0
Other			
Low altitude dash	0.85	3000	1.0
High altitude dash	0.9	30000	1.0
Mid altitude maneuver	0.55	**19000**	**2.0**
High altitude maneuver	0.75	33000	1.3
Operating ceiling	0.65	35000	1.0

The mid-altitude maneuver load factor and altitude, as well as the operating ceiling requirements, are from published MALD literature (Raytheon Missile Systems, 2010). The Mach 0.9 dash speed is based on the quoted performance characteristics of the earlier Northrup ADM-160A (Vick et al., 2002). While the ADM-160A and ADM-160B constitute different designs, they are conceptually similar and designed to mimic the same types of tactical aircraft. The design requirements are assumed to be comparable.

The known MALD performance parameters are highlighted in bold in Table 2. The other parameters for the different mission and instantaneous states are conjectural. For example, the high altitude maneuver is derived from commercial airliner requirements and captures a constraint against high-speed wing buffet. The low-altitude dash condition is based on the performance of tactical aircraft in low altitude penetration missions, which we suppose the MALD has been designed to simulate.

We can now summarizes the MALD validation problem formally:

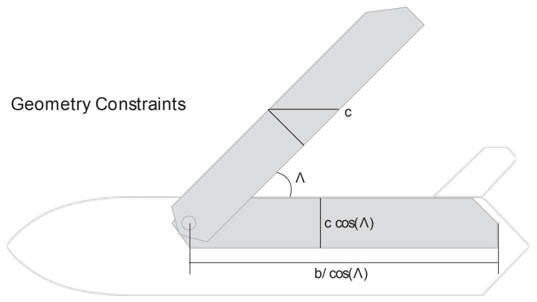
$$
\begin{array}{ll}
\text{minimize} & W_{launch} \\
\\
\text{w.r.t} & S_{ref}, (t/c)_i, \theta_i, AR, \Lambda_{1/4}, \lambda, l_{fuse} \\
 & W_{launch}, W_{MZFW}, W_{f\,j}, T_{sls} \\
\\
\text{s.t.} & R > R_{req} \text{ (all cruise segments)} \\
 & r_{climb} > 500 \text{ (cruise and dash)} \\
 & r_{climb} > 300 \text{ (ceiling)} \\
 & M_{DD}(y) > M_\infty \text{ (all cruise and dash conditions)} \\
 & W_{MZFW} > W_{ZFW} \text{ (weight compatibility)} \\
 & n_z C_l(y) < C_{l\,max}(y) - C_{l\,margin} \text{ (all flight conditions)} \\
 & Re(y) > 1000000 \text{ (all flight conditions)} \\
 & \alpha_{min} < \alpha < \alpha_{max} \text{ (angle of attack limiter)} \\
 & b/cos(\Lambda) < \kappa l_{fuse} \text{ (wing folding)} \\
 & c_{root} cos(\Lambda) < w_{fuse}/2 \text{ (wing folding)} \\
 & v_{req} < v_{fuse} \text{ (fuselage volume)} \\
 & d_{engine}/w_{fuse} < 0.9 \text{ (engine fit)}
\end{array}
$$

The objective, as previously discussed, is to minimize the launch weight (W_{launch}). The design variables include the wing and fuselage geometry, the sea level thrust (T_{sls}), the fuel ratio at different mission states (W_{fj}) and the empty and launch weights. Note that in this problem, the launch weight is both the optimization objective and a design variable. The zero fuel weight convergence is controlled by the optimization as discussed previously in Section 2.

The performance constraints include flight segment range R, which must be greater than the required segment ranges R_{req}. The constraints also include the climb rates (r_{climb}), drag divergence Mach number (M_{DD}), maximum lift ($C_{l\,max}$), wing Reynolds number (Re) and angle of attack (α) limits at the various flight states. Here, the Reynolds number constraints do not represent any physical design limitations. They are, rather, a way to cope with the inability of the high lift and parasite drag components of the design tool to model flow at low Reynolds numbers.

There are also a number of volume and configuration constraints. The fuselage must have enough volume (v_{fuse}) to accommodate the fuel, the signal emulator payload, the flight systems and the engine (sums to v_{req}). It must also be wide enough to enclose the engine, with some clearance, internally. Figure 5.2 further illustrates two wing-folding constraints that specify the fuselage must be wide and long enough to enclose the folded MALD wing.

Figure 5.2 An illustration of the wing folding constraint as applied to the wing chord and length. They state that the wing must be able to fold cleanly into the fuselage.



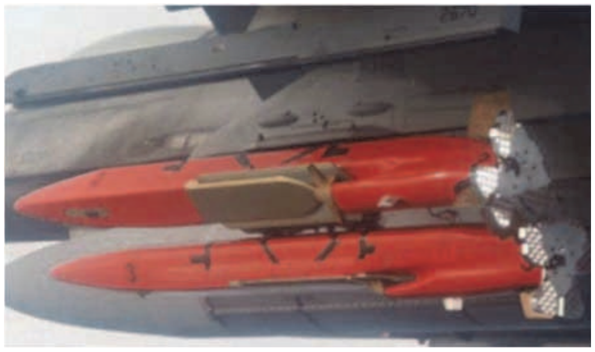
These wing folding constraints link aerodynamic, structure and internal fit considerations. The optimizer can, for example, choose non-optimal fuselage dimensions to accommodate large internal payloads and engines. This illustrates the flexibility of our optimization-driven approach, where we can explicitly and transparently impose constraints to capture complex design tradeoffs.

Design Assumptions

We make a number of important assumptions in the validation process. In many cases the assumptions are meant to fill in gaps in information about the configuration or performance of the MALD vehicle. The outcome of the validation is of course subject to the specific assumptions made. No attempt is made to exercise these assumptions as calibration factors so as to get a better match to the MALD, as that would be contrary to the objective of the validation.

We use the approximate MALD fuselage width and heights to try to capture potential weapon carriage and integration constraints built into the system. This is particularly important since the MALD is designed for rack carriage on a single hard point in the manner shown in Figure 11. The width of the vehicle may be specified by weapon integration considerations. The fuselage length, on the other hand, is a design variable and is optimized with aero-structural considerations and to accommodate the internal components.

Figure 5.3 An F-16 carrying multiple MALD rounds on a single hard point.



Source: Raytheon

The vehicle carries a 40-pound decoy signal emitter. This matches the weight and dimensions of published MALD figures (Raytheon Missile Systems, 2010). We assume an additional 20-pound flight system.

MALD has variable wing sweep. Although the ACES tool can support variable wing sweep, for simplicity we assumed a single, fixed wing sweep for all flight conditions. We do, however, assume a 22% penalty in wing weight for the variable geometry wing. This factor is based on historical trends (Beissner et, al, 1984); it includes both the material weight of the wing hinge about which the variable-geometry wing pivots and the weight of the associated hydraulic actuation systems.

The wing and fuselage structural calculations in ACES assume conventional aluminum construction. The MALD airframe is made of (presumably) lighter composite materials. We apply an empirical 20% weight reduction on the calculated metal fuselage and wing weights to approximate the equivalent composite value.

When computing the internal volumes available to house fuel, payload and flight systems, we assume that 80% of the geometric internal volume of the fuselage is usable. All of the fuel is accommodated within the fuselage.

We apply a typical design weight margin of 10% to the vehicle empty weight to account for growth between conceptual design and production. This accounts for the miscellaneous items not included in our simple weight buildup. We also assume a 10% operational fuel reserve and a 2% unusable fuel weight (both relative to total fuel weight). The fuel reserve accounts for both mission uncertainty and any un-modeled portions of the mission – such as accelerations and maneuvers.

The MALD uses the 150-pound thrust class Pratt & Whitney TJ-150 turbojet, which is developed from the smaller 50-pound thrust TJ-50. Data for the TJ-150 is not available in the open domain, while those of the TJ-50 are (Pratt & Whittney, 2015). We use the TJ-50 engine as the reference engine and scale up the design to the required thrust of the vehicle. The scaled engine maintains the TJ-50's sea level thrust to dry weight ratio of 4.3. We also assume the TJ-150 to have the same sea level static thrust specific fuel consumption (TSFC) at 1.4 pound per hour per pound force. Additionally, we assume a constant 4% TSFC increase and a 6% thrust loss due to engine installation and integration, which are applied at all flight conditions. These losses are representative of engines with an “S-duct” inlet like the MALD.

Finally, typical aircraft skin roughness drag, which accounts for surface imperfections, gaps and fasteners, is 6-9% of the total parasite drag (Kroo, 2001), with the higher number reserved for smaller vehicles, for which the same surface imperfections have a proportionally more prominent effect. We assume the higher value of 9% to reflect the scale of the MALD, its low cost manufacturing process and its exposed wing fold slots.

Results

The optimized MALD vehicle configuration is shown in Figure 5.4 Again, detail geometry such as the fuselage cross-section curves and chine are not defined in the design process and are therefore only notional representation. Note nonetheless the similarity with the MALD photo shown in Figure 9.

Figure 5.4 OpenVSP rendering of the optimized MALD-like vehicle.

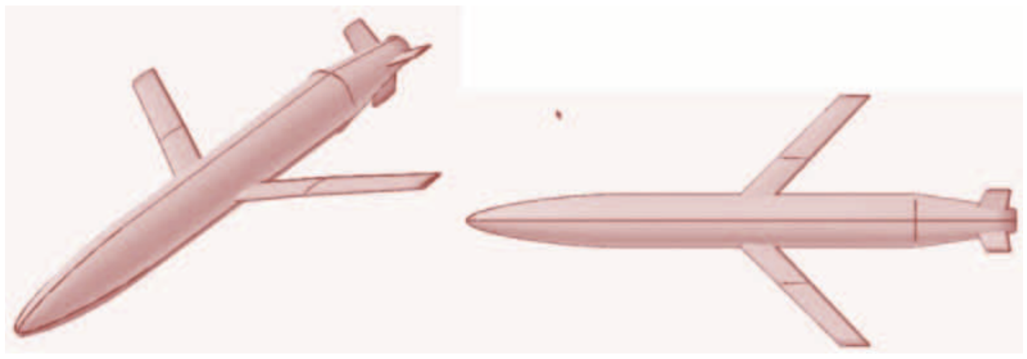


Table 5.2 shows a comparison of the optimized vehicle parameters with the known and estimated values of the MALD. Most values fall within 10% of the published MALD values.

Table 5.2 Comparison of published major MALD design parameters against a minimum-weight, ACES optimized MALD-like vehicle.

	MALD	ACES “MALD”	Difference
S_{ref} (sq. ft)	2.8	2.7	-4%
W_{MTOW} (lb)	280	279	-0.4%
$\Lambda_{1/4}$ (degrees)	Variable	44	N/A
T_{sls} (lb-f)	150	141	-6%
Fuselage Length (ft)	9.5	9.8	3.2%
Wing Length (ft)	5.6	6.3	12.5%

The weight match is better than what one might reasonably expect – given the coarseness of the weight models. More generally, the takeaway here should not be that ACES can somehow predict the weight to within one percent of real vehicles, but rather that the predictions are not dramatically off.

The wing area difference could be caused by subtle differences in designed operating conditions, particularly those associated with high lift requirements such as maneuver and ceiling. The proximity of the solution shows that the inclusion of corner design conditions brings us closer to real-world designs.

The thrust difference is somewhat larger, with ACES under-predicting the required thrust of the vehicle. This could be attributed to differences in the engine lapse rates or errors in drag predictions and thrust/drag bookkeeping. The precise requirements of power intensive flight segments, such as sustained maneuver and dash, could also lead to significant differences in thrust predictions. Finally, it should be said that the precise thrust of the MALD engine is not published, only that it is a 150-pound class engine.

Since we do not simulate the variable sweep, the single-position wing design will be biased towards higher sweep to cope with high-speed dash conditions. To reach the same cruise wing span (for induced drag reasons) at higher sweep would require a longer wing. This is consistent with the optimization results.

In addition to comparing the design outcomes against the MALD, it is also instructive to examine the active constraints to understand what is ultimately driving the design.

Not surprisingly, all segment range constraints are active. This simply means that the vehicle is sized with just enough fuel to perform its mission.

The climb rate requirement at the low altitude Mach 0.85 dash is active. This means that the vehicle is thrust-critical at the low altitude, high-speed condition. In other words, the dash corner flight state is sizing the engine. This is consistent with our expectations because (1) the high-speed dash is undoubtedly a demanding flight state, and (2) the air-launched vehicle does not have a higher drag take off and climb stage, which may otherwise be sizing the engine.

Maximum lift constraints in the wing's mid and outer spans are active in the intermediate altitude (2-g at 19000 feet) maneuver. This means that the maneuver is essentially sizing the wing. Indeed, not including a maneuver condition would lead to a much smaller wing, sized only for the faster cruise state, and an unrealistic vehicle with no operating flexibility.

Finally, the fuselage volume constraint is active, thereby sizing the fuselage length. This is also affirmed by the fact that the wing folding constraints are not active – that is, the fuselage is more than long and wide enough to accommodate the wing.

6. Blended Wing Body UAV Design Study

To better illustrate the workings of the ACES tool in a less conventional design (e.g. not "tube and wing"), we demonstrate a tailless, blended wing body (BWB) UAV design study. The objective is to design a family of UAVs with cruise ranges varying from 600 to 1,800 nautical miles and payload capacities ranging from 100 to 2,700 pounds. Each payload has a corresponding volume requirement.

The blended wing body configuration represents an evolution of the classical (but still unconventional) flying wing configuration. The flying wing, of which the 1950-vintage Northrop YB-49 pictured in Figure 6.1a is perhaps the most iconic example, has no fuselage or tail surfaces, and carries all of its payloads and systems inside the wing. The design of the flying wing is motivated by the desire to minimize non-lifting wetted area. The consequence of this single objective optimization is that flying wings, while potentially efficient aerodynamically, tend to be volumetrically inefficient at all but the very largest of scales, and often suffer from controllability and trim issues. The flying wing configuration has consequently found few applications in production aircraft.

Figures 6.1a and 6.1b A visual comparison of the flying wing (YB-49) on the left and blended wing body (Boeing N+3) design on the right.

Source: NASA

The modern blended wing body, as exemplified by the NASA-Boeing concept in Figure 6.1b, is an attempt to better balance the aerodynamic benefits of all-wing designs with the volumetric efficiency of "tube and wing" configurations. Blended wing bodies add a volumetrically efficient, disk-like center section, which functions as a form of fuselage as well as a trim device (Liebeck, 2004). Admittedly, the distinction between flying wing and blended wing body can seem somewhat artificial to non-experts (and perhaps even experts). The important takeaway is simply that for intermediate-sized, volume-constrained aircraft, the specialization of vehicle components for "lift production" (wing) and "payload capacity" (fuselage) makes a lot of sense.

Historically, a significant challenge in the design of tailless aircraft has been to simultaneously "trim" the vehicle and achieve static stability without significantly compromising aerodynamic efficiency. Trim means that the aircraft is balanced in pitch. Conventional "tube and wing" designs rely on the horizontal tail (or the canard, in some cases) and a long moment arm in the tubular fuselage to provide trim power. In the case of tailless designs, the wing alone has to provide all of the trim moment. This leads to deep coupling between wing aerodynamics and longitudinal trim and stability and complicates the design problem.

Comprehensive analysis of trim is currently not implemented in ACES. This is a significant omission that will be addressed in future development. The omission is partly reflective of the fact that at the conceptual design level, and in the case of conventional aircraft design, the lack of formal trim and stability constraints is unlikely to greatly change the predicted aircraft performance. Well-designed "tube and wing" aircraft have relatively low trim drag – on the order of 1-2% of total drag, which is well within the error bounds of our analysis. The same cannot be said of tailless designs, for which trim is more tightly coupled to the aerodynamic design and can have a profound impact.

Solving the trim problem is predicated on resolving the longitudinal center of gravity (CG). The location of the CG is, in turn, heavily dependent on the details of internal packaging. Although a simple longitudinal CG tracking module is in place in ACES, the internal configuration of tactical aircraft can be highly variable. The designer has many detailed degrees of freedom in internal arrangements, fuel scheduling and active fuel pumping to manage the CG location throughout the mission. In other words, precise CG position is both a complex problem and one with many levers, many of which are not modeled in ACES.

A pragmatic approach, and one that sidesteps the CG problem, is to simply assume a longitudinal stability margin and "position" the CG relative to the wing aerodynamic center. If the vehicle is designed with stability augmentation (typical of tactical aircraft), then it is possible to position the CG close to the aerodynamic center (neutral stability). This allows us to simplify the trim and stability problem to essentially one of achieving a low enough pitch moment coefficient about the aerodynamic center $C_{m_{ac}}$. The Weissinger method can estimate the moment coefficient at all flight conditions, which can then be posed as constraints in the optimization.

Design Requirements

We parameterize the blended fuselage with semi-ellipsoid nose and tail sections and an elliptic cylinder constant section. We remove the overlapping wetted areas between the wing and fuselage at their intersection. The optimizer has the freedom to change the ellipse aspect ratio to create different center sections, or to shrink the fuselage to nothing and recover a flying wing configuration. Finally, to achieve a fuselage-wing "blending," we place constraints on the fuselage to wing height and length ratios.

Table 6.1 details the mission and corner flight state. The UAVs fly similar missions: Take off, ingress to a station at Mach 0.85, loiter for 30 minutes on station, and then return to base. As in the case of the MALD validation in Section 5, we include dash, maneuver, buffet margin and operational ceiling states in the simulation to capture their effect on vehicle sizing. The UAV is designed to operate at higher Mach numbers than the MALD. The Mach 0.95 dashes, for example, are at the edge (or indeed may be beyond) what can be credibly analyzed using the aerodynamic models in ACES. Good results are predicated on (presumed) careful shaping of the nose, fuselage and wing-fuselage intersections to prevent the formation of strong shocks.

Table 6.1 Design flight conditions for the blended wing body UAS

States	Speed (Mach)	Altitude (ft)	Load Factor (g)
Mission			
Ingress	0.85	30000	1.0
Loiter	Optimized	32000	1.0
Egress	0.85	34000	1.0
Other			
Mid altitude dash	0.95	19000	1.0
High altitude dash	0.95	30000	1.0
Mid altitude maneuver	Optimized	19000	2.0
Buffet margin	0.7	36000	1.3
Operating ceiling	Optimized	45000	1.0
Takeoff	Optimized	0	1.0

At each flight state, we impose stall, force balance and over wing shock intensity constraints. We also add trim constraints in the form of a minimum limit on wing planform C_{m_0}. The mission parameters labeled "optimized" in the table are design variables; the loiter, maneuver, ceiling and takeoff Mach numbers are all optimized simultaneously with the aircraft configuration. This formulation gives the optimizer an opportunity to simultaneously optimize the vehicle with its mission. In particular, optimizing the loiter Mach number is important to achieve maximum specific endurance. Optimizing the ceiling and maneuver Mach numbers allows us to trade engine thrust and lift capacity limits at these stressful conditions. This specific optimization problem can be summarized using the follow statement:

minimize W_{MTOW}

w.r.t $S_{ref}, (t/c)_i, \theta_i, AR, \Lambda_{1/4}, \lambda, l_{fuse}$
$w_{fuse}, h_{fuse}, W_{MTOW}, W_{MZFW}, W_{f_j} T_{sls}$
$M_{loiter}, M_{to}, M_{ceiling}, M_{maneuver}$

s.t. $R > R_{req}$ (all cruise segments)
$T > T_{req}$ (all loiter segments)
$r_{climb} > 500$ (cruise, dash and loiter)
$r_{climb} > 300$ (ceiling)
$l_{to} < 8000$ (balanced takeoff length)
$M_{DD}(y) > M_\infty$ (all cruise and dash conditions)
$C_{m\,ac\,planform} > 0.03$ (all flight conditions)
$W_{MZFW} > W_{ZFW}$ (weight compatibility)
$n_z C_l(y) < C_{l\,max}(y) - C_{l\,margin}$ (all flight conditions)
$Re(y) > 1000000$ (all flight conditions)
$\alpha_{min} < \alpha < \alpha_{max}$ (angle of attack limiter)
$v_{req} < v_{fuse}$ (fuselage volume)
$v_{fuel} < \kappa_{wet} v_{wing}$ (wing fuel volume)
$d_{engine}/w_{fuse} < 0.9$ (engine fit)
$d_{engine}/h_{fuse} < 0.9$ (engine fit)
$l_{payload}/l_{fuse} < 1$ (payload fit)
$w_{payload}/w_{fuse} < 1$ (payload fit)
$h_{payload}/h_{fuse} < 1$ (payload fit)
$1 < h_{fuse}/t_{wing_{y=0}} < 2$ (fuselage-wing blending)
$1 < l_{fuse}/c_{wing_{y=0}} < 1.5$ (fuselage-wing blending)

Once again, we optimize the vehicle for minimum weight. The design variables now also include the Mach numbers at loiter, takeoff (M_{to}), ceiling and maneuver.

Many of the performance and component fit design constraints are also similar to those posed in the MALD optimization. We add a positive planform pitch moment constraint at all flight conditions to approximately capture the impact of trim considerations. A small but positive pitch moment budget from the planform can allow for either a more negative airfoil pitch moment (which would improve airfoil performance) or a slightly positive stability margin (which would relax the demands on active stability systems).

Finally, the wing-fuselage blending constraints ensure that the length and height of the wing root are similar to the fuselage values. This couples the dimensions of the wing root and the fuselage, and forces the design to adopt a blended wing body configuration.

Design Assumptions

Many of the design assumptions are analogous to those made for the MALD validation, with minor differences in assumed markup factors like the skin roughness. This section will focus on assumptions that are specific to the UAV study.

First, the UAVs are designed with a radar absorbent material (RAM) weight budget to improve survivability. The area density of the RAM follows Nicolai's simple, public domain RAM weight model (Nicolai and Carichner, 2010). We assume that RAM is applied to 50% of the fuselage and to 20% of the wing and empennage wetted areas. The inclusion of additional area-dependent weight components is expected to further incentivize low wetted area configurations.

Second, owing to the large difference in payload requirements, two different families of engines power the vehicles in the family. The lighter configurations (100- and 300-pound payloads) use scaled versions of the J402-CA-100 turbojet, with a thrust-to-weight ratio of 6.5 and a sea level static thrust-specific fuel consumption of 1.1 lb/(hr-lbf).

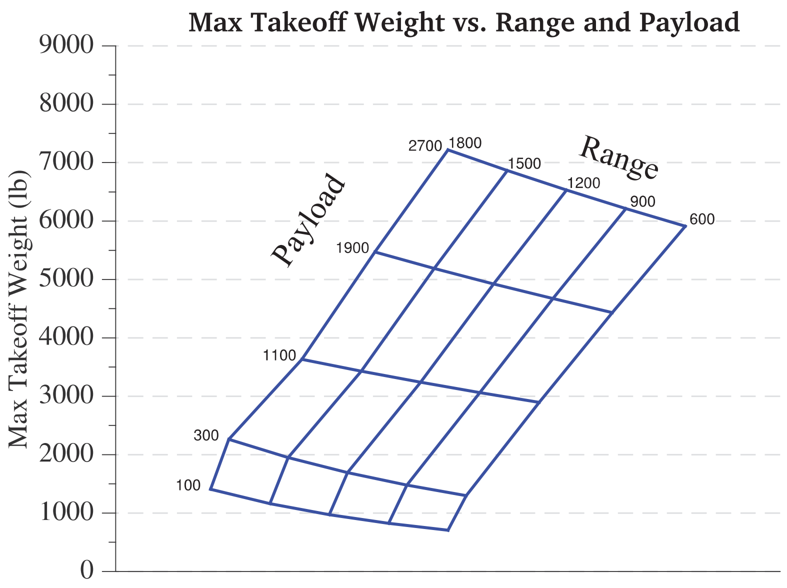
The heavy configurations (1,100-, 1,900- and 2,700-pound payloads) use an engine scaled to the larger, higher bypass ratio Williams FJ44-1AP turbofan. It has a lower thrust-to-weight ratio of 3.9 and a substantially better TSFC of 0.46 lb/(hr-lbf). The differences in the propulsion systems lead to different aircraft weight trends as we change the range requirement.

Results

The goal of the UAV study, representative of the typical ACES use-case at RAND, is to map the variations of vehicle size and configurations as functions of payload and range requirements. The required performance metrics (range, speed and payload) have implications for end-to-end mission effectiveness, while the weight outcomes act as proxies for system costs.

The carpet plot in Figure 6.2 summarizes one the key results of the study: the variation of vehicle weight with range and payload requirements. The graph is therefore based on the outcomes of 25 optimize designs - each intersection point corresponds to an optimized design. Vehicles with the lighter 100- and 300-pound payloads show more pronounced increases in weight with increased range requirements. This is because they are designed with lighter but less efficient turbojet engines.

Figure 6.2 Carpet plot of takeoff weight as a function of range and payload requirements.



We now take a closer look at one point in the carpet plot: the optimized vehicle with a 1,900-pound payload and a 1,800-nautical-mile range. The key design characteristics are summarized in Table 6.2.

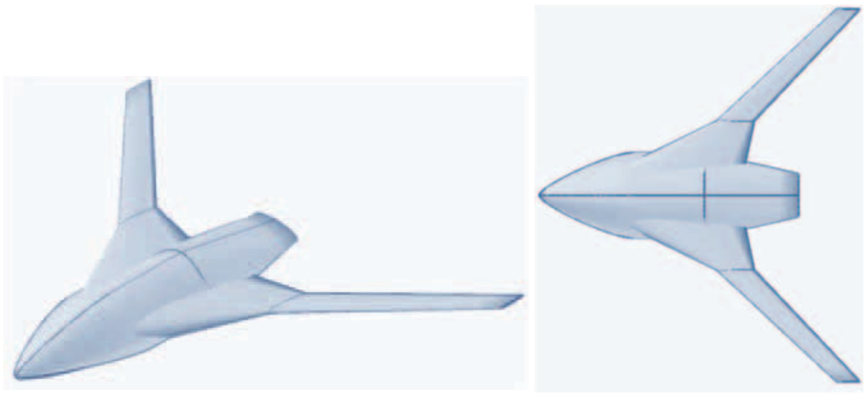
Table 6.2. Optimized characteristics of the 1,900-pound payload and 1,800-nautical-mile range vehicle

Parameter	Value
W_{MTOW} (lb)	5471
W_{ZFW} (lb)	4185
W_{engine} (lb)	689
Wing span (ft)	25.4
S_{ref} (sq. ft)	60.7
$\Lambda_{1/4}$ (degrees)	46.1
Fuselage length (ft)	17.7
T_{sls} (lb-f)	2036
Wing Loading (W/S) (lb/ft sq)	90.1
Thrust to Weight (T/W)	0.37

The relatively high wing loading (90-lb/sq. ft), intermediate thrust-to-weight ratio (0.37) and low fuel fraction of 24% are consistent with requirements for a high transonic cruise vehicle with modest maneuver demands.

The vehicle configuration is shown in figure 15. Since the max wing root thickness is limited by aerodynamic considerations, the forced blending of the wing and the fuselage has produced a vehicle with a short and flat fuselage. The high wing sweep is motivated by both transonic aerodynamics and trim considerations.

Figure 6.3 Optimized BWB configuration with 1,900-pound payload and 1,800-nautical-mile range.



Examinations of the constraint values show that, once again, all segment range and loiter time constraints are active. The fuselage blending constraints are active and are sizing the wing root and fuselage length. The climb constraint at ceiling is critical; the ceiling requirement is sizing the engines.

The spanwise drag divergence constraint is critical in high-altitude dash conditions. Trim requirement is critical in high-lift takeoff conditions. Both conditions are driving the wing sweep.

Maximum lift at takeoff and the takeoff field length are both critical. The wing area is sized by takeoff requirements. This is, in part, because we do not currently model the vortex lift characteristic of highly swept, delta-wing configurations. This omission of the additional vortex lift leads to a conservative, even overly pessimistic, prediction of takeoff and maneuver performance.

7. Conclusion

The ACES conceptual air vehicle design tool is conceived with agility, rationality and tractability in mind, and built on a modern computing platform that leverages open source technologies. The physics models draw heavily from established methods of air vehicle design. We seek to innovate in the software and architectural design dimensions of the problem to contain complexity and facilitate future code extension and reuse. Although portions of ACES remain in development, the tool has already been productively applied to a number of RAND studies. Crucially, we have made efforts toward preliminary, but meaningful, verification and validation. Going forward, we anticipate improvements to the core physics models and mission-state machine, with particular emphasis on structural design and stability and control. Beyond evolutionary improvements, two priority items are (1) to extend the applicability of the tool into the supersonic regime and (2) to generalize the energetics and propulsion modules to include hybrid and electric systems.

References

Antoine, Nicolas E., and Ilan M. Kroo. "Framework for aircraft conceptual design and environmental performance studies." AIAA Journal 43.10 (2005): 2100-2109.

Beissner, F.L., Lovell, W. A., Warner Robins, A., Swanson, E. E., “Application of Near Term Technology to Mach 2.0 Variable-Sweep-Wing Supersonic-Cruise Executive Jet”, NASA Contractor Report 172321, Langley Research Center, Kentron International, Inc., Aerospace Technologies Division, Hampton, VA, 1984.

Bertin, John J., and Michael L. Smith. "Aerodynamics for Engineers, Prentice-Hall." Inc., New Jersey (1998).

Böhnke, Daniel, et al. "Towards a collaborative and integrated set of open tools for aircraft design." 51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition 2013; Grapevine, TX; United States

Bokeh Development Team (2014). Bokeh: Python library for interactive visualization. URL http://bokeh.pydata.org.

Chow, Chuen-Yen, and Arnold M. Kuethe. "Foundations of Aerodynamics: Bases of Aerodynamic Design." New York, Wiley (1976).

Drela, Mark. "XFOIL: An analysis and design system for low Reynolds number airfoils." Low Reynolds number aerodynamics. Springer Berlin Heidelberg, 1989. 1-12.

Drela, Mark. "A User’s Guide to MSES 3.05." Massachusetts Institute of Technology (MIT), Cambridge (2007).

Drela, Mark. "Simultaneous optimization of the airframe, powerplant, and operation of transport aircraft." Hamilton Place, London (2010).

Drela, Mark. “TASOPT 2.00—Transport Aircraft System OPTimization.” MIT, 2010.

Fowler, Martin. Patterns of Enterprise Application Architecture. Addison-Wesley Longman Publishing Co., Inc., 2002.

Freeman, Eric, et al. Head first design patterns. " O'Reilly Media, Inc.", 2004.

Gallman, John W., Stephen C. Smith, and Ilan M. Kroo. "Optimization of joined-wing aircraft." Journal of Aircraft 30.6 (1993): 897-905.

Gray, Justin, Kenneth T. Moore, and Bret A. Naylor. "OpenMDAO: An open source framework for multidisciplinary analysis and optimization." 13th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Fort Worth, TX, AIAA, AIAA-2010-9101. 2010.

Greitzer, E. M., et al. "N+ 3 Aircraft Concept Designs and Trade Studies Final Report." Nasa cr-2010-216794/vol2, NASA Glenn Research Center, Cleveland, Ohio 44135 (2010)

Gur, Ohad, William H. Mason, and Joseph A. Schetz. "Full-configuration drag estimation." Journal of Aircraft 47.4 (2010): 1356-1367.

Hahn, A., “Vehicle Sketch Pad: Parametric Geometry for Conceptual Aircraft Design”, 48th AIAA Aerospace Sciences Meeting, Orlando, FL, Jan 4 - 7 2010, AIAA-2010-657

Hoburg, Warren, and Pieter Abbeel. "Geometric programming for aircraft design optimization." AIAA Journal 52.11 (2014): 2414-2426.

Jason Y. Kao and Hwang, John T and Joaquim R. R. A. Martins and Justin S. Gray and Kenneth T. Moore. “A modular adjoint approach to aircraft mission analysis and optimization”, 56th AIAA SDM Conference, 2015, Kissimmee, FL

Kroo, Ilan, and Richard Shevell. “Aircraft design: Synthesis and analysis.” Desktop Aeronautics Inc., Textbook Version 0.99 (2001).

Liebeck, Robert H. "Design of the blended wing body subsonic transport." Journal of aircraft 41.1 (2004): 10-25.

Lukaczyk, Trent, et al. "SUAVE: An Open-Source Environment for Multi-Fidelity Conceptual Vehicle Design." 16th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference. 2015.

Malone, Brett, and Michael Papay. "ModelCenter: an integration environment for simulation based design." Simulation Interoperability Workshop. 1999.

Mariens, Jan. Wing Shape Multidisciplinary Design Optimization. Diss. TU Delft, Delft University of Technology, 2012.

Martens, P. "Airplane sizing using implicit mission analysis." 5th Symposium on Multidisciplinary Analysis and Optimization. 1994

Martins, Joaquim RRA, and Andrew B. Lambe. "Multidisciplinary design optimization: a survey of architectures." AIAA journal 51.9 (2013): 2049-2075.

Nicolai, Leland Malcolm, and Grant Carichner. Fundamentals of aircraft and airship design. Vol. 1. Amer Inst of Aeronautics &, 2010.

Pratt & Whittney. “TJ-150 Turbojet Engine”. 2015

Raytheon Missile Systems. “Miniature Air Launched Decoy (MALD) & Future Concepts”. NDIA 48th Annual Targets, UAVs & Range Operations Symposium & Exhibition. 10/20/2010

Roux, Elodie. Turbofan and turbojet engines: database handbook. Elodie Roux, 2007.

Sequeira, Christopher J., David J. Willis, and Jaime Peraire. "Comparing aerodynamic models for numerical simulation of dynamics and control of aircraft." 44th AIAA Aerospace Sciences Meeting and Exhibit, AIAA. Vol. 1254. 2006.

Shevell, Richard Shepherd. Fundamentals of Flight. Vol. 2. Englewood Cliffs, New Jersey: Prentice Hall, 1989.APA

Smith, A. M. "High-lift aerodynamics." Journal of Aircraft 12.6 (1975): 501-530.

Torenbeek, Egbert. Advanced aircraft design: conceptual design, technology and optimization of subsonic civil airplanes. John Wiley & Sons, 2013.

Waddington, Michael J., and Robert A. McDonald. "Development of an Interactive Wave Drag Capability for the OpenVSP Parametric Geometry Tool." 15th AIAA Aviation Technology, Integration, and Operations Conference. 2015.

Wakayama, Sean Rikio. "Lifting surface design using multidisciplinary optimization." (1995).

Wolpert, David H., and William G. Macready. "No free lunch theorems for optimization." Evolutionary Computation, IEEE Transactions on 1.1 (1997): 67-82.