Automating Aerospace Synthesis Code Generation

A Tool for Generic Vehicle Design and Technology Forecasting

Thomas Peter Dominic McCall

December 2020

This dissertation is submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
to the
Faculty of the Graduate School of

THE UNIVERSITY OF TEXAS at ARLINGTON

Automating Aerospace Synthesis Code Generation

A Tool for Generic Vehicle Design and Technology Forecasting

Thomas Peter Dominic McCall

The following members of the Committee approve this doctoral dissertation of Thomas Peter Dominic McCall.

Chair

Bernd Chudoba, PhD
Research Advisor
Department of Mechanical and Aerospace Engineering
University of Texas at Arlington

Dudley Smith, PhD
Department of Mechanical and Aerospace Engineering
University of Texas at Arlington

Robert Taylor, PhD
Department of Mechanical and Aerospace Engineering
University of Texas at Arlington

Donald Wilson, PhD
Department of Mechanical and Aerospace Engineering
University of Texas at Arlington

Paul Componation, PhD
Department of Industrial Engineering
University of Texas at Arlington

ACKNOWLEDGEMENTS

Before we begin, I would like to take a moment to acknowledge and thank those who have been directly or indirectly influential and assistive in my journey to complete this dissertation. I, like all, do not walk the path of life alone. Firstly, I would like to acknowledge my supervising professor, Dr. Bernd Chudoba. Without his assistance and direction, I would not be where I am today. Secondly, I would like to acknowledge my many fellow researchers at the Aerospace Vehicle Design Lab. So many of you have been there through it all and in particular I would like to express my gratitude to Doug Coley, Vincent Ricketts, Loveneesh Rana, Ian Maynard, Harin Patel, and David Woodward. Additionally, it goes without saying, but thank you to my family and parents for all your continuous support. Lastly, I would explicitly like to acknowledge and express my sincere gratitude and appreciation to James Haley and Kiarash Seyed Alavi. These last few years have been an adventure and I am glad to have been able to stand by your side and have your company through it all. Through the good, the bad, and the ugly, we walked it together and we will always have that. To my brothers, I thank you.

Thomas McCall
2020

ABSTRACT

The principal development and deliverable of the research presented herein is a generic synthesis assembling decision support environment called *AIDRA-DSS*. The purpose of the system is to develop further expertise and a baseline environment to test complex vehicle automated synthesis architecture synthetization, which would be easily adaptable into a greater cognitive system. This tool has been developed as a precursory and developmental task towards an ultimate objective of an artificial intelligence design and research assistant peer.

The motivation for this research has been to explore the advancement of toolsets for the decision maker and designer operating at the earliest planning and design phase of an aerospace vehicle or program. In particular, the driving motivation of this research is a vision of a future where in the designer is assisted by an artificial intelligence design peer. A vision of the future is one where an artificial intelligent design peer assists the designer in tedious repetitive tasks, design automation, knowledge retention, and more. The goal being reduction in tedious tasks such as data handling, method handling/integration and improvement in time to solution, ease of non-traditional concept consideration, tool reuse/integration, and improvement in design choice and design knowledge extraction and continuation. Such an environment would be advantageous as the early design phase—the conceptual design phase—is ultimately the most significant in determining the success of a program but yet is the shortest in time and sees the least in allocated labor. However, as the development of a true artificial design peer is beyond the scope of a single dissertation, it is identified that a necessary component would include synthesis automation, and hence the principal deliverable of this research.

To address immediate applicability, the system developed is an engineering environment that arrives the user at an applicable synthesis solution toolset to solve a given problem through the provision of standard feedback and decision aiding platforms. That is, it is a framework for automated composable architecture formation that provides a concept, process and method fidelity independent toolset for problem solving. It is a framework that allows engineers to analyze or size any vehicle through a generic synthesis assembly approach. Giving the user the ability to compose a vehicle from different elements, *AIDRA-DSS* creates a tailored sizing code based on the user-designated requirements, removing the tedious task of synthesis architecture assembly from the requirements of the user. The user only need specify what to analyze and the constructs of how to accomplish the analysis.

The solution concept is founded on a decomposition-composition approach. It is a code assembly concept utilizing a warehousing approach. Fundamentally, the user provides a set of inputs specifying the vehicle to be considered, the process of analysis, the methods to use, and the output presentation desired. From these instructions, a synthesizer routine gathers the necessary code elements, both engineering methods and code processing (data handling, method handling, etc.), and assembles the components into a functional synthesis architecture. The synthesis is executed as prescribed by the user and the results are processed and returned to the user.

System functionality and applicability were demonstrated through the execution of a verification case and an exploratory trade study case. The verification case utilized the GHV and the X-51A. In comparison to known design parameter values, analysis results were resolved to less than 5% error, with most error being less than 1%. Successful execution demonstrated proper automated system assembly and method correctness.

The trade study case evaluated air launched airbreathing and non-airbreathing concepts for consideration as reusable hypersonic vehicle research and development platforms. The GHV and X-51A, in addition to the FDL-7/Model-176, served as baseline concepts and configurations for the trade vehicles. In so doing, the blended-body and all-body were represented. Trade variables include concept, configuration, geometric design parameters, payload, mission scenarios, and fuel types. Through the range of trade conditions, a solution space for hypersonic test vehicles was assembled, visualized, and discussed. The concept solutions were considered in light of carrier vehicle geometric and weight constraints.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

Abbreviations

AB	All Body
AB2DS	All-Body 2D Scramjet
ABRCKT	All-Body Rocket
AFRL	Air Force Research Laboratory
AI	Artificial Intelligence
ARRMD	Affordable Rapid Response Missile Demonstrator
BB	Blended Body
BB3DS	Blended-Body 3D Scramjet
BBRCKT	Blended-Body Rocket
CAE	Computer Aided Engineering
CD	Conceptual Design
CI	Computational Intelligence
DACE	Design and Analysis of Computer Experiments
DARPA	Defense Advanced Research Projects Agency
DBS	Databased Systems
DD	Detailed Design
DSM	Distributed Satellite Missions
ESA	European Space Agency
GHV	Generic Hypersonic Vehicle
IS	Intelligent Systems
KB	Knowledge Base
KBE	Knowledge Based Engineering
KBS	Knowledge-Based Systems
KE	Knowledge Engineering
LB	Lifting Body
MDA	Multidisciplinary Design Analysis
MDAO	Multidisciplinary Design Analysis and Optimization
MDO	Multidisciplinary Design Optimization
NASA	National Aeronautics and Space Administration
PD	Preliminary Design
PIDO	Process Integration and Design Optimization

PLC	Product Life Cycle
TSTO	Two-Stage-To-Orbit
UAS	Unmanned Aerial System
USAF	United States Air Force
VA	Virtual Assistant
WB	Wing Body

Greek Letters

μ_a	OEW Margin
ρ_{ppl}	Propellant Density
ρ_{pay}	Payload Density

Variables

E_{TW}	Engine Thrust to Weight Ratio
f_{sys}	System Weight to OEW
I_{str}	Structural Index
C_{sys}	Fixed Systems Weight
k_{crw}	Crew Volume Coefficient
k_{ve}	Engine Volume Coefficient
k_{vs}	System Volume Coefficient
k_{vv}	Void Volume Coefficient
K_w	Wetted Area vs Planform Area
N_{crw}	Number of Crew
OEW	Operating Empty Weight
OWE	Operating Weight Empty
S_{pln}	Planform Area
TOGW	Takeoff Gross Weight
T/W	Thrust To Weight Ratio
V_{pcrw}	Crew Provision Volume
W_{cprv}	Crew Provisions Weight
WR	Weight Ratio
W/S	Wing Loading
W_{crw}	Crew Weight
W_{str}	Structural Weight
W_{sys}	Systems Weight
W_{pay}	Payload Weight

Chapter 1 INTRODUCTION

THE development and demonstration of a generic synthesis environment for improved decision-making and automated synthesis creation is the principal topic and deliverable of this research. This topic is arrived at and exists within a greater scope—artificial intelligence (AI). This greater scope is the driving motivation for conducting the research presented. As such, this chapter documents the principal motivation, the general background on the subject, and topic refinement towards an original research topic (generic synthesis tool), and finalizes with the specifications of the research objective, deliverables, scope, and document layout.

1.1 Motivation

Fundamentally, the motivation for this research is to explore the advancement of toolsets for the decision maker and designer operating at the earliest planning and design phase of an aerospace vehicle or program. From the early aerospace vehicle product gestation phase onwards, the future projects engineer is challenged to develop a level of assurance when committing resources towards a product aimed at achieving an envisioned impact on the future market years after conception. The success of a product is dependent on the quality of the underlying early forecasts, requirement definitions, technology selections, and initial concept and configuration selections. Consequently, the forecasting team and future projects environment is responsible to identify the available product solution space and risk topographies, resulting in the correct choice of the facilitating technologies, baseline concept, and architecture.

It is well known that the designer is supplemented by his tools available. These include software, experience, and knowledge. All are tools of the designer. However, software can be expensive and cumbersome with inherent limitations and problem focus resulting in rigidity and lack of flexibility in addressing non-standard designs. Knowledge and experience require time and dedication. Furthermore, knowledge and experience can be lost. Education is a fundamental approach to knowledge transfer but is frequently relegated to standardized fundamental knowledge and not the particulars an individual acquires through a lifetime of experiences. So how could the designer be better supplemented? How can knowledge be capture and reused? What are ways to improve the decision maker and designer's situation? Blair, in *Launch Vehicle Design Process:*

Characterization, Technical Integration, and Lessons Learned [1], reflects on the situation addressing areas of improvement:

> *" ... Currently, any design synthesis or design update depends on the designer's ideas and experience base on an ad hoc basis. Possible approaches to technology leaps in this area include idea stimulus approaches; use of artificial intelligence and knowledge-based systems to convert designer's judgments and rules of thumb into algorithms; techniques for visualization of the design space; multidisciplinary optimization; and automated synthesis or inverse engineering."*

Similar sentiments have been communicated by an AIAA technical committee in 1991, as stated in *AIAA Technical Committee on Multidisciplinary Design Optimization (MDO) White Paper on Current State of the Art* [2]*:*

> *" ... Multidisciplinary design optimization of aerospace vehicles cannot take place without substantial contributions from supporting disciplines. The identified supporting disciplines and methodologies are the Human Interface Aspects of Design, Intelligent and Knowledge-based Systems, Computing Aspects of Design and Information Integration and Management."*

These statements simply reflect the fundamental motivation of this research. To originally contribute to supporting the decision maker and designer in the development of the aerospace vehicle at the earliest of the design phase via the development of next generation design tools. More specifically, it is envisioned that the aerospace designer and decision maker be augmented by an artificial intelligent (AI) design and research assistant. The original creation of such a system is the motivation of this research.

1.2 Design Process

1.2.1 Design

Engineering design, as stated by Calkins, *"... may be thought of as the arrangement of elements that make up a machine or other man-made system...*" such that "*... an environment is created in which elemental interactions produce a desired result*." [4] More specifically, aerospace vehicle design is the "*... application of the fundamentals of aerodynamics, structures, power plant, stability and control, based upon certain degree of judgement and experience of the individual designer*." [5] The application of

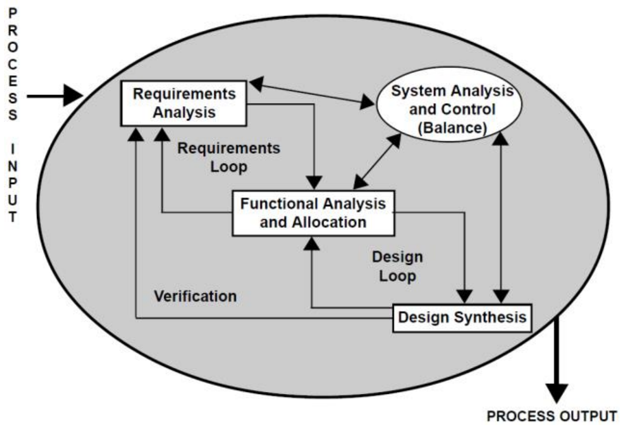


Figure 1-1 System Engineering Process [3]

this knowledge leads to the determination of the design variables that define the vehicle. Calkins [4] further identifies two key elements to the design process, they are synthesis and analysis. These are fundamental components to a design process. They are also a part of the systems engineering processes as defined by Military Standard 499B, which outlines a general problem solving process. [6] In this approach, there are two analysis categories: requirement and function. However, for the discussion here in, it is limited to the simpler construct of analysis in general with the understanding that analysis can be applied to the arena of requirements, function, performance, or any other. For further details on the systems engineering process see refs: [3, 7].

1.2.2 Synthesis

The key element of any aerospace vehicle design methodology is the concept of synthesis. It is the processes to develop concepts and designs where the product (concept/design) is an assembly comprised of core base components; it is a creative activity or phase.[3, 4] It "*... involves the generation of one or more design solutions consistent with the requirements defined during formulation of the design problem and any additional requirements identified during synthesis*."[8] The output of synthesis is a product, a physical architecture.[3] The final product is a solution that has been verified, through analysis, to meet the requirements and performance required. The process, however, can be very time intensive and is *"... one of the important areas to be considered from the standpoint of automation.*"[8]

1.2.3 Analysis

At its core, analysis is the examination of some element. It does not infer or necessitate evaluation. As Calkins defines it, analysis is "*... an examination of a complex system, its elements, and their interactions.*"[4] With the inclusion of evaluation, analysis is the examination of system or element in relation to some given requirement. As such, analysis is "*... any procedure that ascertains whether a given design will meet certain specified objectives*."[9] Within the design process, analysis is a core component of evaluating and verifying that the product of synthesis satisfies the necessary function, performance, and requirements identified and set forth during the design process. The result of analysis is a design update to meet better the function and requirements defined or the verification that the product satisfies the requirements and objectives as laid out.

Analysis includes the classical components of education and is most recognizable by the general community. Different category sets include the classical disciplines: aerodynamics, structural mechanics, propulsion, trajectory, etc. Classical analysis tools fall into the categories of analytical, empirical, and numerical. Commonplace analysis tools/approaches, for example, would include tools such as FEM or CFD.

1.3 Product Life Cycle: Design Phases

An aerospace vehicle is a product of a specific sequence of development, testing, and operation. This sequence of product development and operation is referred to as the product life cycle (PLC). Adapted from Roskam [11], the product (vehicle) life cycle can be divided into four phases: (1) research and development, (2) manufacturing and acquisition, (3) operation and support, and (4) disposal. The research and development phase itself comprises of the classical design phases. There are three. They are conceptual design (CD), preliminary design (PD), and detailed design (DD). The design phases classically occur sequentially.[1] Each phase represents a set of different inputs, tasks, and outputs—completion of which occurs with different toolsets and toolset fidelity. The sequence of events occurring through the CD to DD phases refines the design options into a final design product. This filtration and design convergence is conceptually illustrated in Figure 1-2.

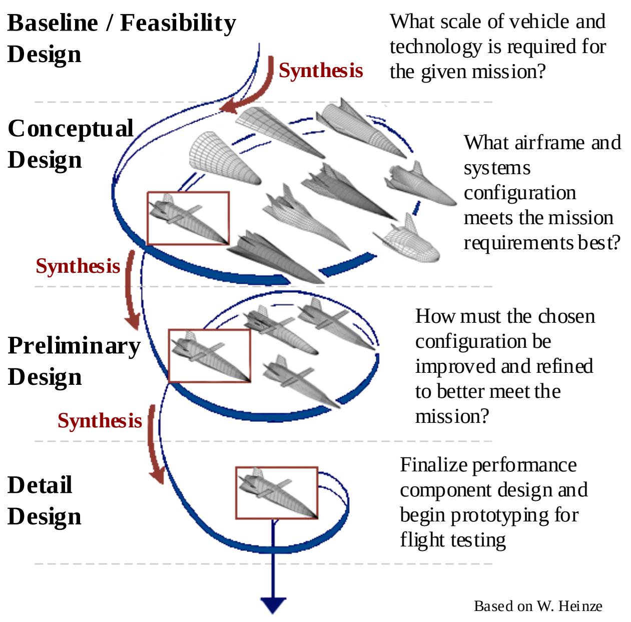

Baseline / Feasibility Design
Synthesis
What scale of vehicle and technology is required for the given mission?
Conceptual Design
What airframe and systems configuration meets the mission requirements best?
Synthesis
Preliminary Design
How must the chosen configuration be improved and refined to better meet the mission?
Synthesis
Detail Design
Finalize performance component design and begin prototyping for flight testing
Based on W. Heinze

Figure 1-2 Design phases and product refinement, modified from [10]

- » Conceptual Design (CD): This design phase involves the tasks of identifying and evaluating sets of plausible concepts2 and configurations3 to satisfy the requirements given and determined during analysis. Outcomes are the identification of a baseline solution concept with principal shape, size, and layout. Lower fidelity toolsets, select small teams, and relatively short turn around characterize this phase.
- » Preliminary Design (PD): Refinement of the design arrived at during the CD phase. Minor modifications to the external design are conducted, as necessary. A larger labor force is introduced along with increased fidelity tools, optimization, and wind tunnel testing.
- » Detailed Design (DD): The decision to manufacture has been made. This phase deals with the generation of detailed part schematics, fabrication, and overall design finalization for

[1] The design cycle generally is considered to execute sequentially and before manufacturing. However, concurrent engineering, which combines/interlays some phases to reduce time and ideally cost, is also popular and has gained traction. Generally, the merged phases are DD and testing/manufacturing though testing and manufacturing can be brought into the earlier phases as well.
[2] A concept is a *"… product or system vision, idea, notion or mental image which maps form to function…"* [12]
[3] Configuration refers to "… *the general layout, the external shape, dimensions and other relevant characteristics*" of the vehicle.[13]

manufacturing of the vehicle. The configuration has been frozen; major design modifications are minimized. This phase has the largest work force and usually the most time allocated.

1.3.1 Product Life Cycle Knowledge versus Design Freedom

The nature of design freedom and cost characterize the design phases. Knowledge and design freedom during the PLC phases are variable. Design knowledge and design freedom are inversely related. As depicted in Figure 1-3, the knowledge available is minimal initially during the conceptual design phase and increases through the PLC phases. The design freedom is exactly the opposite. The maximum design freedom available coincides with the point of minimum knowledge, decreasing rapidly. As such the designer has abundant freedom to consider and evaluate a plethora of design options and combinations, provided times and tools allow. However, generally, neither do, or in the event one does the other does not. Frequently, design exploration is so time costly that the time constraints do not permit it and the advantage of design freedom is significantly reduced or lost. An objective within the community is to attempt to shift the curve imbalance, to bring more design knowledge earlier into the design cycle and extend the design freedom further into the design cycle.[2]

1.3.2 Discipline Integration

Classically the discipline integration across the PLC has also been imbalanced.[2] This is reflected in the Figure 1-3. Classically, certain disciplines have taken precedence during each design phase. Notably stability and control are one of the last to be addressed; aerodynamics and performance are usually favored to the degree that many early designs are driven for maximum performance at the detriment of operational cost and manufacturing. The lack of ability to account for manufacturability, sustainment, and cost earlier in the design process (CD phase) has been an identified issue and is an area for correcting.

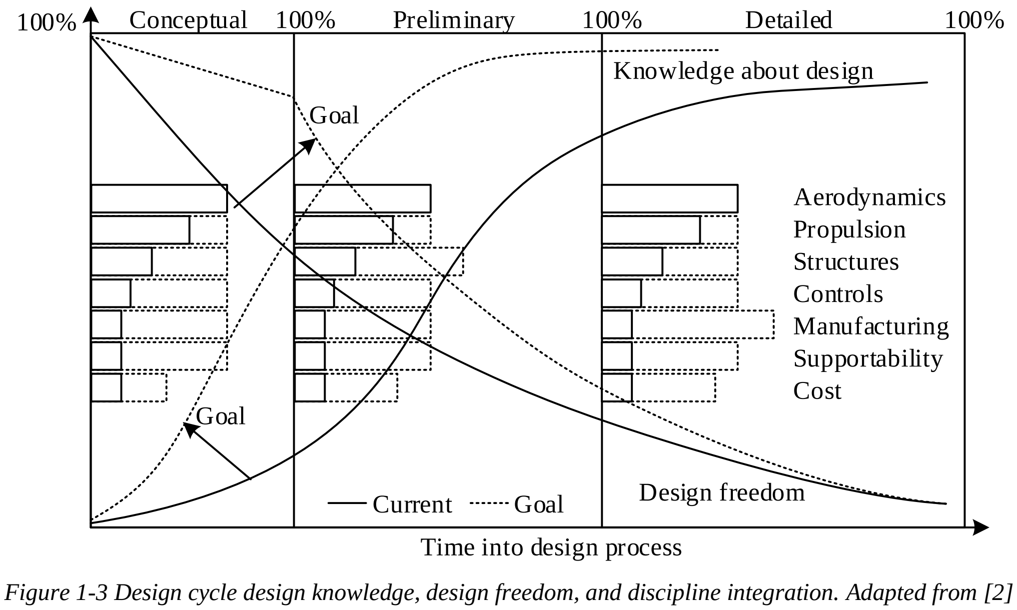


Figure 1-3 Design cycle design knowledge, design freedom, and discipline integration. Adapted from [2]

1.3.3 Cost

The cost for significant design changes increases with PLC phase. Nicolai [14] states, *"... the cost of making a design change is small during conceptual design but is extremely large during detail design.*" This nature is reflected in Figure 1-4. In order to minimize potential cost, it is imperative that the correct design be selected early during the design process, which principally occurs during the CD phase. Approximately 80% of the total configuration is determined during the CD phase.[15-17] These design decisions can account for 70% of the cumulative system cost while only having incurred 1% of the total cost. Small teams, rapid turnaround, and short time allocation characterize the CD phase. The result is that for a highly cost determinant event, comparatively, the cost invested is minimal.

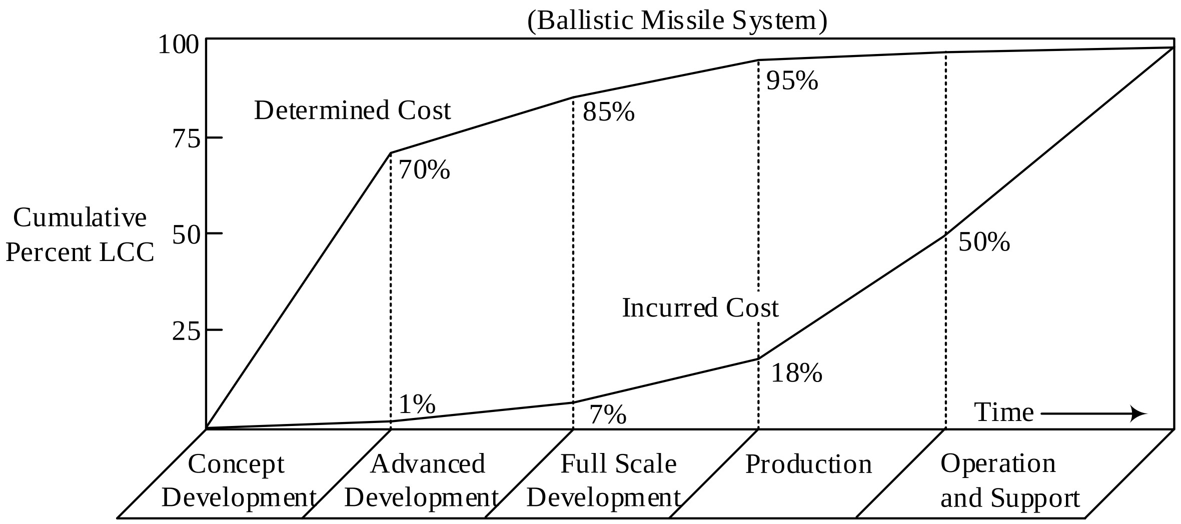


Figure 1-4 Design phases and product refinement in relation to cost. Recreated from [2]

1.3.4 Significance of the Conceptual Design Phase

The CD phase is the phase in which the general design is selected. As Nicolai states, *" ... [t]he fundamental objective of this conceptual design phase is to satisfy the designer and decision maker that the selected concept is worthy of preliminary design continuation.*"[14] Similarly, Torenbeek reflects that " *... [t]he object of this conceptual design phase is to investigate the viability of the project and to obtain a first impression of its most important characteristics.*"[13] The CD phase analysis results in the determination of the primary vehicle concept, configuration, and key design parameters.[18] By the end of the CD phase, approximately 80% of the vehicle configuration is established. As noted previously, approximately 70% of the program cost is established by the decisions made. Given that the actions of the CD phase are so impactful to the total design and the overall cost, the CD phase is critical to a successful program.

1.3.5 Program Exposure and Knowledge

The CD phase requires ideation and therefore creativity and experience. However, a very interesting trend has developed; the project exposure an engineer experiences is decreasing significantly. Half a century ago, an engineer could expect to work on a dozen or more projects. Today, they may be lucky to see the completion of more than one.[19] The result of this phenomenon is the reduction in design experience, knowledge, and exposure. All of which are invaluable to a designer. This illustrates a situation necessitating a system of standardized knowledge retention, transfer, and expression.

1.3.6 Lessons Learned

Due to the increased design freedom, low cost of significant design change, and ability to impact the final product cost, performance, and therefore success, this leads to the conclusion that the CD phase is the most significant and impactful place for the overall product definition and eventual success. Therefore, this research is directly targeted at supporting the aerospace community at this early design phase. The conceptual design phase is characterized by time constraints and low manpower but simultaneously establishes directly or indirectly the probability of success and cost of the program through functional solution identification. This necessitates that the decision maker and designer be best armed during this phase.

1.4 Background and Refining Research Scope

In this chapter section, the research scope and the original contribution objectives are resolved. They are arrived at through a consideration of intelligence; a definition is provided for both human intelligence and artificial intelligence. In addition to the consideration of intelligence, the fields and categories of artificial intelligence are introduced, and finally a general consideration of the application of intelligent systems in aerospace vehicle design is provided. From this, the research direction is identified and selected for this document.

1.4.1 Intelligence

Two questions are addressed. First, what is intelligence? Second, what is artificial intelligence?

1.4.1.1 Human Intelligence

The nature of intelligence is such that it is difficult to define. For millennia, it has been a point of debate. Two definition groups are briefly addressed as they correlate well with the definitions and constructs of artificial intelligence. A common interpretation of intelligence is the notion of multiple intelligences. In the late 1930s, Thurstone [20] correlated intelligence to multiple abilities, identifying nine categories (verbal comprehension, reasoning, perceptual, speed, numerical ability, word fluency, associative memory, spatial visualization). Since Thurstone, Gardner [21] similarly identified intelligence as multiple intelligences working together (visual-spatial, verbal-Linguistic, bodily-kinesthetic, logical-mathematical, interpersonal, musical, intrapersonal, naturalistic) to which emotional intelligence has since been recognized as well.[22, 23] Gardner represents perhaps the most notable author, him identifying intelligence as distinct categories each of which an individual could be weak or strong in.

Intelligence has also been classified as attributes. Sternberg [24] identifies three attributes of intelligence: (1) analytical intelligence, (2) creative intelligence, and (3) practical intelligence. These attributes translate to applicable aspects as problem solving, application of past knowledge to new situations, and adaptability to a new environment respectively. This concept correlates well with that of artificial intelligence (AI).

Krishnakunar summarizes the several ways intelligence has been defined:

» “The ability to learn or understand from experience

» Ability to acquire and retain knowledge

» Mental ability, the ability to respond quickly and successfully to a new situation,

» Use of the faculty of reason in solving problems, directing conduct, etc. effectively”[25]

1.4.1.2 Artificial Intelligence

As with human intelligence, the definition of AI varies depending on the individual asked. Artificial intelligence, in its broadest consideration, is the mimicking of human intelligence by a computational means. The how, what, when, and where distinguish each concept. Harrison [26] notes that, given the variety of AI concepts, the most agreed upon concept “ *... is that AI is the ‘the mimicking, or emulating, of human techniques.’*”. As stated by Munakata, AI is “*... the study of making computers do things that the human needs intelligence to do.*”[27] Frequently, to act intelligent infers some ability to reason. Harrison defines AI “*... as the subfield of computer science that attempts to use computers to emulate the way humans think and reason when solving problems.*” [26] Russel, a notable author in the field, further breaks the definition down based upon thought process, reasoning, and behavior arriving at four distinct definition categories.[28] The four definition categories are (1) systems that think like humans, (2) systems that act like humans, (3) systems that think rationally, and (4) systems that act rationally, where a “… *system is rational if it does the ‘right thing,’ given what it knows*.”[28] Krishnakumar [25] notes that “*... the intelligence of a system is characterized by its flexibility, adaptability, memory, learning, temporal dynamics, reasoning, and the ability to manage uncertain and imprecise information.*” The present section provides general definitions of AI; in the following sections, what AI entails is refined through identification of the fields and categories of AI.

1.4.2 Fields of AI

AI is a very large field. As Russel [28] points out "… *AI systematizes and automates intellectual tasks and is therefore potentially relevant to any sphere of human intellectual activity…it is a truly universal field.*" AI has been categorized into six primary fields: 1) natural language processing, 2) knowledge representation, 3) automated reasoning, 4) machine learning, 5) computer vision, and 6) robotics.[28] Note that the fields generally match or correlate with the attributes and categories of intelligence, as one would expect. A summary of each field is given in Table 1-1.

Table 1-1 Summary of AI fields

AI Field	**Summary**
Natural language processing	Enable effective communication. [28] "Areas such as automatic text generation, text processing, machine translation, speech synthesis and analysis, grammar and style analysis of text etc."[8]
Knowledge representation	Storage of knowledge. [28] "The process of structuring knowledge to be stored in a knowledge-based system."[29]
Automated reasoning	Generate conclusions and answers to a problem from the stored knowledge[28]
Machine learning	Determine new patterns and adapt to changing environment [28] "An adaptive mechanism that enable computers to learn from experience, learn by example, and learn by analogy…[it is] the basis of adaptive systems." [29]
Computer vision	Physical object perception [28] "This topic deals with intelligent visualization, scene analysis, image understanding and processing and motion derivation" [8]
Robotics	Machine mobility and manipulation of objects[28] "This deals with the controlling of robots to manipulate or grasp objects and using information from sensors to guide actions etc." [8]

1.4.3 Tools of AI

The tools developed for the field of AI are many. They can be broken down into three fundamental categories: computational intelligence, knowledge-based systems, and hybrid systems.[30] The different categories are illustrated in Figure 1-5 and are summarized below. Note that in the usage of the term AI in the remaining document, it is considered the inclusive form; that is, on mention of AI, knowledge-based systems, computational intelligence systems, and hybrid systems are all referenced.

1.4.3.1 Computational Intelligence

Computational Intelligence (CI) is distinguishable from knowledge-based AI in that it does not operate on the explicit representation of knowledge; rather it operates on numbers in an intelligent sequence. CI deals only with numerical data, has pattern recognition, does not use knowledge in the same sense AI does, and exhibits adaptively, fault tolerance, and speed and error rates that approaches that of a human.[31] The building blocks of CI are fuzzy logic, neural networks, evolutionary programming, and genetic algorithms.[31]

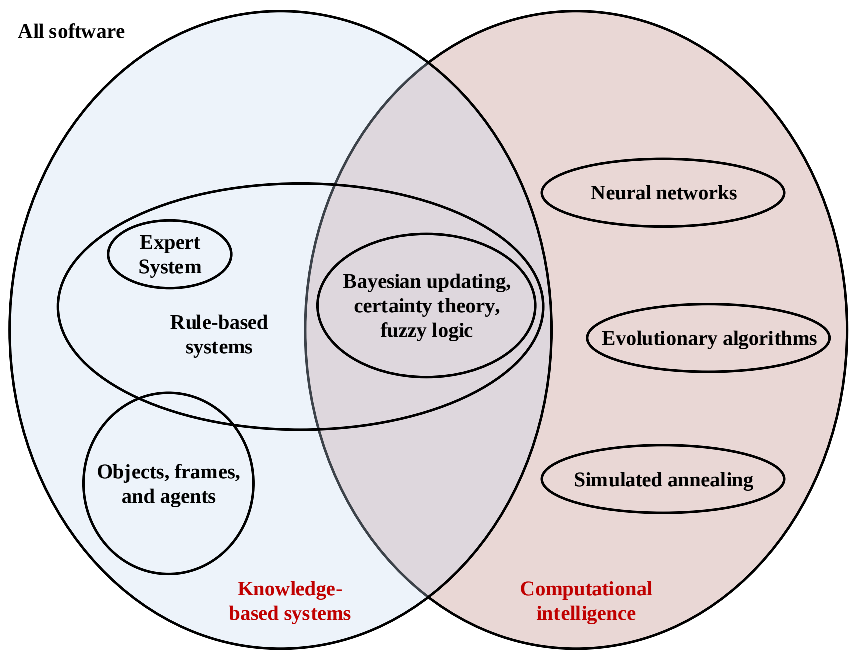


Figure 1-5 Categories of intelligent systems and tools of AI, recreated from [30]

1.4.3.2 Knowledge Based Systems

A Knowledge-based systems (KBS) are computer system that are programmed to store a representation of know-how knowledge about a particular task or field that is used to provide advice, solve problems, and draw inferences.[32] Hopgood [33] notes that a knowledge-based system (KBS) is distinguished from a conventional system (code/software) by its program structure. In a standard software system, the knowledge and system process are intertwined. Whereas in a knowledge-based system, the knowledge element and the control element are separated into two distinctive modules: a knowledge base and an inference engine, respectively. The knowledge base retains the actual knowledge and information in the form of rules, facts, and relationships. The inference engine contains the information on how, when, and what to do with the knowledge stored in the knowledge base.[33] The typical components of a knowledge-based system are illustrated in Figure 1-6. A typical knowledge-based system is the Expert System. An expert system is "*... a type of knowledge-based system designed to embody expertise in a particular specialized domain*".[33] A subset of KBS is Knowledge-Based Engineering (KBE); its focus is on *"...automation of the creation of the CAD geometry, the engineering analysis, and generation of the support information.*"[34]

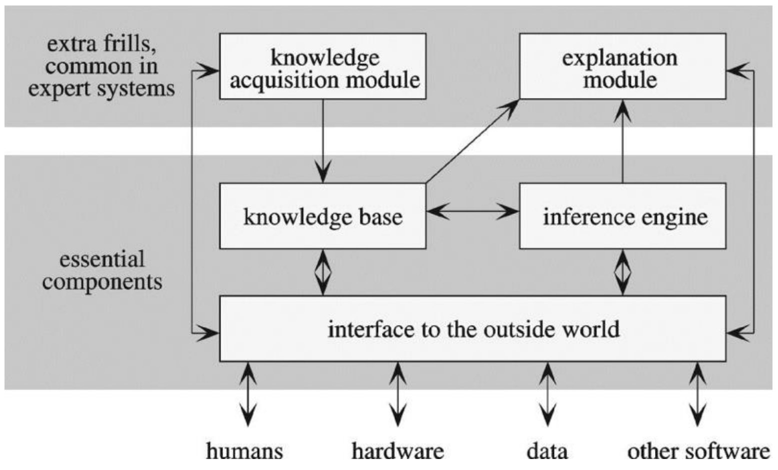


Figure 1-6 Typical components of a knowledge-based system[33]

1.4.3.3 Hybrid Systems

Hybrid systems are those systems that share components or methods of both knowledge-based systems and computational intelligence. KB and CI techniques are not exclusive; they can operate complimentarily within a system in order to address a complex problem with each being applied to its specialized and best suited for the task.[33]

1.4.4 AI in Aerospace

Intelligent systems[4] are common within aerospace as AI and CI techniques are very useful. In this way, intelligent computational applications have included air space management expert systems [35, 36] and naval carrier decision support systems [35-38]. Additionally, methods are applicable to flight performance estimation [39], systems health monitoring [40], and control systems [41] and their design [42]. Intelligent systems have been applied to computer aided design (CAD) and engineering (CAE) [43-46], early initial design generators [16, 47], multidisciplinary design optimization [48-50] and subsystem or disciplinary optimization such as airfoil [51, 52] and trajectory [53, 54]. As engineering is an intellectually arena, it is not surprising that intelligent systems have been found applied across the aerospace industry, including endoatmospheric and exoatmospheric situations.

In this section a brief sampling of intelligent systems in aerospace literature is given. The following is a sampling of the AI and CI applications in aerospace design literature and is not exhaustive as the field is quite large. Almost everything discussed could effectively be described as a hybrid system as design is multidisciplinary and complex. The aerospace problem, both in

[4] Krishnakumar [25] defines an intelligent system as "*... one that emulates some aspects of intelligence exhibited by nature. These include: learning, adaptability, robustness across problem domains, improving efficiency (over time and/or space) information compression (data to knowledge), extrapolation.*"

complexity and data availability, requires multiple tools to be integrated and utilized within the design process. That said, the discussion is by knowledge-based systems (knowledge-based engineering), optimization (computational intelligence), and virtual assistant. Though evident that it is applicable across the domain, the topic is limited to aerospace vehicle design and relevant applications with emphasis on the early CD design phase.

1.4.4.1 Optimization

Perhaps the most affluent region of applied artificial and computational intelligence is in the solving of the optimization problem. Generally, the distinguishing factor is the technique applied whether to the solver itself or the geometric formulation. The techniques have been applied in multidisciplinary design optimization (MDO) of a complex system and to specific component design, such as a structural member or airfoil. The techniques are applicable to the range of aerospace vehicle design including rotary systems, unmanned aerial systems (UAS), high-speed and space systems, in addition to the traditional vehicle configurations. Chae [55] develops and demonstrates—with tip-jet-driven gyrodyne configuration—a conceptual design level fuzzy or soft probabilistic evolutionary algorithm. Optimization application to UAS [56] for configuration independent design space definition for design knowledge identification are demonstrated. UAV wing multi-variable multidisciplinary design optimization with high fidelity CFD and FEA are demonstrated [57] with an evolutionary algorithm. Lee [58] demonstrates the application of an evolutionary algorithm for airfoil section and wing planform design and optimization for aerodynamic performance and radar cross section reduction of combat UAS. Optimization of the airfoil and wing are a significant subject area in literature, examples include [51, 59-62]. In regards to high-speed and space systems, Viviani [63] demonstrates a conceptual design level self-shaping re-entry vehicle configuration approach using genetic algorithm. Bayley [64] demonstrates a study of space access systems including air-launched systems. Mosher [65] gives a tool development for conceptual design of spacecraft that integrates genetic algorithm for design space search. As in the case of the wing and airfoil, trajectory optimization is another area of significant literature. Huang [66] surveys numerical methods including genetic algorithms, swarm, and ant-colony approaches for hypersonic vehicles trajectory optimization. Zotes [54] provides an overview of AI application to aerospace problems such as launch trajectory and interplanetary satellite trajectory optimization.

A common task within multi-objective optimization is data evaluation and knowledge extraction for correct solution identification and general knowledge addition. In the case of multi-objective design, to identify the correct solution from the optimal set requires datamining and specifically to extract design knowledge to determine the best solution. Oyama [67] demonstrates the application of datamining of solar observatory trajectory design solutions found by a multi-objective evolutionary approach. Similarly, knowledge discovery through datamining of optimal solution sets (determined with evolutionary algorithm) is demonstrated for transonic regional-jet wing [68], transonic airfoil design [52], and two-stage-to-orbit (TSTO) fly-back booster wings [69].

As in the evolutionary systems application, neural networks are similarly applied universally within the design arena. They prove advantageous in acting as surrogate models for complex systems reducing the optimization design time or improving rapid technique accuracy. Khurana [61] demonstrates neural networks in conjunction with a swarm optimization approach for airfoil shape optimization. Berke [70] similarly demonstrates the application of neural networks in the approximation of new structural wing design. Khlopkov [71] employees their use in aerodynamic approximation for improved stability and control evaluation in hypersonic vehicle shape based off an optimal solution set.

1.4.4.2 Knowledge Based Systems

Knowledge based systems are generally applied to automate certain tasks within the design process. Given the popularity of MDO over the last few decades and its dependency on a strong geometry generation core, this area has seen significant application. Many modern KBS directly support optimization or analysis. As such many KBS are integral parts of an IS for optimization. These systems would be classifiable as hybrid, however, here the focus is on their KB element consideration. KBS have been applied in the configuration and geometry definition phase of the design process, also referred to as knowledge engineering (KE) or knowledge-based engineering (KBE). Rentema [47] demonstrates a rule based approach for initial concept definition for conventional systems. Similarly, Gong [16] demonstrates a KBE approach to missile design with KE integration as an initial step in baseline configuration definition and evaluation step in an optimization procedure. Similar examples of KBE applied for configuration and geometric model definitions can be found in [44, 72, 73]. Component (part) design is supported by KBE as well [74]. Similarly, as to setup a MDO study requires expertise, so too KBS are applied to support MDO study setup[75]. Additionally, expert systems can be applied to the control and coordination of optimization as demonstrated by Price [76].

1.4.4.3 Advisor (Virtual Assistant)

Up to this point, both optimization and application of knowledge-based systems in engineering (KBE) have been considered. The next intelligent system considered are virtual assistants (VA). The virtual assistant is distinguished from the standard expert advisory systems. A distinguishing ability is natural language processing in particular for easy question-answer interfacing. Well known and commercial virtual assistants include systems such as Amazon’s Alexa, Apple’s Siri, Google Assistant, and Microsoft’s Cortana. A literature review of virtual assistants is given by Martin [77]. He identifies several VAs, however none are within the aerospace vehicle design domain. In the review of design literature, this author has not identified any aerospace design specific VAs either; only three VAs (or VA research programs) related to the aerospace community have been identified. They are NASA’s IBM Watson based systems [78], ESA’s DEA [79], and Daphne [77, 80]. There are advisory system within the expert system domain such as the MDO advisory system [75], however, they fall outside of the range of a modern VA. The vision of a VA would be one that would contain such a system within it in addition to many more system capabilities.

NASA-Watson endeavor is NASA's stepwise research plan towards a more complete VA as is indicated in the technology innovation plan vision, Figure 1-7. NASA has utilized IBM's Watson, a data analytics system. Current work has included the Watson Content Analytics, which identifies trends, connections, and expertise by incorporating and analyzing thousands of documents.[81] In addition, NASA conducted a proof of concept VA with the Watson Pilot and Aerospace Innovation advisor proof of concept that "*... generates leads to hard questions and provide evidence for new paths…"[81]* Both are steps to the greater objective.

ESA's DEA and Daphne target applicability is to the space domain with mission/trajectory planning. DEA is "*…an expert system to support decision-making at the early stages design of spacecraft, a Knowledge Engine for mission design, facilitating Knowledge Management and Reuse.*"[79] It is still in early development. Similar to DEA, Daphne is a virtual assistant to support high level design of distributed satellite missions (DSM).[77] According to Virosi [77], Daphne has a capacity of natural language interfacing for information quarry in addition to trade space exploration tools. It has been tested at NASA. It is specifically for application in satellite mission design.

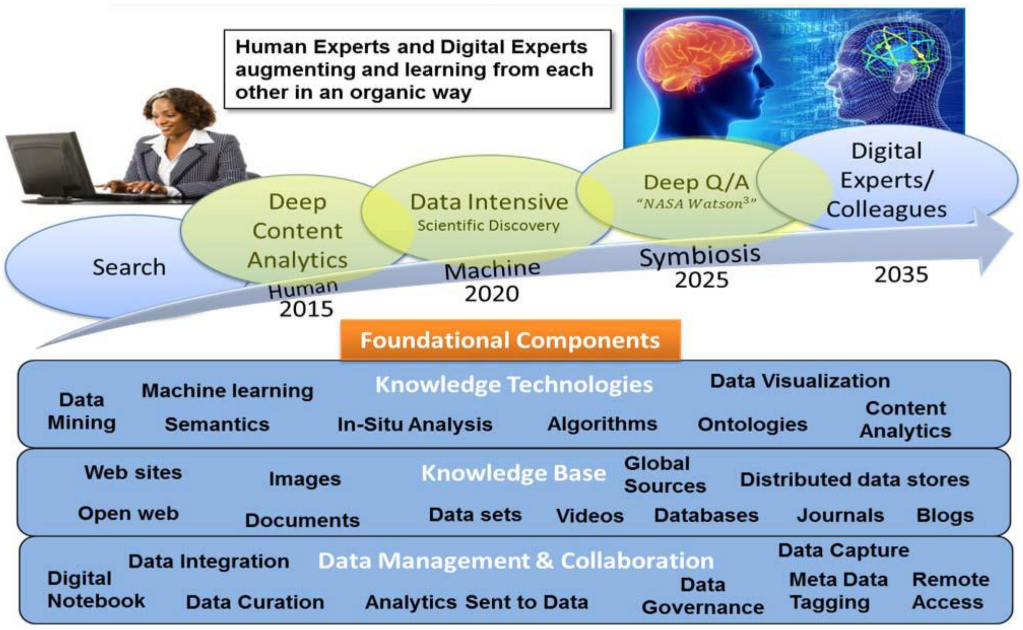


Figure 1-7 NASA's data analytics and machine intelligence capability vision [82]

1.4.5 The Great Problem

In Section 1.3 Product Life Cycle: Design Phases, the PLC was introduced and the CD phase identified as a critical design phase where in design freedom is maximum but yet design knowledge is minimum. An area of research within the community to attempt to alleviate some of the issues within the CD phase is the automation of elements of the design process. Automation of the design process requires computer software. Within the software domain, an applicable tool for automation is AI. A consideration of AI in literature as applied to design and principally the CD phase is given. Recall, that the research motivation has been to contribute to CD tool development through AI and specifically advancement toward an artificial intelligent design and research assistant, which in literature is representable by the virtual assistant. On consideration of the literature, it is evident that research and development heavily focuses on design automation, particularly through optimization systems (a more in-depth account of automation and design tools, in particular multidisciplinary design optimization, is given in the proceeding chapter). On consideration of the availability of systems within the virtual assistant domain (a class beyond the typical hybrid expert system), there are few within the aerospace domain and even fewer (none?) within the conceptual design domain applied to aerospace vehicle design and synthesis. In terms of potential to contribute to research, this is excellent; however, to construct an entire virtual assistant for the CD phase is beyond the scope of a single dissertation. As such, with the target identified and verified as a point of research, it is necessary to identify what this construct could be and what system within that could be addressed to move towards this final objective.

1.4.6 Vision and Research Scope Reduction

There primary research goal is to create a virtual assistant for aerospace vehicle design. The virtual assistant envisioned is an intelligent system that is tasked to show many of the characteristics of a cognitive system. A cognitive system is a system that "… *performs the cognitive work of knowing, understanding, planning, deciding, problem solving, analyzing, synthesizing, assessing, and judging as they are fully integrated with perceiving and acting.*"[83] The system is envisioned to support the decision-making process by providing an intelligent, adaptive, and parametric framework for systems design, strategic planning, and technology forecasting. Emphasis is placed on non-traditional systems, high-speed systems, and space access systems with a focus on the highly abstract CD phase. Envisioned capacity includes simulation/analysis, synthesis, knowledge extraction and reuse, simulated flight-testing, full modularity for ready future modification. Some system specifications include:

» knowledge generation and retention through dynamic knowledge base & data base

» scenario based multidisciplinary design analysis and optimization (MDAO)

» self-composing architecture capability with configuration, hardware, and mission independence

» visualization and interpretation of design space topography

» natural language interfacing

» rational action without human oversight.

On considering the architecture constructs specified, in light of past and current research within the research group that this dissertational research is conducted, included research activities have been synthesis system development [84-87], space program planning [88], technology portfolio forecasting [89] in addition to current research by other research team member applied to vehicle design data aggregation and knowledge extraction. Each topic area would be an element within a virtual assistant. As such, a meaningful and impactful research direction could include continuation of anyone system; for the purposes of this research, the topic area is synthesis system development.

As such, this research is reduced from the domain of AI and VA development specifically, to the development of a synthesis system implementation for future integration into this cognitive system environment. However, until such point that this becomes plausible the system is required to serve as a useful standalone implementation in the synthesis domain of conceptual design. Since the current research endeavors and the necessity for integration within the greater VA system in the near future, the synthesis system shall address and incorporate automation and within concept construction incorporate potential for further automation.

1.5 Research Outlook and Scope

The research outlook includes four topics. They are problem statement, research objective, research deliverables, and research scope. Each is considered.

1.5.1 Problem Statement

The aerospace domain has no virtual assistants for aerospace vehicle design. There should be a virtual assistant for design. There are vast quantities of data and knowledge ready to be employed in addition to analysis methods. Design data is unsurmountable and require computer assistance for evaluation and knowledge extraction. Design cycle time is constrained, design process elements need to be automated as much as feasible. A key element for aerospace vehicle design is synthesis, any virtual assistant for aerospace vehicle design should incorporate automated design synthesis.

1.5.2 Research Objective and Contribution

There are two research objectives aimed at originally contributing to aerospace science. They relate to the principal deliverable and the application of the deliverable to a useful problem.

Principle Objective:

» Development of a complex vehicle conceptual design synthesis tool to assist the decision maker and designer in the analysis and evaluation of design options.

» Develop a synthesis assembly automation framework.

Secondary Objective:

» Demonstrate system operability through a reusable hypersonic test vehicle case study.

The principal objective is formulated with the intended future application of such a system in a larger framework. Such frameworks could include but are not limited to intelligent systems, technology portfolio planning systems, and program or architecture planning systems. Fundamentally, the goal is to advance the decision-making and the design process through the integration of rapid and flexible analysis capability earlier into the decision and design process. The goal is to develop an adaptable synthesis design tool with general applicability and increased transparency.

1.5.3 Research Deliverables

There are three principal deliverables from this research. They fall into the categories of tool specification, tool development, and tool application. They are as follow.

» Specifications for an automated synthesis generation toolset.

» A decision support environment with an integral synthesis assembly tool for tailor made code.

» Solution topographies for air-launched and reusable hypersonic test vehicles.

1.5.4 Research Scope

The research topic itself is vast. The consideration of artificial intelligence, synthesis, vehicle design, and optimization anyone topic can have any number of potential research topics and approaches. This research however is conducted within the scope of addressing one specific area and part that is common to all—synthesis architecting. Within the research environment that this research is conducted, development areas include program planning [88], portfolio planning [89], vehicle synthesis tool development [84-87], knowledge base [90] and database [91]. This research is conducted within the evolution of and lessons learned from said research and specifically in continuation of the work presented by [86, 87].

1.6 Document Outline

To accomplish the objectives towards original contributions to aerospace science as identified, the problem is addressed through a constructive sequence. The sequence follows the tasks of situational illumination, solution identification, solution implementation, system verification, and system application. Logically, this document is organized into this sequence.

» Chapter 1: Introduction – This chapter identifies the motivation and objectives of this research. An introduction to principal topics is given. This includes intelligence, artificial intelligence, and an envisioned AI research and design assistant framework and critical components. From the identification of critical components, the research objectives and deliverables are defined.

» Chapter 2: Literature Review – In this chapter a review of engineering design synthesis tools is given. Emphasis is placed on design automation. Specifications for a generic synthesis system are identified.

» Chapter 3: Solution Concept – A generic synthesis generating solution concept is given. The critical conceptual components are identified and discussed.

» Chapter 4: Concept Implementation – The previous chapter identifies the fundamental solution concept. This chapter documents the manifestation of the system concept into a functional system.

» Chapter 5: Verification and Application – In this chapter the application of the tool is demonstrated. Successful tool development is illustrated through a verification case and a trade study case. The vehicles of consideration are air launched reusable hypersonic cruisers of both airbreathing and non-airbreathing type.

» Chapter 6: Conclusion – Concluding statements are made. The research is summarized; principal deliverables and contributions reiterated. Recommendations for areas for advancement and improvements are given.

» Appendix A: Case Studies Expanded – This section contains expanded description of the baseline vehicles and expanded results of the trade study not included in the main text. This includes more details pertaining to the convergence behavior and selected enlarged solution space and solutions.

Chapter 2 LITERATURE REVIEW

In this chapter, the design tools of the conceptual designer are considered. The objective is two part. First, a consideration of the various tools employed in design. Second, the identification of lessons learned and specifications for a future automated synthesis assembly and decision support environment system.

2.1 Design Classes

The CD phase is characterized by decision-making. Synthesis or design tools are the closest broad categories of toolsets available to the aerospace design engineer that assist the engineer in making decisions. The designer employs the tool to a given problem in order to arrive at a condition such that a designer can make an informed design decision.

Approaches to aerospace vehicle design can be categorized. Chudoba [17, 18] provides a historical review of flight vehicle design synthesis systems and tracks their evolution. After identifying over a hundred system, he identifies five classes of design. Class 1 – 5 correspond to design by experimentation, manual design processes by means of design handbooks and texts, automation through software (discipline specific and local optimization), automation through multidisciplinary design optimization (MDO), and lastly configuration independent design with AI integration (in particular Knowledge-based systems). The evolution of design through these five generations of synthesis systems illustrates a level of increasing proficiency at and automation of systems integration and design exploration.

Fundamentally, the classes identified can be narrowed into two groups. Design with automation and design without automation. In this context, automation generally relates to the act of executing a design process task in an automatic fashion, that is without human immediate direction or minimal involvement. Note that we distinguish automated design and design automation as automation can be applied either to the identification of a design through design variable modulation (MDO) or to the automatic generation of synthesis codes that can themselves include automated design search.

The following sections consider the different design approaches and tools involved. Particular emphasis is placed on design automation and automation of design systems as an AI system of the type prescribed would require automation at every level. Furthermore, automation of product development tasks is key to increased productivity and reduction in time to market and costs.[44] For completeness, the classic non-automated (texts) are considered as well as a point of reference.

2.2 Classical Design: Texts and Programs

Classical synthesis toolsets can be separated into two categories. The categories are text-based (Class 2) and computer-based (Class 3-4). The text-based toolsets are generally either in the form of design handbooks or textbooks. The computer based are software that have an implemented process and analysis routine. The computer-based systems here are distinguished from the more modern systems that are addressed in a later section.

2.2.1 Design Texts

Design texts chronical and attempt to communicate design knowledge and the multidiscipline nature of the aerospace vehicle. The intent of the texts is to educate and communicate knowledge in a transparent fashion. The design texts address the multidisciplinary reality of aerospace vehicle design, generally addressing each principal discipline—aerodynamics, propulsion, stability and control, cost, etc. Classically, they address the design through analytical and empirical methodologies. As physics is a constant, many of the references share similarity in knowledge and even methods directly. However, since many are founded on empirical/statistical methods, they can be non-applicable or useless when addressing non-standard concepts where in the data and experience does not exist.[92] A representative selection of aircraft and launch vehicle design texts are presented in Table 2-1 and Table 2-2, respectively. Many of the texts are well-known aircraft design texts in the community: Torenbeek, Raymer, Roskam, and Nicolai are standard design texts in academic teaching environments.

Table 2-1 Selected aircraft vehicle design texts

Author	**Year**	**Title**	**Reference**
Corning	1960	Supersonic and Subsonic, CTOL and VTOL, Airplane Design	[93]
Wood	1964	Aerospace Vehicle Design Vol. 1, Aircraft Design	[94]
Loftin	1980	Subsonic Aircraft: Evolution and the Matching of Size to Performance	[95]
Torenbeek	1982	Synthesis of Subsonic Airplane Design	[13]
Roskam	1985	Airplane Design	[96]
Raymer	1989	Aircraft Design: A Conceptual Approach	[97]
Stinton	1998	The Anatomy of the Airplane	[98]
Anderson	1999	Aircraft performance and design	[99]
Fielding	1999	Introduction to Aircraft Design	[100]
Jenkinson	1999	Civil jet aircraft design	[101]
Howe	2000	Aircraft Conceptual Design Synthesis	[102]

Author	Year	Title	Reference
Schaufele	2000	The Elements of Aircraft Preliminary Design	[103]
Nicolai	2010	Fundamentals of aircraft and airship design Volume 1, Aircraft design	[14]
Sadraey	2012	Aircraft Design: A Systems Engineering Approach	[104]
Gudmundsson	2013	General Aviation Aircraft Design: Applied Methods and Procedures	[105]
Sforza	2014	Commercial Airplane Design Principles	[106]
Kundu	2019	Conceptual Aircraft Design: An Industrial Approach	[107]

Table 2-2 Selected Space Access Vehicle design texts

Author	Year	Title	Reference
White	1963	Flight Performance Handbook for Powered Flight Operations	[108]
Wood	1963	Aerospace Vehicle Design Vol. 2, Spacecraft Design	[109]
U.S. Air Force	1965	Space Planners Guide	[110]
Humble	1995	Space Propulsion Analysis and Design	[111]
Logdson	1998	Orbital Mechanics: Theory and Applications	[112]
Hammond	2001	Design Methodologies for Space Transportation Systems	[113]
Suresh	2015	Integrated Design for Space Transportation Systems	[114]

The principal concerns with design texts are that they focus on a particular topic and by definition are static. Design texts (as do many software systems) are generally dedicated to a particular concept or configuration and low speed. As a result, many non-traditional and high-speed vehicles are not addressed. The classic textbook, though a significant general aid, serves as an information and knowledge library for fundamental knowledge transfer and application as necessary to low fidelity analysis or early design variable estimations. Although they represent potential method libraries for rapid low-fidelity conceptual design and excellent educational resources for the burgeoning student and engineer, they generally do not represent the state of the art in terms of advanced computational approaches, design automation and optimization, nor address non-traditional concepts. Note that these tools are still highly valuable and applicable in certain situations and even are employed within computational systems as discussed next.

2.2.2 Design Computer Systems

With the advent of computer systems, naturally grew a community of aerospace vehicle design software. Many design tools were built. Like the classical design texts, many early (and even modern systems) tended to be highly focused and would tend towards a monolithic nature, becoming difficult to maintain and modify, especially to address non-traditional concepts. Additionally, many early systems would be distinguishable as being conducting discipline specific or multidisciplinary analysis (MDA) but distinctly not integrating a multidisciplinary design optimization (MDO) framework. Modern frameworks have generally transitioned to a modular approach, allowing for improved system modification, adaptation, and method fidelity variance.

MDO has also become a heavily focused upon element and topic (perhaps to the detriment of development in systems for early conceptual design and program planning where in the optimal design identification is less important than the identification of what concept to even consider to be optimized and for what conditions). Note however, that since these computer systems allow for method fidelity variation, they lend to application beyond the conceptual level and are applied up to a preliminary design synthesis level.[115]

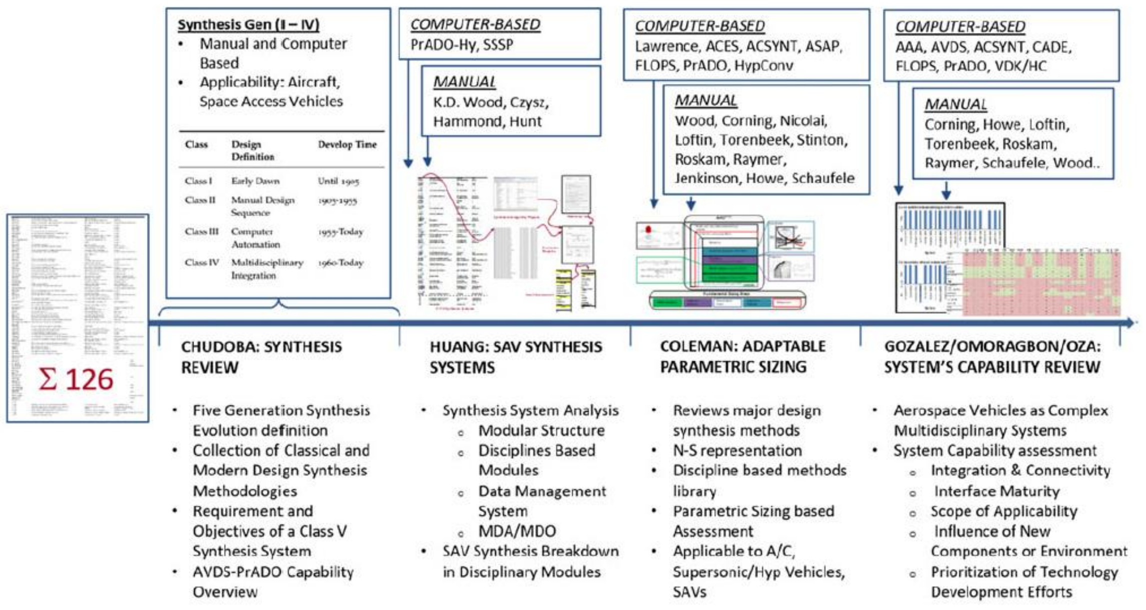
Table 2-3 Selected aircraft vehicle design software tools [86]

Acronym	**Year**	**Full Name**	**Developer**
AAA	1991	Advanced Airplane Analysis	DARcorporation
ACSYNT	1987	AirCraft SYNThesis	NASA
AVDS	2010	Aerospace Vehicle Design System	Aerospace Vehicle Design Laboratory
CADE	1968	Computer Aided Design Evaluation	McDonnell Douglas
FLOPS	1994	FLight OPtimization System	NASA Langley Research Center
Model Center	1995	Model Center Integrate - Explore - Organize	Phoenix Integration Inc
ODIN	1974	Optimal Design Integration system for synthesis of aerospace vehicles	NASA Langley Research Center
PrADO	1986	Preliminary Aircraft Design and Optimization	Technical University Braunschweig
pyOPT	2012	Python-based object-oriented framework for nonlinear constrained optimization	Royal Military College of Canada
VDK/HC	2001	VDK/Hypersonic Convergence	McDonnell Douglas, Hypertec

2.2.3 Synopsis of Systems Reviews

In continuation of Chudoba's review of synthesis approaches, Huang [84], Coleman [85], Gonzalez [86], Omoragbon [87], and Oza [89] have conducted additional surveys of existing aerospace vehicle synthesis tools with a focus on the legacy systems. Figure 2-1 summarizes the sequence of reviews. As mentioned, Chudoba identified many conceptual design systems and postulated a categorical classification scheme. Huang continued the system evaluation with emphasis on considerations for space access vehicles. He surveyed 115 synthesis systems with application to aircraft, helicopter, missile, and launch vehicle design, documenting them based on development history, design logic, module evaluation, and software development, noting both the advantages and disadvantages of each system. Huang noted an absence of system focus on space access vehicles and recommended that future systems address generic design and modular multidisciplinary design capability, multidisciplinary design optimization, data management systems, and dedicated vehicle conceptual design knowledgebase.

Coleman [85] continued systems evaluation. He identified three stages within the conceptual design process—parametric sizing, configuration layout, and configuration evaluation. He evaluated the synthesis systems based on these three sub-phases of the conceptual design phase. The survey forced him to identify the necessity for a readily available process and methods library that would include direction on how and when to implement them. The intent of the libraries being the allowance of necessary design elements (process and methods) for rapid adaptation to solve a new problem. Colman goes on to document and populate a process and method library and implement a parametric sizing tool based on this knowledge.


Synthesis Gen (I – IV)
Manual and Computer Based
Applicability: Aircraft, Space Access Vehicles
COMPUTER-BASED
PrADO-Hy, SSSP
MANUAL
K.D. Wood, Czysz, Hammond, Hunt
COMPUTER-BASED
Lawrence, ACES, ACSYNT, ASAP, FLOPS, PrADO, HypConv
MANUAL
Wood, Corning, Nicolai, Loftin, Torenbeek, Stinton, Roskam, Raymer, Jenkinson, Howe, Schaufele
COMPUTER-BASED
AAA, AVDS, ACSYNT, CADE, FLOPS, PrADO, VDK/HC
MANUAL
Corning, Howe, Loftin, Torenbeek, Roskam, Raymer, Schaufele, Wood..
Σ 126
CHUDOBA: SYNTHESIS REVIEW
Five Generation Synthesis Evolution definition
Collection of Classical and Modern Design Synthesis Methodologies
Requirement and Objectives of a Class V Synthesis System
AVDS-PrADO Capability Overview
HUANG: SAV SYNTHESIS SYSTEMS
Synthesis System Analysis
Modular Structure
Disciplines Based Modules
Data Management System
MDA/MDO
SAV Synthesis Breakdown in Disciplinary Modules
COLEMAN: ADAPTABLE PARAMETRIC SIZING
Reviews major design synthesis methods
N-S representation
Discipline based methods library
Parametric Sizing based Assessment
Applicable to A/C, Supersonic/Hyp Vehicles, SAVs
GOZALEZ/OMORAGBON/OZA: SYSTEM'S CAPABILITY REVIEW
Aerospace Vehicles as Complex Multidisciplinary Systems
System Capability assessment
Integration & Connectivity
Interface Maturity
Scope of Applicability
Influence of New Components or Environment
Prioritization of Technology Development Efforts

Class	Design Definition	Develop Time
Class I	Early Dawn	Until 1905
Class II	Manual Design Sequence	1905-1955
Class III	Computer Automation	1955-Today
Class IV	Multidisciplinary Integration	1960-Today

Figure 2-1 Synthesis systems review summary [116]

Gonzalez [86], Omoragbon [87], and Oza [89] evaluated several synthesis systems based on a broad criteria set for future systems development. Emphasis transitioned in part from a traditional synthesis and sizing system evaluation approach towards technology portfolio planning and forecasting capacity review, or rather a review of systems in light of technology portfolio planning and forecasting. Omoragbon notes, the review is conducted to "*... understand the applicability of existing synthesis systems to the acquisition problem.*"[87] Gonzalez [86] states that the review has centered on assessing aerospace synthesis system's capability "*... to characterize, analyze, and solve classical and new/novel aerospace problems.*" Table 2-4 specifies the capability evaluation criteria. Additionally, evaluated are data handling capacities as outlined in Table 2-5. Both text (by-hand) and computerized systems were considered.

Table 2-4 Synthesis system evaluation criteria [86, 87]

	System Capability
1. Integration & Connectivity	
a	Can assess each hardware technology independently
b	Can assess multiple disciplinary effects for each hardware
2. Interface Maturity	
a	Can combine hardware technologies to form a vehicle
b	Can combine hardware technology disciplinary effects
3. Scope of Applicability	
a	Conceptual design phase applicability
b	Product applicability
4. Influence of New Components or Environment	
a	Modular hardware technologies
b	Modular mission types
c	Modular disciplinary analysis methods
5. Prioritization of Technology Development Efforts	
a	Able to match hardware technology disciplinary models to problem requirements
b	Data management capability
6. Problem Input Characterization	
a	Methodological problem requirements

Table 2-5 Data management system evaluation criteria [86, 87]

	Data Management Criterion
a	Easy to create, change, delete, and view projects and project data.
b	Accommodates all project types and project information
c	Supports entry of annotative comments and appending documents, images, and links for project
d	Accommodates hundreds/thousands of projects
e	Supports data import from your existing systems and databases
f	Supports data export to your existing systems and databases
g	Supports dependency links among projects
h	Provides data cut-and-paste, project cloning, and data roll-over
i	Provides completeness/error checks and data warnings
j	Allows multiple portfolios and portfolio hierarchies (parent-child links)
k	Allows dynamic portfolios (portfolios defined based on latest project data)
l	Provides search, filter, and sort
m	Provides data archiving
n	Provides statistical analysis of historical data (e.g., trend analysis)

From the review, Gonzalez identifies the significance of the open-ended integration platforms (presented in Section 2.3.4 Process Integration and Design Optimization Tools), in particular the modularity, flexibility, and freedom they offer; however, he also identifies that the freedom comes at the loss of a structured tool with method and processes selection and integration, as represented by the classical monolithic systems. Gonzalez identifies a need for a bridge between the two approaches. He describes this bridge environment as "*... an environment with the adaptability of an integration platform, while implementing the knowledge gained from classical conceptual design methodologies to aid the user in the creation of synthesis systems tailor-made to solve given problems*."[86] This leads him to identify the following system specifications:

» "Stores/implements classical design methodologies, both in terms of analytic process and disciplinary methods

» Cross references hardware applicability to stored analytic processes and disciplinary methods

» Allows matching of the analysis framework to problem requirements

» Allows visualization of the ability of the analysis framework to address problem

» Allows comparison of aerospace synthesis systems

» Allows measurement of the multidisciplinary integration level of the analysis framework". [86]

The result of Gonzalez [86], Omoragbon [87], and Oza [89] research was a synthesis toolset for composable systems. An overview is given in Section 2.4.6. Note that this research is a continuation of their work.

2.3 Multidisciplinary Design Optimization

In the following section multidisciplinary design optimization (MDO) is addressed. The topic itself is vast and a full treaty is beyond the scope of this work. For an in-depth review of MDO and its architectures see references: [117-121]. The objective here is not to give a detailed account of MDO and its processes or techniques. Rather, the goal is to obtain a general concept introduction and insight into the world of MDO and identify system attributes and recommendations that would be integral to future design systems. This section address defining MDO, specifying its fundamental process, identifying significant components of MDO, integration tools, and lastly lessons learned and specifications for a MDO environment.

2.3.1 What is MDO?

Sobieszczanski-Sobieski [117] defines MDO as a "methodology for the design of systems in which strong interaction between disciplines motivates designers to simultaneously manipulate variables in several disciplines." Fundamentally, MDO is the application of an optimization routine to a multidisciplinary design analysis (MDA) routine to find the best solution. At its core, multidisciplinary design optimization facilitates the identification of hard to find design solutions by balancing potentially non-intuitive tradeoffs between the subsystems of a complex system.[122] By definition, MDO is distinctly different from a singular optimization approach. That is, optimization applied to a trajectory or structural optimization problem does not infer MDO; MDO requires more than one disciplinary interaction. Additionally, multidisciplinary design optimization of the system does not provide that any one subsystem is optimal. As Rafique notes the "multidisciplinary solution might not be the solution for any one discipline analyzed separate from other disciplines, but is the best solution accounting for interactions."[123]

MDO has become very popular. A reason for the impressive degree of applied MDO is that an aerospace vehicle is a highly complex system with many interlacing disciplines and design variables. It is well known that the aerospace vehicle is a complex system, is multidisciplinary by nature, and as such, for the demanding criteria placed upon the designs, the system's components cannot be designed and developed in isolation.[124] It is necessary to have a design approach that can account for the many interdependencies within the design scope.

2.3.2 Fundamental Process Components of MDO

Vandenbrande [122] describes the design space exploration through implementation of a MDO process as comprising of three fundamental elements: a design explorer, a multidisciplinary design analysis model, and a optimizer. The architecture is illustrated in Figure 2-2. The design explorer is the component that controls the initialization and continuation of the exploration process by generating a design point for the MDA model as denoted by $(x_1, x_2, \dots, x_n)$. The MDA model is an analysis set that can analyze the generated design point for each discipline considered. The output of the analysis is denoted by $(f_1, f_2, \dots f_n)$ and feeds the optimizer. The optimizer is a mathematical optimization scheme to search the design space for the best design solution, given some design criteria and or constraints. The optimizer is closely coupled with the design explorer.

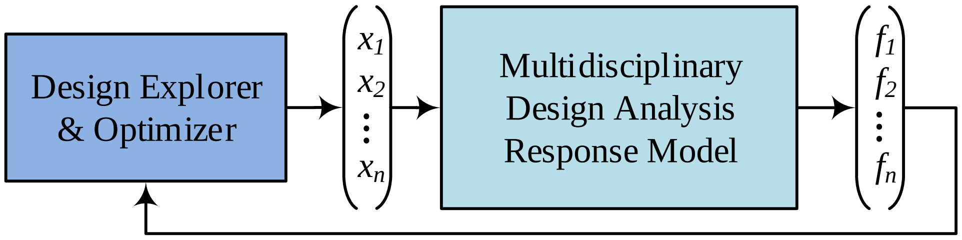

Design Explorer & Optimizer
x_1
x_2
x_n
Multidisciplinary Design Analysis Response Model
f_1
f_2
f_n

Figure 2-2 Generic MDO based design space exploration process [122]

2.3.3 Components of MDO

The above section provided a very top-level understanding of the MDO process and its core components. However, the components of MDO can be further identified. Sobieszczanski-Sobieski [125] initially proposed that MDO is formulated by six components. He identifies them as approximations, system mathematical modeling, design-oriented analysis, decomposition, design space search, human interface, and optimization procedures. The component tree is illustrated in Figure 2-5. Each is summarized below. For further overview on the components of MDO see [117, 125, 126].

» Approximation Concepts: a secondary approximate analysis method callable by a design space search engine to approximate the solution with sufficient accuracy rapidly. Necessitated due to excessive computation cost of the design-oriented analysis component; full analysis is called as required to maintain prescribed error levels. This component correlates to the Multidisciplinary Design Analysis Response Model element identified in the previous section. The approximation approaches include polynomial functions, neural networks, surrogate models, and metamodels. A current area of study for these approximation routines is referred to as Design of Experiments (DOE).

» System Mathematical Modeling: set of engineering methods applied in a modular approach and in an intelligent manner to reduce computational cost. Mathematical code models are modular; the monolithic coding approach is avoided. Current research objectives include the quantification of the non-classical design disciplines and phases such as manufacturing, sustainment, and flight-testing. To reduce data transfer points and computational costs, methods are intelligently reduced or combined, and numerical methods guarantee matching output to input nodal coordinates for synchronization of variable parameterization for reduced workload on data processing and potential analysis grid (mesh) regeneration.

» Design Oriented Analysis: a component consisting of engineering design analysis processes, data management systems (database), and data visualization. Design analysis processes include procedures for analysis execution including repetitive full analysis application to answer the “what if” question, partial analysis execution for low-cost re-evaluations (re-execution of certain modules as necessary dependent on input parameter variation and the reuse of non-affected data), multi-fidelity analysis or fidelity analysis selection, and sensitivity analysis. In regard to data management and storage, data should be stored in a manner for ready retrieval and reuse by the system and designer as well as be communicated effectively visually.

» Decomposition: the act of dividing a complicated optimization problem into less complicated coordinated optimization tasks that can be solved while retaining the multidisciplinary connections. This is illustrated in Figure 2-3. There are three decomposition approach classes—hierarchical, non-hierarchical, and hybrid. In the hierarchical approach, the system is a parent-child pyramidal process where in the data flows between a parent and a children set; data does not directly traverse child-to-child. The non-hierarchical approach does not restrict communication between the children, as such, no parent-children sets are discernable. A hybrid system is one that consists of both hierarchical and non-hierarchical. Decomposition can be a nontrivial task; tools such as genetic algorithms have been applied to the decomposition problem.

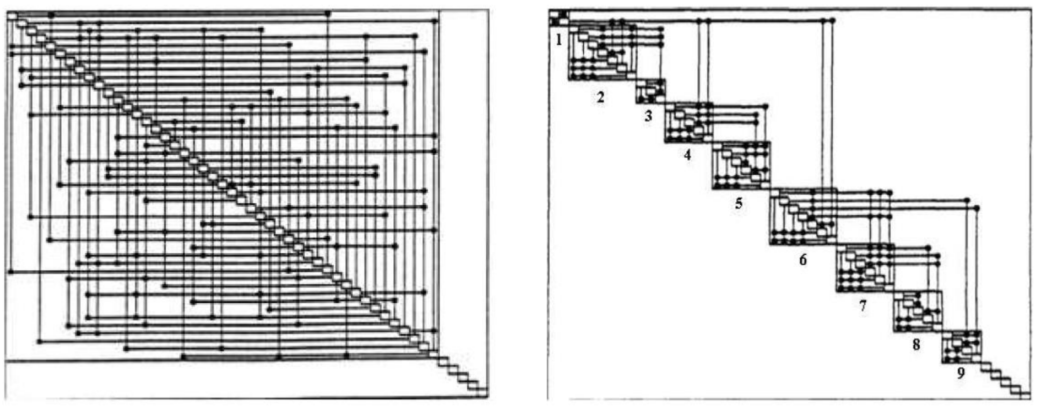


Figure 2-3 Illustration of system’s N^2 organization diagram before and after decomposition and reassembly [127]

» Design Space Search: Evaluation of the design space in search of the “best” or optimal solution given the optimization criteria and design constraints. A mathematical solver seeks the optimum solution. There are many optimizers available for use. The search algorithms include control theory and computational intelligence (AI) based approaches.

» Optimization Procedure: a procedure that organizes the here identified optimization elements into a coherent execution format. There are many approaches and architectures, the selection of which is dependent on the problem and computing resources at hand. An example process is shown in Figure 2-4.

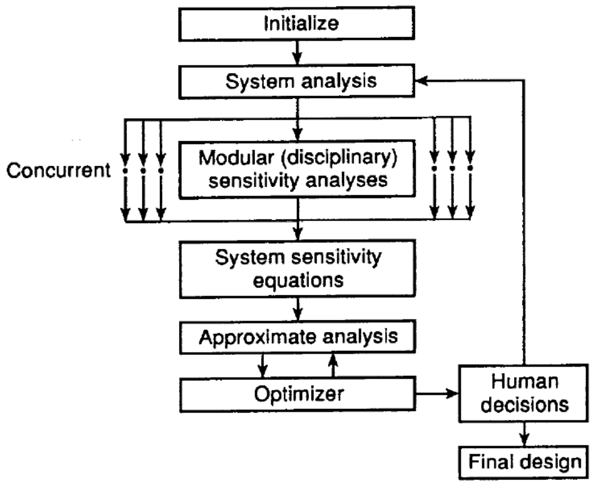


Figure 2-4 Example optimization procedure for a Non-hierarchical system [125]

» Human Interface: a means for the user to interact directly with the optimization process and execution. It provides access to intermediate results for review and judgment as well as intervention in process setup, execution, and termination. Generally, direct human involvement is required in setup and exaction; the setup and execution are not fully automated.

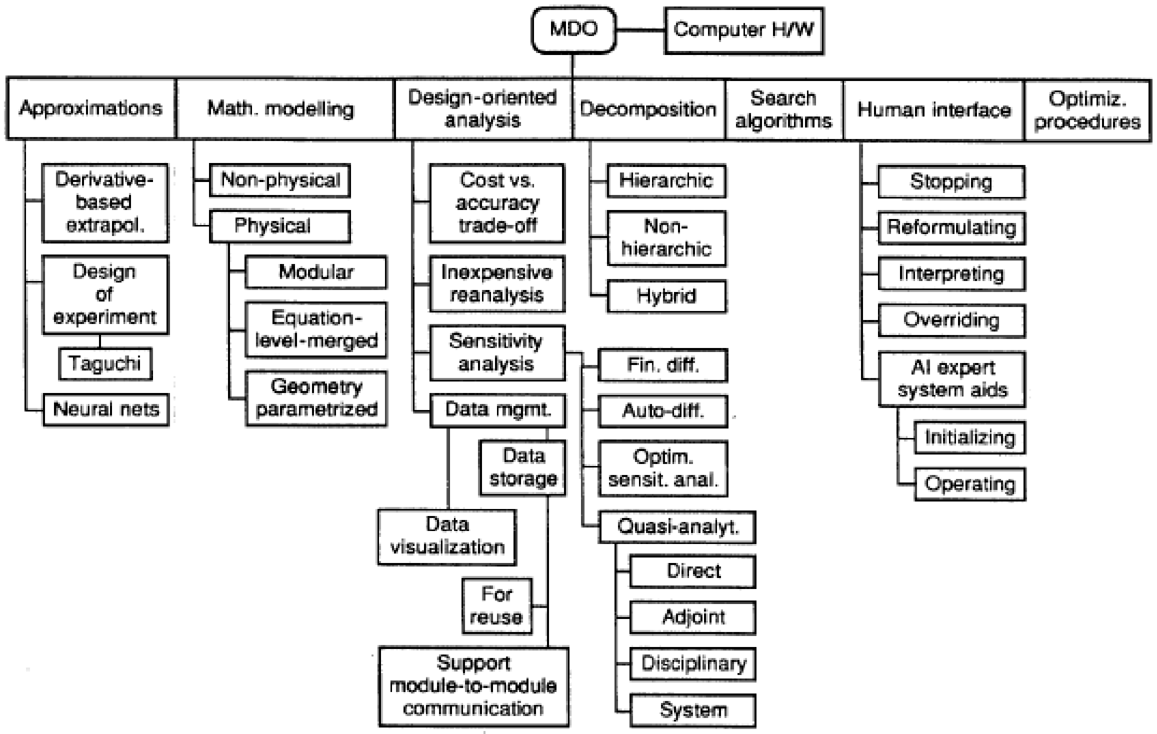


Figure 2-5 Principle components of MDO [125]

2.3.4 Process Integration and Design Optimization Tools

To assist in the optimization process, many commercial and open-source tools have been developed. A common approach to high fidelity MDO design architecture creation is the use of process integration and design optimization (PIDO) execution control software. PIDO system offer a integration capacity of third-party software, optimization, visualization, statistical analysis, and data management.[128] In regards to capability for third party code integration, pre and post processing, and algorithm availability provided, the PIDO software provide similar capability.[120] However, van Gent [129] notes that the PIDO platforms, though developed for similar purposes, each can vary in their approach to user interaction, workflow concept, component integration, distributed execution, derivatives, and convergence.

PIDO systems have been identified and summarized by van Gent [129], Riccardi [127], and Simpson [120]. There are many commercial and open-source systems. Riccardi [127] performed a literature review on the systems identifying and describing over twenty systems. Table 2-6 is a exert from the review of PIDO systems. Simpson [120] likewise identifies and discusses PIDO systems but with emphasis on system capability in both metamodeling and optimization. Riccardi [127] notes that "*…optimization strategies included are mostly the best known algorithms for deterministic and stochastic optimization. Hybrid optimization approaches between the already present strategies are not envisaged.*" Common PIDO systems include CAFFE [130], ModelCenter [131], Dakota [132], OPTIMUS [133], modeFRONTIER [134], and RCE [135].

Table 2-6 Composable system software tools, table excerpt from reference [127]

Software	**Main Features**
AML, AMOpt, Technosoft, 2002	Interfaces with existing tools for structural analysis and post processes analysis. Generative modeling. Integration of third party applications. XML data handling. Process Parallelization. Visualization tools. Multiplatform.
BOSSQuattro, Samtech, 1997	Open design and optimization architecture for parametric analyses, design of experiments, multidisciplinary optimization and sensitivity analysis, statistic analyses and updating. It can make use of internal solvers or integrate external optimization algorithms.
Caffe, Desktop Aeronautics, 2000	Collaborative Optimization framework. Integration of existing code for analysis and optimization. Management of the design process on multiple distributed platforms. GUI. XML data handling.
DAKOTA, Sandia Web, 2009	Flexible and extensible interface between simulation codes and analysis methods. Containing algorithm for deterministic and stochastic optimization, parameter estimation and sensitivity analysis. Multilevel parallel object oriented framework.
iSIGHT, Dassault Syst`emes Simulia, 2007	Capability of include commercial CAD/CAE software and internally developed programs. Interfaces for custom applications and Excel spreadsheets. Design of Experiments, Optimization, and Approximations technologies.
Kimeme, Cyber Dyne, 2011	Platform for multi-objective and multidisciplinary design optimization. Coupled, by means of scripts, with third-party software. Integration of custom optimization and/or analysis algorithms. Graphical design environment for problem definition, analysis and visualization of the results. Software network infrastructure to distribute the computational load.
MDICE, NASA, 1998	Multidisciplinary Analysis. Interface with commercial software for computer aided design, grid generation, computational fluid dynamics, computational structural dynamics. Visualization tools. Computing environment for the concurrently and cooperatively operation of many computers.

ModelCenter, PhoenixIntegration, 1998	Visual environment. Workflow graphically constructed. Data Fitting. Quick wrapping of batch mode programs into the modeling environment. Up to 30 optimization algorithms with definition of objectives, variables and constraints.
modeFRONTIER, ESTECO 1998	Multi-disciplinary and multi-objective optimization and design environment. Coupling to many existing computer aided engineering tools. Post processing results analysis. Visual environment. Simultaneous use of simulation software on different machines.
Nexus, iChrome, 2011	Linking to a list of third party commercial tools. Plugins for specific custom analysis tools. Trade-off, design of experiments, statistical analyses, response surface and metamodelling studies. Multi-objective optimisation algorithms. Visual environment.
OptiY, OptiY e.K., 2005	Multidisciplinary design environment. Providing direct and generic interfaces to many CAD/CAE systems, intern codes and externs programs through predefined interfaces. Graphical workflow editor. Modern optimization strategies, probabilistic algorithms for uncertainty, reliability, robustness, sensitivity analysis, data-mining and meta-modeling.
OPTIMUS, Noesis Solutions, 1996	Process Integration and Design Optimization software. Design of experiments and response surface modeling for design space exploration. Visual environment. Graphic workflow editor.
PASS, Desktop Aeronautics, 2005	Applicable to Aircraft Design. Rapid analysis coupled with optimization tools. Wide range of appropriate, real-world constraints. It is built on a modular, extensible framework that allows for the implementation of higher-fidelity analysis codes into the conceptual design process. Visual environment.
HyperWorks - HyperStudy, Altair Engineering, 1999	Design of experiments. Meta modeling approximations. Collection of single and multiobjective algorithms. Stochastic studies. Post processing and Data Mining. Parameterization of analysis models.
VisualDOC Vanderplaats Research and Development, 1998	Multidisciplinary design, optimization, and process integration software. Optimization, design of experiments, response surface approximation, and probabilistic (robust and reliability-based) analysis. Integration of virtually any CAE analysis software. Graphic workflow editor.

From the point of view of this research, the point of interest in these systems are their capacity for tool integration, levels of system automation, and process flexibility. Optimization itself is not the objective research rather automation of synthesis creation.

2.3.5 MDO System Specifications and Lessons Learned

MDO excellent for discipline integration, and search for optimum solutions within certain bounds but is a solver not an approach for AI. It is a lower-level attribute that could be employed but is not the driving core of a flexible, multi-problem agent. It would be a tool of a greater actor just as it is for the current designer. However, MDO is a fundamental tool of the designer and must be accounted for in any new design approach or system. As part of the review of literature on MDO, specific focus and attention was applied on identifying the MDO tool / system requirements for MDO in aerospace. In the following section, specifications are summarized.

2.3.5.1 Automation

Automation in paramount, as many systems as possible should be automated. This includes automated data transfer between and execution of analysis, including high-fidelity.[136] The system should automate or support the automation of the repetitive elements of the MDO process and design.[72, 137, 138] Furthermore, automation should apply to the pre and post processing of data as well as its transfer between tools.[138] Automation is the key to reducing time-to-market and cost, and increasing productivity.[44]

2.3.5.2 Early Concept Definition

A system should suggest an initial concept construct. The system should assist the designer in identifying a proper starting point (initial concept) for the next design sequence.[47]

2.3.5.3 System of Systems (vehicle-of-vehicles)

Systems-of-systems represent a challenging as each component system can have its own requirements and function. An optimal system-of-systems may result in non-optimal subsystems; however, this is not necessarily a negative. When addressing system-of-systems, it is necessary to ensure to address that the system-of-systems configuration is not constant, it can dynamically vary with time.[139]

2.3.5.4 Multiple Concept and Design Phase Applicable

A system should be flexible such that it can adapt to different design cases and design phase.[137] This includes flexibility in configuration and design phase process requirements.[138] In particular, as geometric modeling is critical to many MDO systems, the geometry module should not be a limiting factor to concept applicability. It should not limit application to traditional configurations.[72] Fundamentally, "*...it should be possible to design any type of aerospace vehicle using any (appropriate) methodology...*".[115]

2.3.5.5 Tool Integration and Distributed Computing

A design tool should be capable of integrating design tools and in particular distributed design tools.[136-138] This includes commercial off-the shelf, in-house tools, and legacy systems.[72, 128, 138] Integration should occur in a user friendly fashion.[128] As part of integration, the system should grantee proper data handling/correctness between the various tools.[138] Additionally, there should be no limited to the number of integrated systems, modules, or methods accessible to the system.[115]

2.3.5.6 Variable Fidelity

Generally, the literature suggests fidelity variability. Systems should incorporate both high and low fidelity analysis tools. [72, 115, 137, 138] It is additionally suggested to incorporate automated fidelity variation as required.[72]

2.3.5.7 Robust

Robust system framework is necessary.[72] Automated design (MDO) tools can be complex and brittle. By their nature they are established to address a specific problem and generally cannot venture far, in terms of configuration evaluation, from the initial problem definition and is limited to the constraints imposed in problem setup.[140] As such, systems should allow for easy interactive user control and modification of the optimization problem setup.[128]

2.3.5.8 Transparency

It should go unsaid that a system should be transparent. However, many systems are not. In particular the design process workflow should be visible.[72]

2.3.5.9 Geometry

Optimization is heavily dependent on the geometry being defined. As such, a MDO tool should include parametric geometry generation and agile manipulation techniques.[136, 141] Furthermore, the geometry model implementation should not limit the configuration applicability, in particular it should not limit application to conventional configurations.[72]

2.3.5.10 Visualization and Solution Exploration

A critical component of a solution package should be a visualization capability. Visualization of both the solution, solution space, sensitivities, and geometry should be available.[72, 142] Visualization should be in an automated fashion.[47] It is necessary to support the decision processes.

Significant to any design processes is the identification of the best solution. A systematic approach to design space exploration is necessary to improve the design process.[122] Additionally, an approach to identify why a particular solution is superior to another is needed [128] In particular the incorporation of tools for risk assessment and mitigation is needed. Mathematical optimal solutions are not always the best or correct solution.[136]

2.3.5.11 Software independent

Naturally, it is recommended to have the system software independent.[115, 140] That is, the system should not depend on any one software, especially third party software. By retaining software independence, the user maintains more control and reduces potential cost.

2.4 Selected Design Systems

In the following section, a consideration is given to specific more modern representative systems that resemble or are of interest to the current research. In particular, they are considered for their approaches to and application towards the conceptual design and design automation or more importantly, automation of design synthesis.

2.4.1 AIDA: Artificial Intelligence supported conceptual Design of Aircraft [47]

AIDA is an example case of an AI-KB approach to conceptual design. AIDA addresses the early concept definition phase of the conceptual design. Many conceptual design tools focus heavily on MDO and do not address well the initial concept definition and exploration phase (circa 2004), where in the concept itself is not necessarily even defined yet. AIDA attempts to define concepts for the early conceptual design through the application of AI and investigate the usefulness of various AI techniques in application to such a situation.

The solution logic is formed around addressing sequentially the author identified design cycle phases: suggesting, simulating, evaluating, proposing modifications. It is a modular system; it consists of four modules. A case-based reasoning (CBR) module for initial concept generation. The Function module that utilizes rule-based reasoning (RBR) techniques and sensitivity analysis to refine the initial concept into a feasible concept. Geometrical module to generate a CAD model of the concept through feature-based techniques and constraint-based modelling. Lastly, the central user interface, which controls system integration and data transfer. The system utilizes early conceptual design level methods as seen in the classic text references such as Torenbeek [13] and Roskam [96]. On consideration of the system, Rentema states that the system is "... *useful for 'configuration design' type of design tasks, but is less suitable for innovative and creative design.*" This is inherent in its dependency on established rules and concept elements in its libraries. Additionally, Rentema notes that such an implementation (CBR and RBR based approach) requires significant experience and pre-processing effort in order to populate and operate the system's data and knowledge bases.

2.4.2 Aircraft Design Automation and Subscale Testing [143]

A framework for automating the design and manufacturing process of Micro Air Vehicles (MAV) was developed at Linköping University by Lundström [143]. The goal of the system is described as " *... to find a method for MAV design and optimization from a holistic viewpoint, i.e. not a method for optimizing single subsystems, such as motor or propeller, but a method that embraces all disciplines of MAV design.*"[143] Additionally, Lundström identifies two key drivers. They are the utilization of off-the-shelf components where possible and geometric shape optimization in view of aerodynamic properties, internal component layout, and stability criteria.

Like many design systems of its type, the system is modular. The systems control interface is through a Microsoft Excel spreadsheet. Disciplinary and component subsystems formulate the rest of the system. These include a geometry model, an aerodynamic model, and an off-the-shelf propulsion parts database. The subsystems are integrated through modeFRONTIER, a PIDO system. Design automation occurs through the use of an optimizer. The optimization routine is sourced from modeFRONTIER. It utilizes a single-objective and a multi-objective GA for optimization. A dual-stage optimization routine is enacted. The geometry and aerodynamic models and analysis can be selected as either high or low fidelity. Low fidelity techniques are handled within Excel. A high-fidelity geometry selection incorporates Catia; a high-fidelity aerodynamics selection utilizes PANAIR, a panel method code. The output of the system is a CAD model and part list. The CAD model can be passed to a 3D printer for manufacturing. Distributed computing has also been demonstrated. Additionally, Lundström notes that the system could be considered a hybrid system. That is, it employees both CI and KB methods. He states that it employs heuristic knowledge within the geometry tool expressed as rules and constraints. Additionally, the system incorporates a GA optimization routine, which is categorized as CI.

One of the distinguishable features of the system is its dual fidelity optimization routine. It was identified that for high fidelity optimization, the optimization routine would generate an excessive number of non-feasible solutions. To rectify the problem a dual stage optimization approach was implement. The optimization processes were separated into a low fidelity stage and a high-fidelity stage. The low fidelity stage solutions, which have effectively been filtered for feasible solutions, are used as a starting point in the high-fidelity stage. The routine proved to be robust and user friendly.

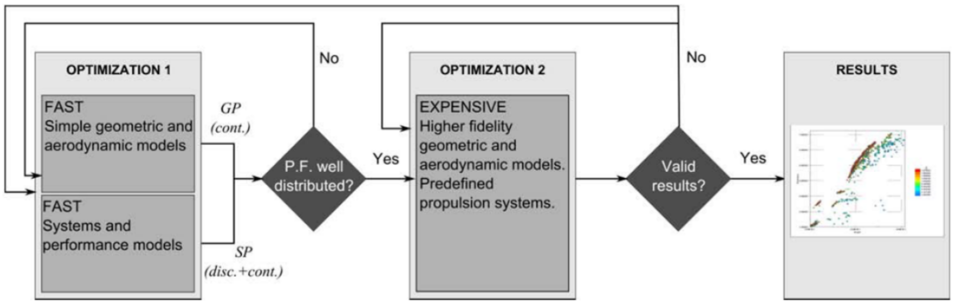

OPTIMIZATION 1
FAST
Simple geometric and aerodynamic models
FAST
Systems and performance models
GP (cont.)
SP (disc.+cont.)
No
P.F. well distributed?
Yes
OPTIMIZATION 2
EXPENSIVE
Higher fidelity geometric and aerodynamic models. Predefined propulsion systems.
No
Valid results?
Yes
RESULTS

Figure 2-6 Dual-fidelity optimization process [143]

The system demonstrates a class of AI hybrid (KB+CI) systems. The general system is example of very standard approach to tool creation found in literature. Multiple separate tools are integrated with a PIDO tool and optimization conducted by means of a CI algorithm, frequently an evolutionary type such as a Genetic Algorithm. Additionally, the system demonstrates a more unique approach that is less common, which is the integration of an off-the-shelf parts library. The system demonstrates a solution approach to handling the difficulty in feasible solution search, employing a dual-fidelity approach. Similar approaches are taken in other tools but with variation in the optimization algorithm. Furthermore, the concept for rapid prototyping, with a demonstration of output to 3D printer for manufacture and subsequent testing is an interesting concept.

2.4.3 GLADOS [140]

Genetic Learning Automated Design Optimization Software (GLADOS) "... *represents a flexible evolutionary algorithm based architecture intended to allow for the generation of conceptual or preliminary design stage aircraft designs without any human beings in the loop*." [140] The researchers' objective was to develop a system to automate portions of the design process to reduce human involvement and thereby reduce cost and time to completion. The original motivation for the system was to address the problem of multi-variant high modularity complex system design through the application of an evolutionary algorithm. They identify and propose a solution to three identified issues. They attempt to address design space biasing[5], commonality or modularity in complex system design, and the rigidness inherent in many MDO architectures[6]. Addressing these issues resolves into "*…the three most important traits are being able to naturally search a much larger section of the design space, enable straightforward development of high commonality and modular systems and be expressive enough to be capable of recursion and therefore meta-level self-improvement.*"[140] In [140] the authors provide a description of the ideal concept and an account of a significantly reduced proof of concept. Of principle interest is the ideal system concept.

The concept is a recursive tool that can generate potentially feasible design concepts from a concept component library, populate the design variables, evaluate the design suggestion, and identify potential solutions to be carried over to the next design evolution sequence. The authors summarize GLADOS as:

> *… a large assembly of component, sub-system, sub-discipline and operational level analysis modules wrapped in an evolutionary algorithm framework that ultimately selects designs based on simulated natural selection, with fitness being assessed by operational simulation of each trial design.[140]*

The system concept consists of a: concept library, trial design synthesis subsystem, modeling and analysis framework, fitness module, evolutionary framework, and artificial intelligence and machine learning. The GLADOS concept is built around an evolutionary routine for initial concept generation and subsequent evolution. The concept employees a warehousing approach where in an updateable concept warehouse of existing design elements can be queried and formulated into an initial candidate design for further analysis and optimization. The generated concept constructs are modified for correctness and filtered by evaluation of feasibility by the trial design synthesis module. Each trial design is analyzed and modeled by a modeling and analysis framework that includes optimization; the framework is characterized by a tiered analysis processes where in each tier can incorporate greater design and analysis fidelity. A fitness module quantifies each trial construct based on some evaluation criteria, which is utilized in the evolutionary algorithm to populate the next evolution. A general inclusion of AI and CI methods are described as included for system efficiency improvement including approximation routines as well as a capacity for self-population of the concept library and analysis association. Self-population and analysis discernment through a recursive approach is a key trait of the system concept.

The concept presented is interesting for its non-standard approach to the design automation problem. Much of literature, for the optimization problem, shows a standard approach of problem

[5] Design space biasing is the event of experiencing or implementing bias into a proposed solution set due to underlying experience, favoritism, or exposure to certain solutions, thereby not considering potential alternatives.
[6] MDO architectures are generally setup to address specific problems, configurations, and solution space boundaries/constraints, which restricts the architecture's applicability; they are not generic design architectures.

specific code formation through PIDO systems integrating high fidelity third-party analysis modules. The GLADOS concept presents an in-house approach to system integration with a user created tier-based refining concept definition and analysis approach. When compared to the concept for Aircraft Design Automation and Subscale Testing, the GLADOS concept proposes to address initial design construct feasibility through a dedicated evaluation system, similar to AIDA. The tiered optimization process is of note; tiers of analysis/optimization where in the user can control the design freedom of each tier such that proper a natural evolution of design refinement (similar to the conceptual to preliminary design tasks) can occur and thereby minimize computing power required is a notable approach.

2.4.4 Daphne [77, 80]

Daphne is distinctly different from most other design tools. Daphne is a virtual assistant to support high level design of distributed satellite missions (DSM); it is quite possibly the first of its kind.[77] The objective of Daphne is "*… to help system engineers reduce their cognitive load when exploring large tradespaces for DSMs by providing them with easier and timely access to relevant information."* [77] According to Virosi [77], Daphne has a capacity of natural language interfacing for information quarry in addition to tradespace exploration tools such as scatter plots, model inspection and explanation, and data mining.

An illustration of Daphne architecture is shown below. It has a user interface (a web front-end), an architecting element (Daphne Brain) that controls user requests, software snippets (Roles) that utilize the Backend and Data Sources to obtain the answer to the user's quarry. Backends are code elements that compute the information requested by the Roles using the data acquired from the Data Sources. Data sources include three databases: an Expert Knowledge Database, a Design Solutions Database, and a Historical Database.[77]

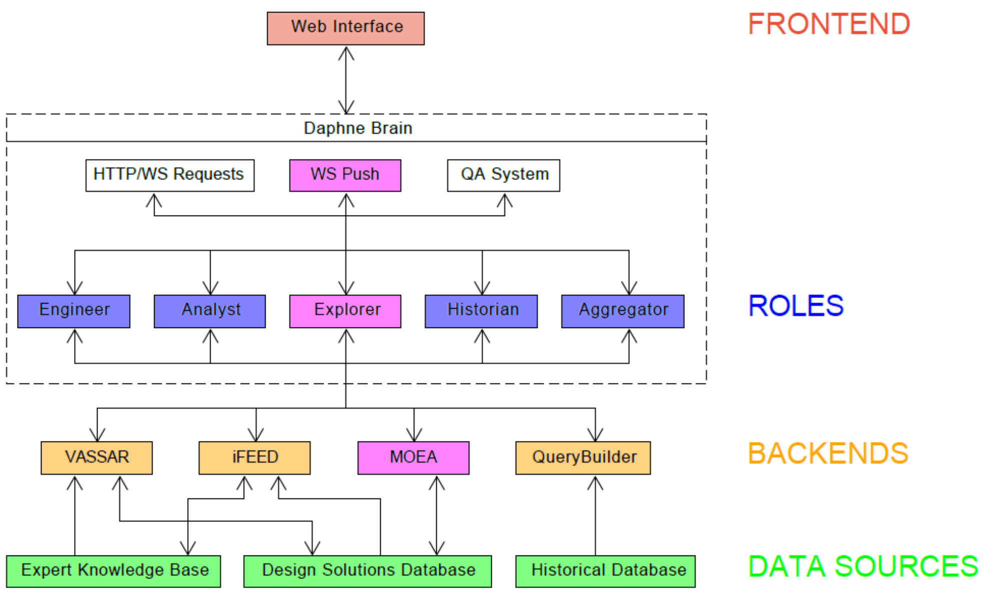


Figure 2-7 Daphne architecture [80]

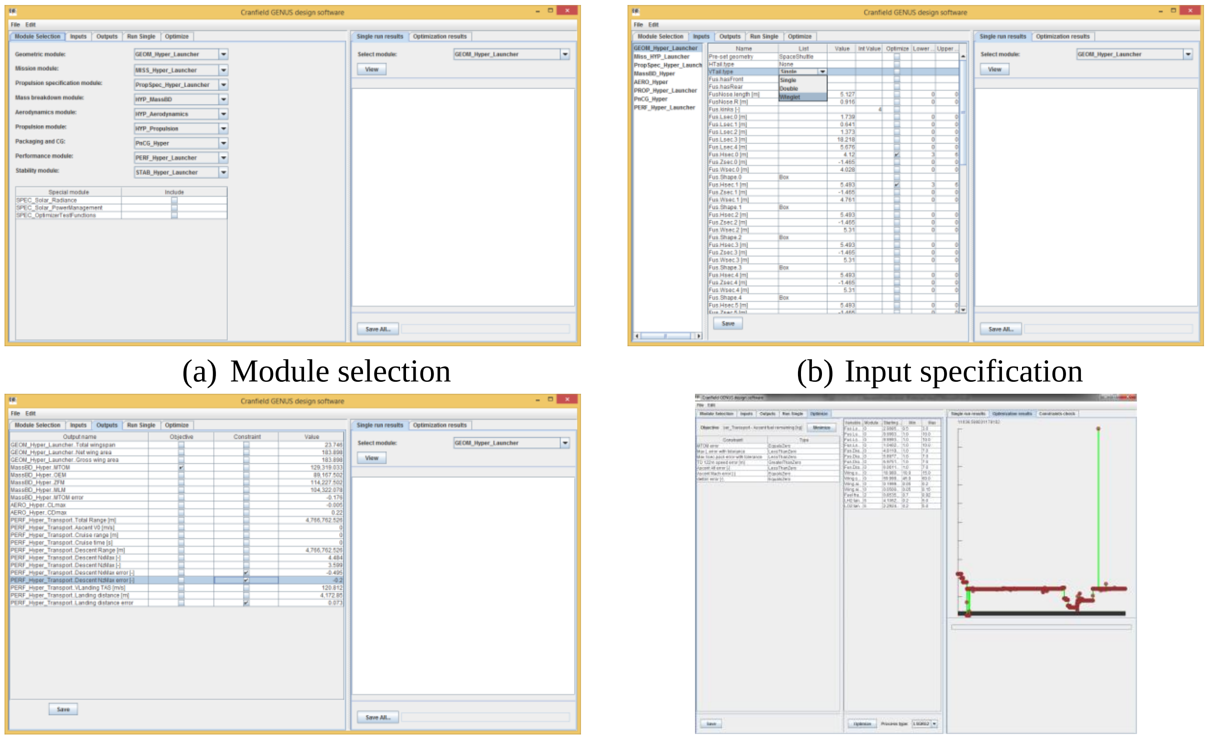
Daphne represents the interesting design aid of the virtual assistant. If one recalls the principal motivation for this research—effectively a virtual assistant/peer—Daphne most resembles it at least in general practice. In light of the research problem being addressed (automation of design synthesis), in comparison to the Daphne architecture, this research addresses a task within the "Engineer" role above. The "Engineer" role's function is to "Evaluate new architectures" and "Answer questions about architecture performance and cost."[80]

2.4.5 GENUS [115]

Developed at Cranfield University by Szirozák [115], GENUS is a design framework that *"... provides a sufficiently generic platform that can be utilized for the conceptual level design of specific classes of aircraft, including, but not limited to hypersonic transports, space launchers, blended-wing-body and solar-powered aircraft.*"[115] The motivation for the development of the system was to assist the educational system, specifically university students and researchers. The author notes that in university, programs have students spend a significant portion of their project time on method development and integration with insufficient time to actually appreciate the end result or conduct specialized feature analysis.[7] System requirements included: modularity, expandability (unconstrained fidelity level and method count), flexibility (capacity to address any vehicle; generic synthesis tool), independence (software independent, non-proprietary, source code language with significant longevity prognosis), sustainability (easily maintained and expandable), and performance (reasonable performance on a standard desktop or laptop).

GENUS is a design environment to provide a modular, flexible framework both for designers to use existing and for researchers to develop new methods for aerospace vehicle design. Fundamentally, it is a shell environment similar to the PIDO tools. That is, it itself is an integration and optimization environment where in the user prescribes, through a transparent user interface, the analysis modules, inputs, internal variables, objectives, and constraint. The system also provides the results visually through the GUI and as text file. GENUS is based on a modular (library, warehouse equivalent) framework. Modules are divided by the essential modules and the non-essential or "special modules". There are nine essential modules: Geometry, Mission specification, Propulsion specification, Mass breakdown, Aerodynamics, Propulsion, Packaging and CG, Performance, and Stability and Control. In addition to these there in the Atmosphere module. It has a clean GUI, Figure 2-8, and is programed in Java. It has capacity for a single run analysis and optimization. All iterative analysis occurs through the optimizer, there is no indication of inherent automated trade study or sensitivity analysis. The analysis process is linear and rigid, occurring in the order of the methods shown. In summary, GENUS provides a transparent conceptual design method integration environment with single point analysis and optimization capability; automation tasks include method integration through data handling and design space search by the optimizer.

[7] As a former teaching assistant for the undergraduate aerospace engineering senior design course, the author can concur with this sentiment.



(a) Module selection (b) Input specification

(c) Output results (d) Optimization setup and results

Figure 2-8 GENUS graphical interface [115]

2.4.6 AVD^{DBMS} [86, 87, 89]

Aerospace Vehicle Design Database Management System (AVD^{DBMS}) was developed at the University of Texas in Arlington by Gonazlez [86], Omoragbon [87], and Oza [89]. The system was developed to address two issues. First, as Gonazlez states it is *"... a methodology for the composition of complex multi-disciplinary systems (CMDS) through the automatic creation and implementation of system and disciplinary method interfaces*."[86] It is an environment to alleviate the difficulties in synthesis architecture creation and to diverge from the classic monolithic system by assembling the synthesis architecture per problem definition. According to Gonzalez, it attempts to bridge the gap between classical monolithic systems and the shell integration systems (PIDO). The second reason it was conceived was to assist in technology forecasting and portfolio definition—the object of Oza's work—as to evaluate many potential concepts, requires a robust, problem specific generation architecture. Within the literature of aerospace vehicle design and design automation, this approach is distinctly different; rather than focusing on automating the design search as in optimization routines, this approach automates portions of the creation of the design tool itself, the synthesis architecture (this does not preclude that optimization does not occur within the architecture either). AVD^{DBMS} interface is MS Access, the system generates synthesis architectures as MATLAB scripts.

The concept behind AVD^{DBMS} is a decomposition-recomposition approach. The idea being that systems (legacy monolithic codes) and vehicles can be decomposed into their base constructs (process, methods, hardware, etc.) and placed into a repository where from a new system can be assembled from these parts and part associations to solve a new problem. The system's foundations are its databases and knowledgebase repositories. The system consists of a reference library, variable library, methods library, process library, vehicle library, and the actual system interface for problem architecture definition and assembly (referred here as Main System). Each has a GUI interface for access and modification. Each library contains the decomposed elements according to its name. The methods are associated to hardware applicability. The actual architecture definition and assembly process occurs within the Main System GUI. The process is divided into four stages. These stages are matching, selecting, arranging, and generating. Through these stages the user selects a project vehicle, a project process, defines the trajectory profile, selects analysis methods and resolves any system conflicts such as multiple disciplinary method associations per hardware through the definition of method constraints. From these selections the system assembles the methods and process into a syntactically correct sizing tool. The user then is free to use the resulting tool to solve their specific problem. Note that the system does identify input, output, and interdisciplinary variables, however all input value defining, synthesis tool execution, and post processing occurs outside of the system and by the user. Once the architecture is generated, the operation and actions of AVD^{DBMS} are complete.

AVD^{DBMS} represents a class of synthesis tools that itself does not solve the design problem but rather generates the tool that is used to solve the design problem. AVD^{DBMS} has been created to provide the designer with a tool of tools; it is a tool to create tools that are tailor-made to the exact problem at hand with the fidelity and robustness as determined by the user. It is different class of automation; it automates the creation of the synthesis architecture rather than design automation through automating the design search. Although this system represents a promising approach to automation, it is noted that the system does not directly contain post processing, an input/output interface, and is limited in vehicle decomposition level and limited in its process application. As noted, this research is a continuation of the synthesis design effort at the AVD Lab at the University of Texas at Arlington as represented by AVD^{DBMS}. This system and the work by Gonzalez [86], Omoragbon [87], and Oza [89] will be referenced more in the following chapters.

2.5 Summary and Specifications for Future Systems

2.5.1 Summary and Discussion

This research began with AI and design peer being identified as a probable solution to dilemmas within design. However, a true design peer was determined infeasible for a single researcher and as such a perceived necessary element, design automation (automation of synthesis) was identified. This chapter has presented a review of design tools with particular emphasis on design automation and the necessary components, in addition to a consideration of select representative systems. The result is the identification and consideration of toolsets in literature and the approaches to the design problem, in particular those applying automation.

Aerospace synthesis design approach is categorizable. A classification scheme was presented. From the classes, of note are two classification groups: text-based systems and computer-based systems. The text-based systems are representable by the classical design texts, by definition are not automated, and represent libraries of knowledge and early conceptual design analysis methods. The second classification set are computer systems that automate part of the design process. Two types are distinguished, the monolithic and non-monolithic system. The classical computer systems tended towards monolithic nature where in, though with highly impressive knowledge integration and accuracy, they were compiled upon as new capability were added leading to lack of maintainability, modifiability, and applicability to new concepts. The second type, non-monolithic, are generally specifically design modular approaches where in the modules (including third-party software) are integrated through some integration scheme and can be specifically developed for the problem at hand. Within these system MDO has been a critical component as well as focus in literature on design automation.

Within aerospace vehicle design literature, design automation generally infers MDO. It is the process of automating the process of design refinement through solution space search. MDO is a significant focus within the literature but is not the only solution nor necessarily the correct solution in all situations, though from literature one would not be wrong to assume it were given it is so widely applied and touted. Optimization tasks are frequently time consuming, can have massive software and hardware requirements, and require expertise in and of itself to set up properly. In effect, there has been a trade of one problem (the monolithic design codes) for another. To assist in MDO execution process, many integration and process control software have been developed both commercially and as open-source software; naturally, these systems are widely used. A selection of systems available was given.

A conclusion that can be derived from literature is that there is a lack of focus on the early conceptual design within tool development, specifically the initial problem gestation phase and initial potential concept solution selection. Many systems/research/tool development jump quickly to optimization without considering if the object they are optimizing is even the right choice or starting point. Some systems attempt to address this through evolutionary algorithms and concept part libraries but not all. This is similarly reflected in the literature on road mapping and program planning, systems are advised but without numerical proof of why they should be in the first place. There is a need to address the early conceptual design phase and even the pre-design phase. As MDO has been praised for bringing better solutions and more accurate, higher fidelity approaches earlier into the design process, perhaps it is necessary to improve analysis tools in the parametric definition cycle of conceptual design and even into the pre-design operations as well. This calls for rapid concept exploration environment, that is truly generic in vehicle consideration, which can provide analysis capability of fidelity levels prescribed by the user and be operable quickly. This leads to the specification of a generic conceptual design decision support environment.

2.5.2 Specifications for a Future System

As noted, this research is in part a continuation of the work by Gonzalez, Omoragbon, and Oza. AVD^{DBMS} was a proof of concept with its own limitations. AVD^{DBMS} demonstrated the ability to have an environment in which, through user interaction, synthesis tools could be generated to address a specific problem. However, the system had an inherent limit to the complexity of the problem and as such an inherent limit to the cases in which the tool could be applied.[87] Furthermore, the system exists in an MS Access environment, which does not lend well to continuation, in particular towards AI—a primary research objective of the local research group. (As a result, the research deliverable here in presented has been created to address these issues.) As such, many of their identified solution construct requirements remain. In addition to those identified, several are added here in order to address some of the short comings of AVD^{DBMS} and to advance the concept to a more capable place with potential for increased automation and eventual adaptation into a greater cognitive framework. Based on the synthesis system review and addressing issues identified in AVD^{DBMS}, the following are identified as the primary guidelines and requirements for a next-generation synthesis capability.

2.5.2.1 General Design System Guidelines

» Flexibility: modularity to handle various fidelity levels, unique concepts, and unique configurations.

» Expandability: ability to and easy implementation in the expansion of the underlying framework and capability when new data, knowledge, and processes are identified and require addition.

» Transparency: transparent to the user of the operation of processes and systems, the methods, underlying knowledge, data, etc.

» Rapidity: quick turnaround, able to adapt and keep up with a rapid environment and quick turnaround deliverable times; minimal time of operation to output.

» Operability: low user learning-curve.

» Sustainability: system should be based on a coding language likely to continue into the foreseeable future.

2.5.2.2 System Specific Specifications and Guidelines

The following specifications are shared in the specifications for AVD^{DBMS} by Gonzalez [86] and Omoragbon [87].

» Employ a decomposition-composition solution approach.

» Store and implement design analytical processes.

- Store vehicle hardware concepts.
- Store and implement engineering disciplinary analysis methods.
- Associate hardware-method-process applicability.
- Assemble synthesis architecture.

The following specifications are added to those identified by Gonzalez, and Omoragbon.

- Architecture specifications occurs within a decision support environment (interface).
- Improved system transparency in both method / process specifications and architecture generation.
- Architectures should be assembled as fully contained scripts.
- Assembled architectures should be fully executable and distributable.
- Include capability for system-of-systems (vehicle-of-vehicles) consideration.
- Include capability for multiple design analysis processes association or a tiered processes approach (sub-processes within a primary process).
- Allows evaluation of results (data post processing and GUI return for assessment and inquiry).
- Allows for specification and generation of standard or user defined solution visualizations.
- Increased automation or capacity for automation of architecture generation process.
- Allow for porting into a greater system and allows expansion and integration of data mining and increased post processing towards a design recommender.

2.5.3 Document Outlook

The remaining document address the solution concept, the implementation of the concept, and verification and demonstration of the concept implementation. As noted in Chapter 1, this research is in continuation of the research endeavor by Gonzalez [86], Omoragbon [87], and Oza [89]. As such, much parallelism is drawn between their research and solution concept presented and developed in this research.

Chapter 3 SOLUTION CONCEPT

In this chapter, the solution concept is presented. A general solution concept for a general synthesis automated generation decision support environment is given. For detailed manifestation of the concepts presented here, see Chapter 4 Concept Implementation.

3.1 General Solution Concept

To address the requirements identified, the objective is to develop a decision support environment for the aerospace domain, specifically targeting the conceptual design phase. Within the frame of the decision support tool, it is necessary to implement a framework for automated composable analysis architectures. That is, the system shall not be bound to any one vehicle concept or configuration, nor shall it be bound by the process or objective function definition. Furthermore, the tedious task of synthesis architecture assembly is removed from the requirements of the user. The user only need specify what to analyze and the constructs of how to accomplish the analysis. To accomplish this task, it is necessary to implement an auto coding approach. A top-level solution for such a system is illustrated in Figure 3-1. The solution concept is founded on a decomposition-composition approach. It is a non-graphical code assembly concept. The primary components of the concept are the composable complex system components, the synthesis generator, and system results. Fundamentally, the user provides a set of inputs specifying the vehicle to be analyzed, the process of analysis, the methods to use, and the output presentation desired. From these instructions, a synthesizer routine assimilates the necessary code elements, both engineering methods and code processing (data handling, method handling, etc.), and assembles the components into a functional synthesis architecture. The synthesis is executed as prescribed by the user and the results are processed and returned to the user according to the user's deliverable specifications. Each core component is summarized below.

» Inputs: A collection of user selections during software interaction. They are acquired through a GUI interface. They specify the components of the synthesis system as related to the engineering analysis problem.

» Complex System Elements: It is a set of libraries populated with the building blocks necessary to assemble a synthesis analysis code. It consists of three libraries: product, methods, and processes. These follow the complex system decomposition approach described below.

» Synthesis Generator: A collection of processes to assemble the base components (complex system elements) into a functional synthesis analysis code based on the user's inputs. Sub processes include input mapping, library queries, component gathering, and component assembly. The output of the process is a tailor made fully functional synthesis analysis code for the specific problem at hand.

» System Results: The system results group is a collection of processes to execute the synthesis code, archive the results (data), process the results, and return the results to the user through the DSS in a meaningful form. Part of the result return is the auto generation of meaningful figures for design evaluation and insight.

Each principal component of the solution approach is discussed in detail in the following sections.

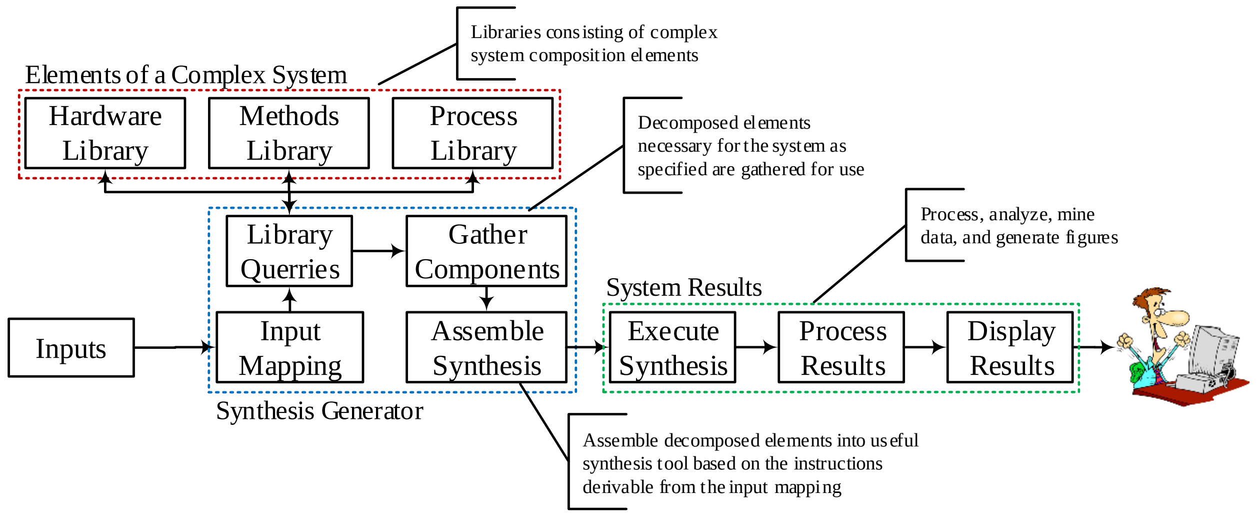

Libraries consisting of complex system composition elements
Elements of a Complex System
Hardware Library
Methods Library
Process Library
Decomposed elements necessary for the system as specified are gathered for use
Library Querries
Gather Components
Process, analyze, mine data, and generate figures
System Results
Inputs
Input Mapping
Assemble Synthesis
Execute Synthesis
Process Results
Display Results
Synthesis Generator
Assemble decomposed elements into useful synthesis tool based on the instructions derivable from the input mapping

Figure 3-1 General solution concept

3.2 Decomposition Concept

As stated, this research and development effort has been conducted within the evolutionary synthesis development arc of [84-87, 89]. In particular, this is a continuation and adaptation of the concepts laid down by Omoragbon [87] and Gonzalez [86]. The following general description is adapted from [87]. Note that in the following discussions the term complex system is used frequently. The term is used in two connotations. First as the system being decomposed and being labeled complex as it consists of the identified decomposition groups. The term is also used to refer to a system-of-systems or what one would perceive natively as a complex system such as an aircraft or ship. The discussion at hand indicates which construct is being used.

For a given complex system, a tripartite decomposition routine is enacted. A complex system is described by the product, process, and methods. Alternatively, the term complex architecture could be used. The product is the physical description of the complex system; generally, this is the vehicle of interest. A product is described according to what it is, what it does, when it does it, and requirements or limitations of operation. These conditions define four categories of a product—structure, function, operational event, and operational requirement. The process is the numerical and organizational approach to solving the problem, and the methods are the analytical, numerical, or empirical means and their dependencies to approximate a physical condition. Each category is addressed in more detail in the following sections.

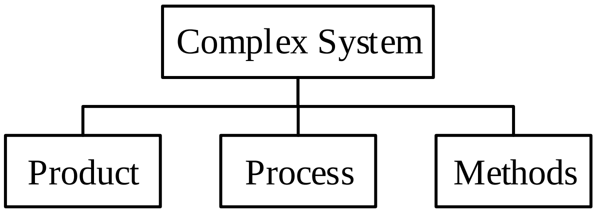


Figure 3-2 Three elements of a complex system

3.2.1 Product

The product is the complex system being considered. The system is defined according to structural subsystem (structural decomposition), functional subsystem (functional decomposition), operational event, and operational requirement. Omoragbon initially identified just three classifications (functional subsystem, operational event, and operational requirement), however, a fourth (structural subsystem) has been added to better handle the more complex situations (vehicle-of-vehicles) and complex mission and operation description.

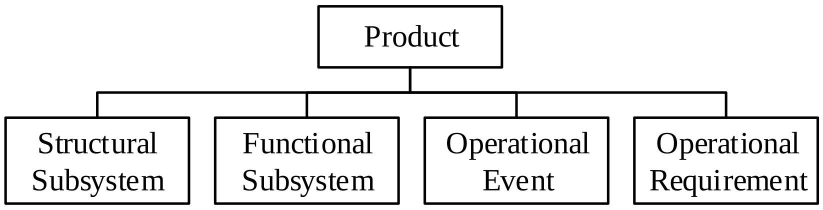


Figure 3-3 Product decomposition

3.2.1.1 Structural Subsystem

The structural subsystem decomposing (hierarchical decomposition) is a standard decomposition approach by parent-child system reduction. It is a mapping of the structural components and their structural subcomponents, continuing in subcomponent layer refinement, as necessary. For the solution concept discussed herein, the structural hierarchical decomposition scheme is included in order to expand system capability to include the vehicle-of-vehicles case. The vehicle-of-vehicles is a specific condition of the system-of-systems concept. A system of systems is " *... a set or arrangement of interdependent systems that are related or connected to provide a given capability.*"[144] Structural decomposition decomposes a system into its subsystems and the subsystems into their elements. Likewise, a vehicle-of-vehicles is a complex system where in the parent vehicle is composed of sub-vehicles, which are composed of systems. A parent vehicle can have any number of child vehicles. Furthermore, each vehicle is its own complex system in the sense of the decomposition approach being discussed. Each one is its own complex system with its oven product, process, and method decomposition. The complex system-of-systems (vehicle-of-vehicles) structural hierarchical decomposition in general is illustrated in Figure 3-4. In terms of an aerospace vehicle, this decomposition approach is illustrated in Figure 3-5.

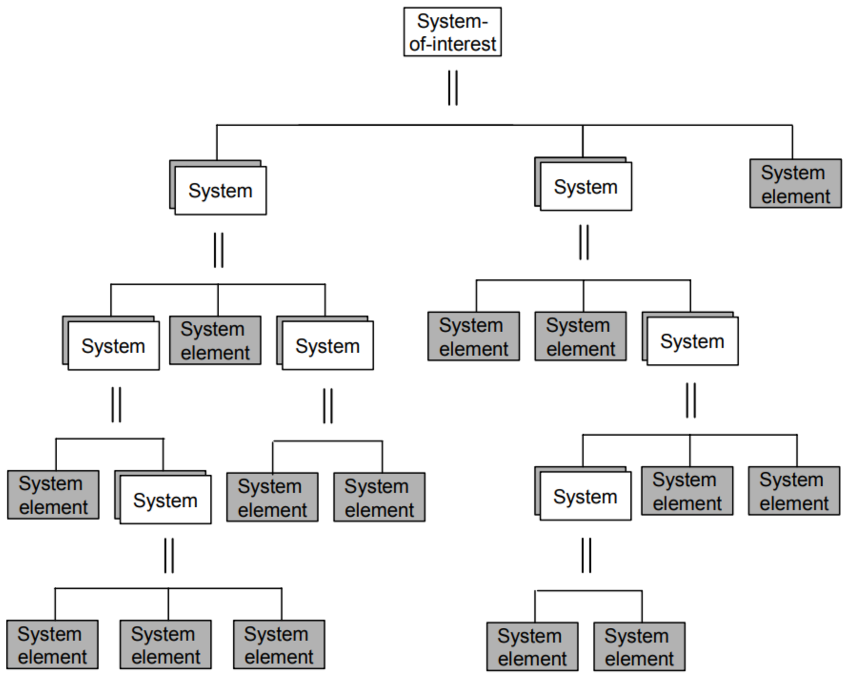


Figure 3-4 System-of-systems structural tree decomposition [145]

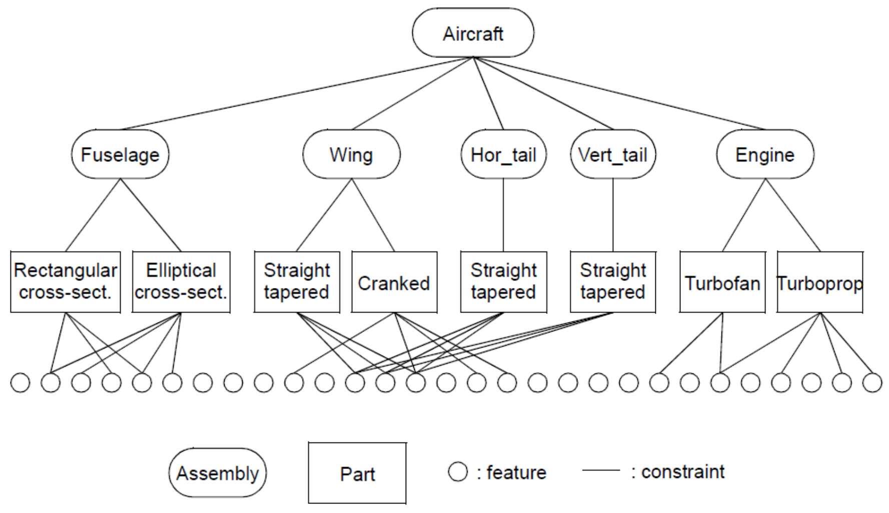


Figure 3-5 Illustrative example hierarchical structural decomposition [47]

3.2.1.2 Functional Subsystem

The functional subsystem represents the decomposition of a complex system's product by component function. Functional decomposition is the association of function (purpose) to the systems element (hardware). Various functionality includes lift source, drag source, thrust source, volume source, etc. Various function categories are illustrated in Figure 3-6. Within these categories, one or more elements could be associated. For example, a thrust source can be airbreathing, but within airbreathing there are many design options, the element could be a turbojet, a turbofan, or even a ramjet or depending on the design criteria.

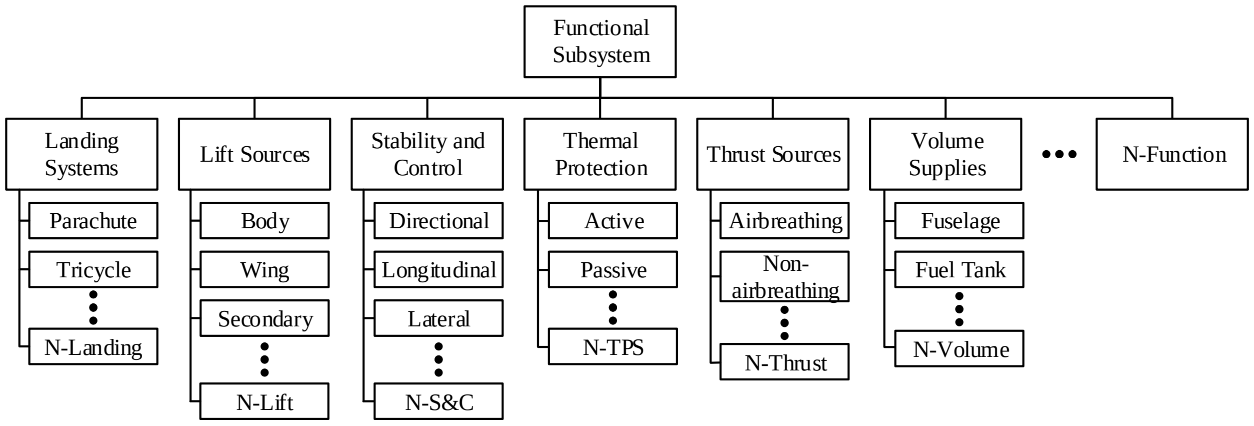


Figure 3-6 Functional subsystem decomposition categories adapted from [87]

Omoragbon intentionally includes a functional decomposition approach to allow for a synthesis system to better address, trade, and evaluate factors such as acquisition, TRL, and maintainability. Furthermore, through functional decomposition, a complex system is describable as a shell construct with certain attributes that are populated per unique design case. They are product design details of which can be populated and traded dependent on the user's intentions. Such an approach allows for more readily the inclusion of more detail and analysis-oriented approaches within the pre-conceptual design phase (road mapping, architecture and program planning, etc.) or easier inclusion of manufacturing, servicing, sustainment and other like constraints that manifest much later in the product life cycle but are integral to the success of a program long-term.

3.2.1.3 Operational Event

The Operational Event describes the vehicle by environmental and operational conditions of its use. The classification and description are in regard to the total system operation and is distinguishable from its hardware. The Operational Event category is subcategorized by mission type, flight profile, speed range, gravitational body, and altitude range.

3.2.1.3.1 Mission Type

A vehicle has an objective, that objective is accomplished through the execution of a specific mission. The mission type is a label to describe mission and ultimately the objective of the vehicle.

A vehicle can have more than one mission type. For example, an advanced single-stage-to-orbit vehicle would perform a mission of (1) space access, (2) orbital operations, (3) Re-Entry, and, potentially, (4) point-to-point. These and other possible mission types are identified and described in Table 3-1.

Table 3-1 Mission types

Name	Description
Point-to-Point	Transportation of a cargo from one latitude and longitude to another, generally atmospheric bound but not necessarily so;
Space Access	Transportation of a good or service to space or the transfer of sufficient energy from one system to another allowing the recipient to reach space; the system does not have to achieve orbital conditions but could.
Sub-orbital	Transportation of a cargo to space but in a manner in which the vehicle or cargo does not have sufficient velocity to achieve orbit;
Re-Entry	Vehicular atmospheric entry to a body of influence from orbital conditions with a start condition outside of the effective atmosphere (space);
Orbital	Exo-atmospheric (space) operation at sufficient speed and energy for a vehicle to maintain an orbit;

** Space (outside of the effective atmosphere) for Earth is considered 100+ km.*

3.2.1.3.2 Flight Profile

The mission flight profile is segmented into its constituents. These are the classical flight profile components such as takeoff, cruise, climb, etc. It is common for a flight profile to be segmented into its core components as it can make communication and analysis simpler. The flight profile segments options are indicated below. These values are updatable and can be changed by the user if a particular one is necessary and not currently available. The focus is on atmospheric flight though elements can be readily expanded to include orbital operational elements.

» Warmup	» Taxi	» Takeoff	» Climb
» Cruise	» Loiter	» Dash	» Turn/Maneuver
» Descend	» Deployment	» Rendezvous	» Re-Entry

In the consideration of the multi vehicle case, a vehicle system can comprise of multiple distinct missions or flight profiles. It is possible for a component vehicle to have a different mission and objective than the parent or other secondary component vehicles. This is especially true in the case of two-stage-to-orbit (TSTO) or reusable launch systems such as the Falcon 9 or Falcon Heavy. In these cases, the total system (all component vehicles acting as a single vehicle) act on a single flight profile until the systems disengage and each execute separate flight profiles as fragmented systems but each with uniquely different objectives, requirements, or flight profiles.

3.2.1.3.3 Speed Range

The speed range defines the operational speed range experienced by the vehicle. A vehicle, within the definition of the problem, can have a combination of selections. For example, a hypersonic vehicle can experience subsonic, transonic, supersonic, and hypersonic conditions. At each speed condition, different phenomena can occur and as such, the vehicle’s complete description would have to account for this (atmospheric dissociation, shock formation, variation in aerodynamic center, etc.). The speed is an indicated for many physical phenome occurrences. The speed range values are indicated in Table 3-2. The speed ranges follow standard speed ranges experienced within atmospheric flight; naturally, orbital conditions could be considered as well.

Table 3-2 Speed range categories

Name	Speed Range (Mach Number)
Subsonic	< 0.8
Transonic	0.8 – 1.2
Supersonic	1.2 – 5.0
Hypersonic	5.0 – 10
High Hypersonic	10 – 25
Re-Entry	25 >

3.2.1.3.4 Gravitational Body

The gravitational body is simply the specification of the principal body of gravitation that the vehicle operates. The majority of aerospace problems are relegated to operating on Earth; however, there are cases where in the principal body is not Earth but other bodies such as Mars. As such, not to be constrained by the body of influence, this parameter is a necessary descriptor. The gravitational body is not only applicable to the consideration of defining the gravitational parameter but is also directly linked to the atmospheric model required. Atmospheric operation is a consideration discussed in the “Altitude Range” descriptor.

3.2.1.3.5 Altitude Range

The altitude range is defined through the atmospheric operation zones. The atmospheric zones are for Earth and are indicated in Table 3-3. Naturally, a vehicle can operate in any combination of atmospheric zones, generally in a continues form. A vehicle, within the definition of the problem, can have a combination of selections. Note, that the problem does not have to be limited to Earth, but for this case, it is used as the principle gravitational and atmospheric zone of influence.

Table 3-3 Operational altitude zones

Zone	Altitude
Troposphere	0 – 8 (14.5) km
Stratosphere	8 (14.5) km - 50 km
Mesosphere	50 km - 85 km
Thermosphere	85 km - 600 km
Exosphere	600 km – 10,000 km
Exo-atmospheric	100 km > (Kármán line)

3.2.1.4 Operational Requirement

The complex system exists and operates within limitations and requirements. The Operational Event category describes the vehicle by the environment and operational conditions of what it does whereas the Operational Requirement describes the limits in which the system is required to do what it does. The limitations of the system can be categorized by (1) *regulations*, and (2) *specifications*. Regulations are government restrictions or standards imposed upon the system. Examples are safety standards, emission standards, noise regulations, etc. The operational requirement specifications are additional conditions mandated upon the system that are not hardware, function, or regulation descriptors. Such mandates include human rated, vulnerability, survivability, propellant, manned, unmanned, etc.

3.2.2 Process

After the product description, the next element in the decomposition of the complex system is the process. The process is the analytical process to solve a given problem. A process is specified independently of the product; the process description has no indication or direct connection to the product. It is product independent. There are two types of processes: primary process and secondary process. The primary process is the driving instructions for the total vehicle analysis. It encompasses all process for a given complex system. The secondary process is a process that executes within the operations of the primary process. There can be any number of secondary processes but only one primary. Each process is decomposable by its system elements and its disciplinary elements.

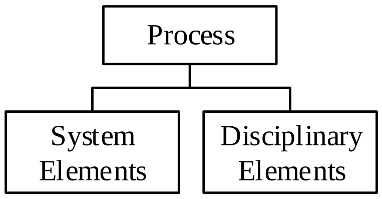


Figure 3-7 Process decomposition categories

3.2.2.1 System Elements

The system elements describe the mathematical components of an analysis process if that process has some objective function. The objective function specifies the mathematical criteria for convergence or optimization. However, it is not necessary that a process have an objective function; the absence of an objective function indicates a process that is not iterative. If the process is not iterative, then there are no decomposable system elements. The system elements are independent variable, dependent variable, and objective function. Each is described below.

» Objective function: the objective function is a mathematical expression that specifies the process' criteria for satisfactory termination. This is usually applied as a convergence or optimization criteria. The process continues until the objective function is either satisfied or determined unattainable. The objective function consists of dependent and independent variables.

» Independent Variable: the objective function's variables that are independent of the analysis and are known (guessed) initially. They are the values searched for to converge or solve the analysis process.

» Dependent Variable: the objective function's variables that are computed through the operations of the process and are an output of the analysis, as such they depend on the value of the independent variable. They must be output by the disciplinary operations as specified by the disciplinary elements.

3.2.2.2 Disciplinary Elements

In addition to the objective functions as described previously, the process is composed of disciplinary events and their order of operation. The disciplinary elements are the descriptive elements that define the operational order of the process and categorize the analytical sub processes. Each is described below.

» Disciplinary Event: the disciplinary events are the specification of an analysis set execution that is categorized by a topic of analysis. Classically, the disciplinary events are the categories aerodynamics, weight and balance, propulsion, geometry, stability and control, etc. However, they are not limited to these and can be varied or added to depending on the process and the overall topic being addressed. Disciplinary events can be constructed to have standard variable outputs that the encompassed analysis is required to generate.

» Disciplinary Order: the disciplinary order of operation is the specification of the order of process disciplinary events. The specification is linear; however, non-linear attributes are accounted for through the internal operands of the disciplinary event.

3.2.3 Method

The method group is an assembly of descriptors identifying a particular analysis element. The analysis element is what one would consider as an engineering analysis method. It can be numerical, analytical, or empirical. A method is described by its product applicability, its variables, and its analysis.

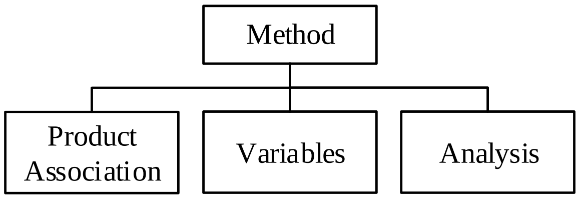


Figure 3-8 Method decomposition categories

3.2.3.1 Product Association

The product applicability follows the functional decomposition described in section 3.2.1 Product. A given method is applicable or associable to a particular hardware, hardware function, mission, or operational event or requirement and is describe through these conditions. The method is associated to the hardware, function, mission, operational event, and operational requirement in the same manner as the product. Fundamentally, this is necessary for proper method selection and system assembly and operation.

3.2.3.2 Variables

The variables category encompasses the variables that define the methods inputs, outputs, and constraints.

» Input: They are the variables required as inputs (known conditions) by the method in order to operate properly and return the output.

» Output: They are the variables that are solved for within and returned from the method. They may be required by other methods within the same discipline/discipline process or other disciplinary methods.

» Constraints: The constraint variables are the variables that, if any, constrain the application of the method to a specific variable value condition. An example would be Mach Number; a subsonic method may be only applicable during Mach Numbers of 0 to 0.8 for example, and so that method would be constrained to a given Mach Number range.

3.2.3.3 Analysis

The Analysis block contains the elements that describe the method according to process discipline, assumptions, and analysis body. The process discipline is exactly that, it is the specification and subsequently the mapping of the method to a specific discipline. Specifically, it is the specification of the discipline event as categorized by the Disciplinary Element (section 3.2.2.2 *Disciplinary Elements*). The analysis assumptions are the specification of the assumptions within the analysis itself. The analysis body comprises of the mathematical relations that makeup the method. The assumptions and analysis body are not so much classifications as actual description and embodiment of the method.

3.3 Mapping and Synthesis Generation

In the previous section, the decomposition of the complex system was described. The purpose of executing a decomposition as laid out is to have the necessary information in a capacity to identify and assimilate the necessary components to generate and execute a synthesis code. The construction of a synthesis code occurs through the mapping of the selections and the assembly of the decomposed elements based on the mapping function into a correctly composed code. This section discusses the mapping concept and code assembly concept.

3.3.1 Decomposition-Composition Mapping

The system construct discussed in this chapter, is based on the concept of decomposition and composition. For a composition to occur (the assembly of parts into a whole) the parts must exist. As such, the parts for composition are the decomposed elements (product, method, process) that must already exist within the framework. If it does not, it must be added. These points of product, method, and process storage are the hardware, method, and process library as illustrated in Figure 3-1.

A mapping of inputs is the composition of the decomposition selections made by the user into a coherent product, method, and process statement that is fundamentally the instructions for system assembly. The decomposed elements (process objective function, disciplines, methods, product, etc.) are mapped. Mapping of dependencies is the specification of the association of the parts in the global picture. The mapping concept is notionally illustrated in Figure 3-9. For every vehicle, its subsystem hardware, and the hardware's functionality, place of function, process discipline event, and the limitations and requirements placed upon it are associated. This is done for every component identified. The mapping statement is utilized by the synthesis generation routine to assemble a functional synthesis tool.

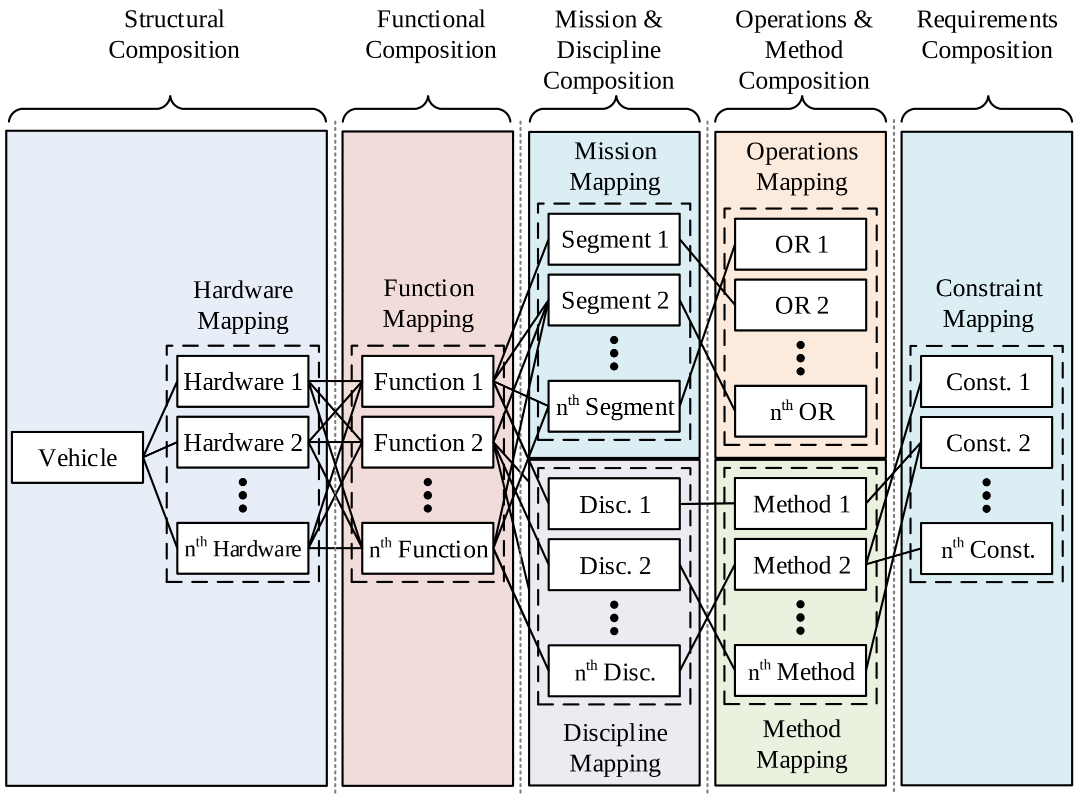


Figure 3-9 Notional mapping of decomposed elements

3.3.2 Synthesis Generation

Synthesis generation occurs through the assembly of base components into a usable form. This is illustrated in Figure 3-10. The base components include methods and processes. Each is a readymade analysis file, description, template, or data file. The correct methods and processes are selected based on the mapping of the user inputs on defining the product, methods, and process of the complex system architecture they are building. Conceptually it is a simple notion. The details of the implementation are in Chapter 4 Concept Implementation.

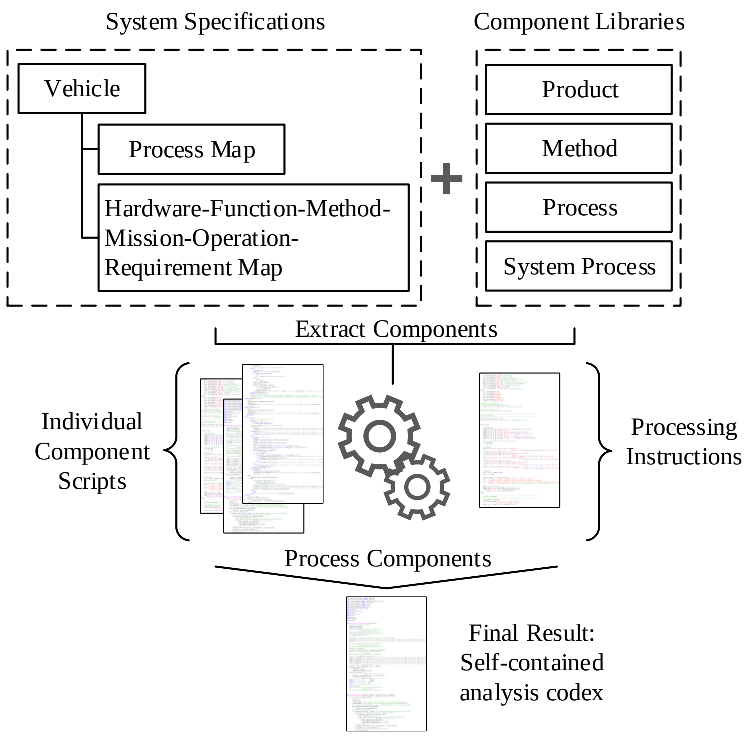


Figure 3-10 Notional synthesis generation

3.4 System Results

The system concept as defined has fundamentally three results. First, the system generates a synthesis code—that is the first result. The second result is the numerical values generated from the execution of the synthesis code generated. Lastly, the third system result, are the figures and diagrams generated and returned to the user. Each is discussed in brief below.

3.4.1 Synthesis Code

The synthesis code is the fundamental system output. All other system results depend upon this component. A system design decision is whether to assemble the synthesis code as a modular system (dispersed files) or as a self-contained system. A goal of the system being developed is transparency and ease of distribution. As such, the synthesis code is determined to be assembled as a self-contained entity. That is, all necessary decomposable elements (product, methods, and process) are contained within the synthesis file. This allows for ready distribution and control of method and information disbursement.

3.4.2 Synthesis Execution Results

The execution of the synthesis code results in analysis data. The result data is the second principal system output. The data is archived for later reference, mining, or reuse. The data is stored in a database for easy retrieval. The data is saved with every successful design iteration in the event of system or function error.

3.4.3 Return Results

The system data is processed and returned in a format for decision-making. The principal deliverable is the solution-space topography. That is, a visualization of the results for the identification of correct solutions and insights into the design problem. The solution concept allows for the automated generation of diagrams at the behest of the user. Any design variable could be visualized and assessed. A standard figure set is established. A standard set includes execution summary (convergence report) and solution topographies by standard sizing variables.

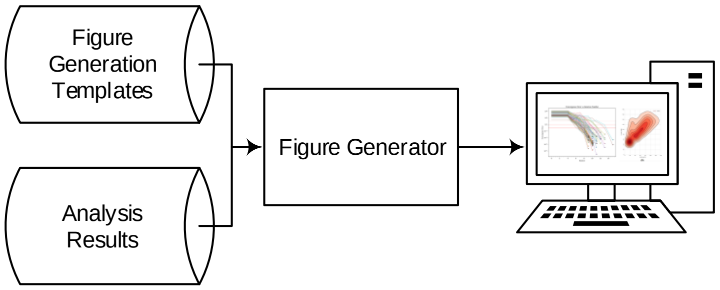


Figure 3-11 Figure generation construct

3.4.4 Recommendations

Although not implemented in the current evolution of the system generated, the solution concept accounts for the possibility for the integration of some form of a design recommender system. The recommender system would process the data, mine the results, evaluate the results, and make some design suggestions for revision or best solution set identification. However, the inclusion of a recommender system, given the other efforts of this research, is beyond the scope of a single dissertation. This element is not addressed beyond the identification of its place and usefulness.

3.5 Chapter Summary

In this chapter, the solution concept for a composable vehicle-of-vehicles synthesis assembly decision support environment was presented and discussed. The overall arching DSS concept is a semi-automated tool-of-tools. Its primary purpose is the assembly of methods into a sizing toolset to better help in decision-making. Each sizing toolset is specifically generated to solve the problem at hand.

The system is founded on the principles of system component decomposition and re-composition. Core elements—products (vehicles), processes, and methods—are described in their base components as specified by the decomposition approach. These core elements reside within depositories until needed. Through operation of a DSS, the user’s inputs are translated into a system assembly instruction function that, through a determined assembly routine, identifies, extracts, and assembles the decomposed core elements into a newly composed synthesis architecture. The synthesis architecture is executable on assembly or stored for later use or distribution. On execution, results are stored, processed, and presented to the user.

In the following chapter, this solution concept is flushed out into a functional toolset. The details of the concept’s manifestation are given.

Chapter 4 CONCEPT IMPLEMENTATION

This chapter documents the product of the concepts discussed in the previous chapter. The product is the principal deliverable of the research conducted. Recall, the product is a generic synthesis tool for rapid sizing/analysis architecture generation ready for integration into a follow on intelligent automated environment. The product is referred to as *Artificial Intelligence Design and Research Assistant Decision Support System* (AIDRA-DSS). A general system description, file system, component architecture setup, approach to system execution for problem solving, and a consideration of the systems front-end and core back-end components are the topics addressed in this chapter. Each is addressed sequentially in the following sections.

4.1 Description, Structure, and Core Components

Addressed in this section are ADIRA-DSS, its objective, and the general architecture of the system. The system's architecture includes the individual files and their organization as well as the key environments: front-end and back-end.

4.1.1 Description and Objective

AIDRA-DSS is a framework for the selecting and processing of synthesis and design analysis options for an identified vehicle of vehicles set, resulting in the generation of sizing or analysis codex that can be executed externally or internally of the framework, resulting in the presentation of standard or nonstandard decision supporting diagrams for rapid and substantiated decision making. AIDRA-DSS is a tool designed to be an environment to assist the user through accelerating design problem exploration and decision-making. The system is developed and applied for aerospace; however, the system is topic independent. That is, it is in theory not limited to anyone subject area. This system is not limited to aerospace and, as long as the designer carries the proper methodologies and processes, a vehicle can be sized or analyzed, such as a car or ship.

AIDRA-DSS has two objectives. The first objective is to explore, develop, and prepare a modular-synthesis-architecture-assembly tool for transition into a cognitive system or other AI framework. This is the driving objective of this research. In this respect, the purpose of the system is to develop further expertise and a baseline environment to test complex vehicle automated synthesis architecture synthetization that would be easily adaptable into a greater cognitive system. The second purpose is to serve as a useful engineering environment that arrives the user at a synthesized solution toolset, based on user selections, to solve a given problem by providing standard feedback and decision aiding platforms. The second objective one could consider as an intermediate objective to provide immediate system utility while driving towards the greater objective of a cognitive design and research assistant.

AIDRA-DSS is a computer software system. The general construction is illustrated in Figure 4-1. *AIDRA-DSS* has been developed in Python with GUIs written in QT. The system relies on SQL based relational database sets. The system files can be broken into two types: those that comprise the front-end and those that comprise the back-end. The front-end is the system's GUIs. The back-end is a collection of files that support the front-end in operation, such as database files and execute other tasks in the compilation of design codes. In the following sections, the systems architecture, including its files, are identified, described, and file location given.

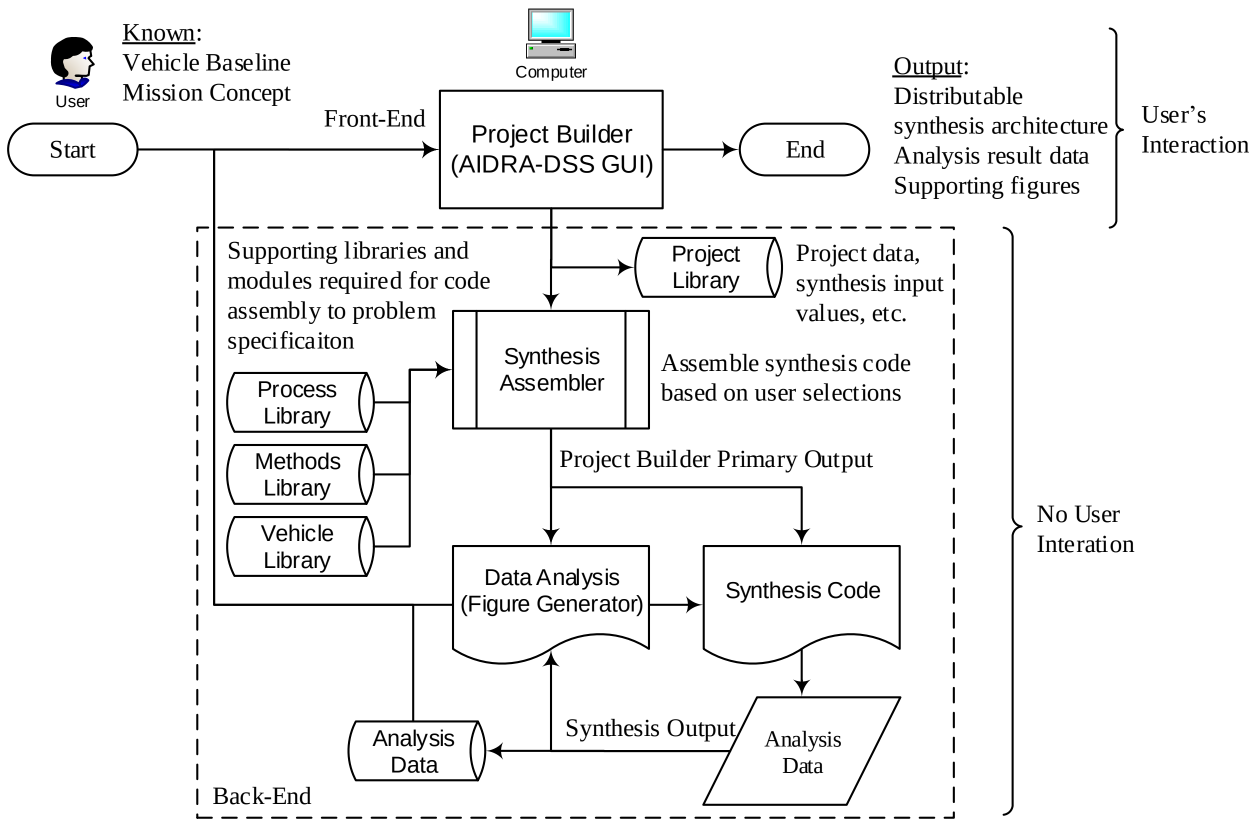

User
Known:
Vehicle Baseline
Mission Concept
Computer
Start
Front-End
Project Builder (AIDRA-DSS GUI)
End
Output:
Distributable synthesis architecture
Analysis result data
Supporting figures
User's Interaction
Supporting libraries and modules required for code assembly to problem specificaiton
Project Library
Project data, synthesis input values, etc.
Synthesis Assembler
Assemble synthesis code based on user selections
Process Library
Methods Library
Vehicle Library
Project Builder Primary Output
No User Interation
Data Analysis (Figure Generator)
Synthesis Code
Synthesis Output
Analysis Data
Analysis Data
Back-End

Figure 4-1 System architecture summary

4.1.2 Front-End

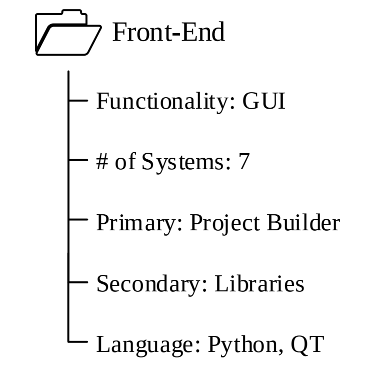


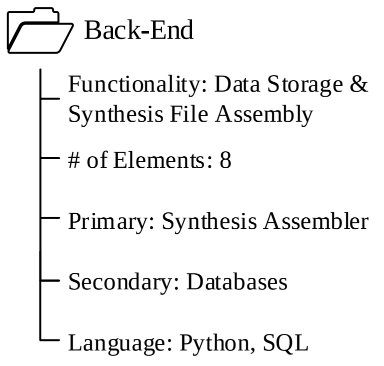
Figure 4-2 System front-end summary

The front-end is the GUI. The front-end is summarized in Figure 4-2. There are seven GUIs. Each GUI set corresponds to a different system component or module. These subsystems are *the Project Builder, Variable Library, Reference Library, Methods Library, Vehicle Library,* and *Process Library*. Each is summarized in Table 4-1. All front-end components are python based. The GUI framework is QT. Note that the python files not only contain the instructions for GUI formation but also the instructions for front-end to back-end interfacing. That is, the files contain non-GUI specific code that is required for proper GUI operation; this code is considered as back-end material. Front-end is limited to only the specific graphical interface.

Table 4-1 AIDRA-DSS's front-end systems

GUI	Description	File
Project Builder	Primary GUI file for DSS operation. Interface for DSS execution and problem solution execution	projectBuilder.py
Variable Library	Interface to handle system variables used in method development and project building	variableLibrary.py
Reference Library	Interface to a library of references that are used to support method, processes, and vehicle definition and knowledge retention	referenceLibrary.py
Methods Library	Interface to add or remove system analysis methods	methodsLibrary.py
Vehicle Library	Interface to define or remove system vehicles	vehicleLibrary.py
Process Library	Interface to create processes for analysis and synthesis	processLibrary.py

4.1.3 Back-End


Back-End
Functionality: Data Storage & Synthesis File Assembly
# of Elements: 8
Primary: Synthesis Assembler
Secondary: Databases
Language: Python, SQL
Figure 4-3 System back-end overview

The back-end files are categorize into database or operational files. The database files support the operation of the front-end. It is sumarrized in Figure 4-3. The operational files are the files that are required and contain the algorithms to properly operate the front-end (distinctly different from the GUI definition syntax), link the front-end with the back-end databases, and process the front-end option selections into a cohesive and executable sizing/analysis program. Each back-end specific file is listed in the table below; included is a brief description, indication of file type, and what front-end file it supoorts. As noted in the previouse section, the front-end python files also include the algorithms (considered as part of the back-end) that conduct the linkage between the front-end and back-end as well as the algorithms required during the operation of the GUI, such as dynamic filtering algorithms that are required to correctly populate GUI elements. All backend elements are required, however, a uniquely different and critical element is the Synthesis Assembler.

File	Description	Type	Supports
projectDatabase.db	Database to store option selections from projectBuilder.py	Database	projectBuilder.py
variableLibrary.db	Database to store option selections from variableLibrary.py	Database	variableLibrary.py
methodsLibrary.db	Database to store option selections from methodsLibrary.py	Database	methodsLibrary.py
vehicleLibrary.db	Database to store option selections from vehicleLibrary.py	Database	vehicleLibrary.py
processLibrary.db	Database to store option selections from processLibrary.py	Database	processLibrary.py
referenceLibrary.db	Database to store option selections from referenceLibrary.py	Database	referenceLibrary.py
synthesisAssembler.py	Set of algorithms to assemble the synthesis tool from the selections of *Project Builder*	Operation	projectBuilder.py

Table 4-2 AIDRA's primary back-end files

The Synthesis Assembler is the element that, as the name implies, assembles the synthesis code. A problem's elements are defined during the principal operation of the *Project Builder* (discussed in detail in later sections). Given the components of the problem, such as the vehicle selections and decomposition, the processes, and the methods selection, the Synthesis Assembler extracts the information from the systems databases and, using an auto-coding instruction algorithm, assembles the synthesis code with correct order of operation and initialization. The code is assembled automatically as per the user's specifications. The result is a unique and tailor-made synthesis code specific to the problem at hand. The Synthesis Assembler's output is a single aggregate file containing all necessary definitions and information required to execute the sizing and analysis. The Synthesis Assembler is discussed in detail in section 4.4 *Back-End: Synthesis Assembler and Architecture*

4.1.4 File Locations and Folder Tree Structure

AIDRA-DSS is built with relative file pathing. The system does not depend on a specific location for initialization and operation. All subdirectories are created through the system's operation and are generated relative to the location of the main file (projectBuilder.py). The file structure is discussed below.

AIDRA-DSS primary files' structure is illustrated in Figure 4-4. The file folder structure is relative to the main project folder. The main project folder is the folder that the user creates as the primary place for system execution and contains all necessary source files and databases. The user can indicate a specific file path for result output; however, the default structure is as illustrated.

» GUIs: Folder of system front-end GUI files.

» Databases: Folder containing databases for GUI operation.

» Processes: Folder containing process pseudo code and function text files; the folder is segmented into subfolders for each and named accordingly.

» Methods: Folder of the methods' pseudo text file and python code file; subdirectories for each category are created.

» Results: Folder containing all materials used and generated for and from the execution of a project study.

The "Results" folder itself is separated into pre and post project execution. The folder and subfolders are created on project synthesis code generation. For each project execution, a new "Project Name" folder tree is created, and the "Project Name" folder renamed according to the user specified project name. The organization scheme is depicted in Figure 4-5. The "Pre" folder contains the subset material required and used for the specific project operation. It effectively is an archival of the materials used in the project. The subset material includes the system databases, methods, and processes used for the project build as well as the synthesis file generated during system execution.

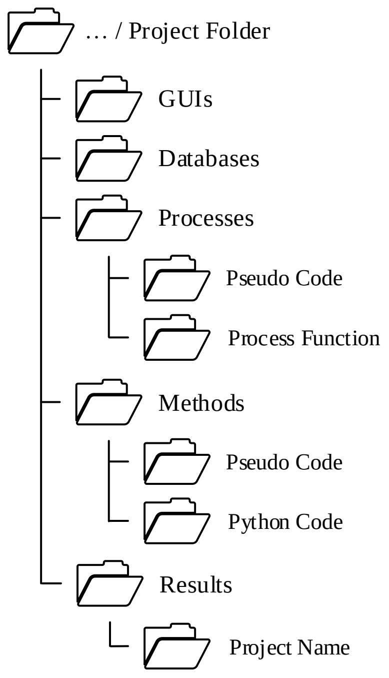


Figure 4-4 Project folder-file structure

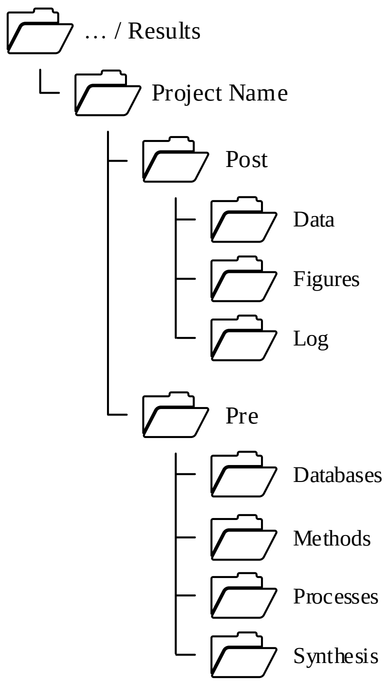


Figure 4-5 Results folder-file structure on project build

The "Post" folder contains the output of the project execution. This includes the data, figures, and system logs; the folder organization follows the naming scheme. The naming scheme relates the folder content and is self-evident to folder content; therefore, the matter will not be addressed in further detail.

4.2 Process to Problem Solving

AIDRA-DSS execution to arrive at the problem solution follows a specific sequence. This is not to be confused with the procedure to execute any specific code but rather the order of operation of the system's tools to arrive at the synthesis code and ultimately synthesis results. The overall problem-solving process is shown in Figure 4-6. The overall process is constant for all problems. Differentiation of projects occurs in the user's selections during the process execution.

The process to problem solution has four action items. The process is illustrated in Figure 4-6. For any given problem where in this system is utilized, the process steps are: define the study, create the necessary supporting elements if they are not already in system, create and execute synthesis architecture for the given problem, and, either based on initial study definition or on architecture execution results, iterate the definition itself or iterate system subcomponents, as necessary. The process repeats until satisfactory completion of the user's objective. The overall process is simple however each step is a process unto itself. Each is briefly addressed below; a detailed consideration of the components utilized within the steps is given in Section 4.3 Front-End: Core Components Description.

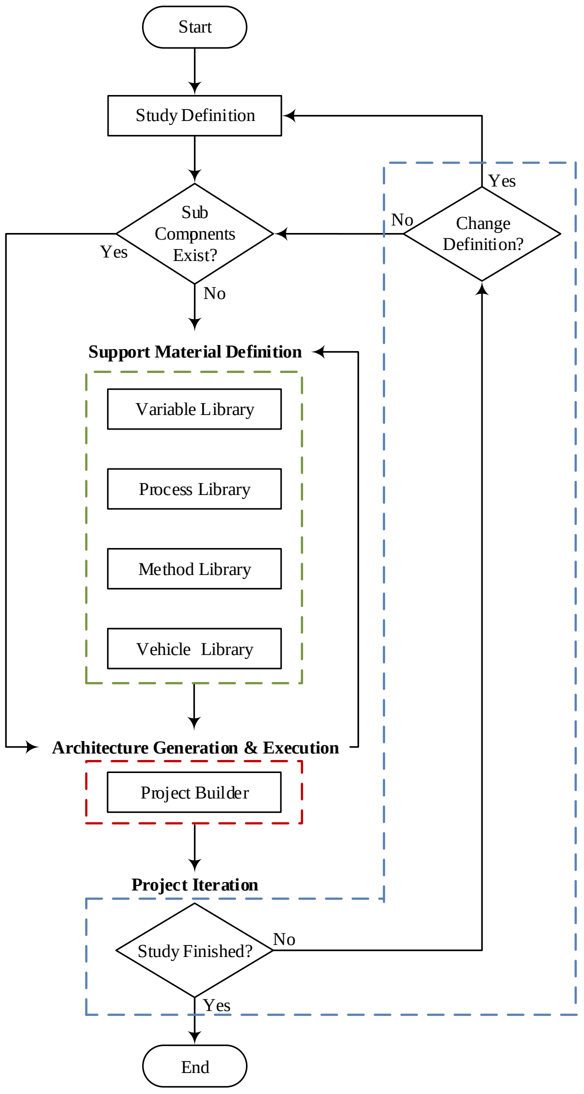


Figure 4-6 General execution process to AIDRA-DSS

4.2.1 Study Definition

In the study definitin step the user user defines the problem. Problem definition includes the identification of the overall objective of the study and the general elements of the study. Specifically, these are the elements required to create a synthesis system through the use of the *Project Builder* routine. They include, but are not limited to, the vehicle, process, mission, trajectory, trade study, and even the mehods required. By the end of this step, the user has clearly defined all system elements of a classical design study that are required to execute the *Project Builder*—generation and execution of a synthesis architecture. This is necessary as these elements should exist within the system for rapid operation; determination if they exist and, if not, to add them to the system is the subject of the next step—Support Material Definition. Note, however, many principle components can be generated from within the *Project Builder* in the event that, while operating, new requirments or study definition elements are determined necessary and yet are not currently available or in the event that the user is unaware of what components already exist in a usable form.

4.2.2 Support Material Definition

The Support Material Definition step involves the generation of the materials required for the architecture generation step. Architecture generation occurs within *Project Builder*. The supporting elements needed for architecture generation include the vehicles, processes, and methods required for the problem at hand. These correlate to the *Vehicle Library, Process Library, and Methods Library*. On identification of what is required to solve the problem in the Problem Definition step, the user must now verify that the required library elementals exist. In the event that they do not, the operator must initiate and execute the system's libraries sub-processes as necessary to add the required elements. The process is repeated as necessary per element required.

Additionally, to reiterate, any of the base elements identified during Problem Definition, can be created during the operation of *Project Builder* during the Architecture Generation and Execution step. In this way, if an element required was not foreseen, such as a particular method, it can be added to the system during *Project Builder* operation. Each supporting library is accessible through the *Project Builder*. Once the support libraries are populated to a critical level, the user could move directly to the *Project Builder* with confidence that the base elements exist and, if not, can be added, as necessary.

4.2.3 Architecture Generation and Execution

The Architecture Generation and Execution action item embodies the primary purpose of AIDRA-DSS. This action item is the execution of a process to arrive at an architecture to solve the given problem. The process to arrive at an architecture and its execution is the process of executing *Project Builder*. The *Project Builder*, similar to the supporting libraries, has its own procedure of execution. *Project Builder*'s process is discussed in detail in section 4.3 *Front-End: Core Components Description*, however, a brief discussion of it is given below.

Given the problem definition and setup, the user executes the *Project Builder* to arrive at the generation and execution of a synthesis architecture tailor made to address the problem as defined. With the vehicle and process, as required by the problem definition and subsequent creation in the support libraries, the user executes the *Project Builder*. The user selects the desired methods to model the vehicles and uses these option selections, in addition to others such as trajectory options, to assemble the core components of a synthesis architecture. The system is executed either in system with results and visualization displayed to the operator or the architecture is generated and executed later at the user's digression. With the architectures generated and executed per the problem definition, the user now considers if the problem definition has been satisfied, this takes the user to the last step in the process—Project Iteration.

4.2.4 Project Iteration

On system execution and result accumulation and review, a study can be deemed either complete or requiring iteration. A study is considered complete when the system is no longer required and the study definition is satisfied in terms of system applicability. In the event that the study is deemed complete, the system is no longer necessary and the process of system execution ends. In the event that the study is not complete with the given processes sequence, then the process repeats but with either a change in a design variable, element selection (such as the method or process), or study definition. On identification of the iteration element, the process restarts at either Problem Definition or Architecture Generation and Execution depending on if the definition or if the architecture generation element selections require iteration, respectively.

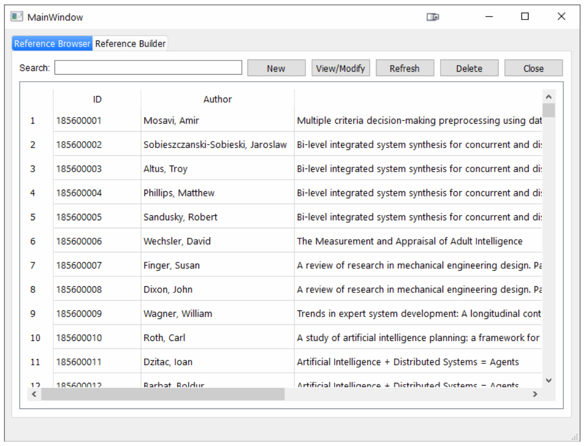
4.3 Front-End: Core Components Description

Seven principal system modules form AIDRA-DSS. The systems are: *Reference Library*, *Methods Library*, *Vehicle Library*, *Process Library*, and *Project Builder*. Each is addressed in the following sections.

4.3.1 Reference Library

The *Reference Library* is a user interface to a database of references and file correlations for both knowledge gathering and retention, and for system method, process, and vehicle referencing. The database entry listings and new entry creation form are illustrated in Figure 4-7 and Figure 4-8 respectively.

The *Reference Library* is the least complicated of the system modules. Two tabs form the front-end. The first—Figure 4-7—is the browser page; tabulated and shown is an aggregate of references according to author and title. From this page, a reference can be selected for modification or deletion, or a new reference entry procedure can be initiated. On both modify and new, the second tab—Reference Builder—is shown. Figure 4-8 shows an example for the Reference Builder tab. The tab's form has two regions. First, the general reference information is displayed. If it does not exist, as for a new entry, then the entry fields are empty and are awaiting for the information to be added. The second half of the window is a non-editable region showing where the reference has been applied, such as a project or method. Tracking the references application allows for easy accountability for reference use in method, vehicle, or process building.

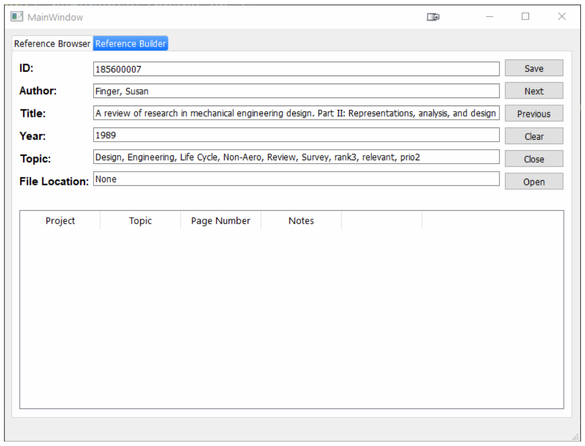

MainWindow

Reference Browser | Reference Builder

Search:

	ID	Author	
1	185600001	Mosavi, Amir	Multiple criteria decision-making preprocessing using dat
2	185600002	Sobieszczanski-Sobieski, Jaroslaw	Bi-level integrated system synthesis for concurrent and di
3	185600003	Altus, Troy	Bi-level integrated system synthesis for concurrent and di
4	185600004	Phillips, Matthew	Bi-level integrated system synthesis for concurrent and di
5	185600005	Sandusky, Robert	Bi-level integrated system synthesis for concurrent and di
6	185600006	Wechsler, David	The Measurement and Appraisal of Adult Intelligence
7	185600007	Finger, Susan	A review of research in mechanical engineering design. Pa
8	185600008	Dixon, John	A review of research in mechanical engineering design. Pa
9	185600009	Wagner, William	Trends in expert system development: A longitudinal cont
10	185600010	Roth, Carl	A study of artificial intelligence planning: a framework for
11	185600011	Dzitac, Ioan	Artificial Intelligence + Distributed Systems = Agents
12	185600012	Barbat, Boldur	Artificial Intelligence + Distributed Systems = Agents

Figure 4-7 Reference Library listing and start page


MainWindow

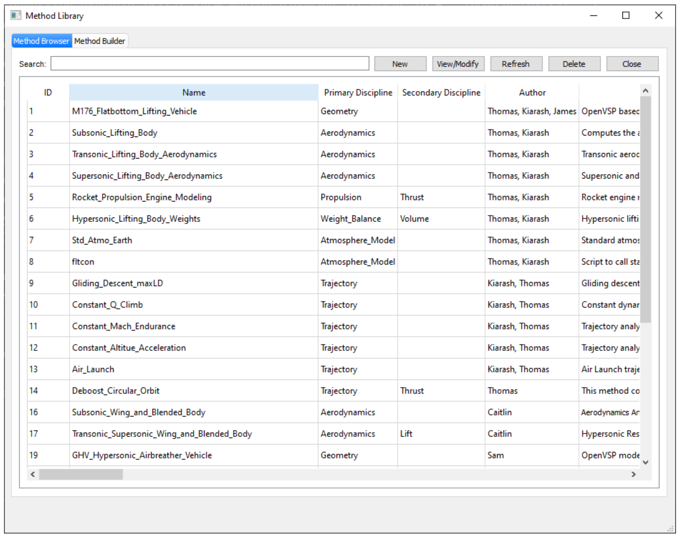
Reference Browser | Reference Builder

ID:	185600007
Author:	Finger, Susan
Title:	A review of research in mechanical engineering design. Part II: Representations, analysis, and design
Year:	1989
Topic:	Design, Engineering, Life Cycle, Non-Aero, Review, Survey, rank3, relevant, prio2
File Location:	None

Project	Topic	Page Number	Notes

Figure 4-8 Reference input and documentation form

4.3.2 Methods Library


Method Library

Method Browser | Method Builder

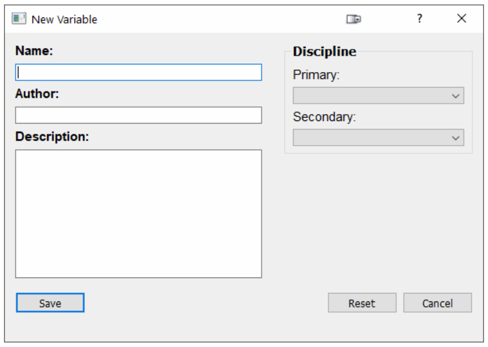
Search:

ID	Name	Primary Discipline	Secondary Discipline	Author	
1	M176_Flatbottom_Lifting_Vehicle	Geometry		Thomas, Kiarash, James	OpenVSP basec
2	Subsonic_Lifting_Body	Aerodynamics		Thomas, Kiarash	Computes the a
3	Transonic_Lifting_Body_Aerodynamics	Aerodynamics		Thomas, Kiarash	Transonic aeroc
4	Supersonic_Lifting_Body_Aerodynamics	Aerodynamics		Thomas, Kiarash	Supersonic and
5	Rocket_Propulsion_Engine_Modeling	Propulsion	Thrust	Thomas, Kiarash	Rocket engine r
6	Hypersonic_Lifting_Body_Weights	Weight_Balance	Volume	Thomas, Kiarash	Hypersonic lifti
7	Std_Atmo_Earth	Atmosphere_Model		Thomas, Kiarash	Standard atmos
8	fltcon	Atmosphere_Model		Thomas, Kiarash	Script to call sta
9	Gliding_Descent_maxLD	Trajectory		Kiarash, Thomas	Gliding descent
10	Constant_Q_Climb	Trajectory		Kiarash, Thomas	Constant dynar
11	Constant_Mach_Endurance	Trajectory		Kiarash, Thomas	Trajectory analy
12	Constant_Altitue_Acceleration	Trajectory		Kiarash, Thomas	Trajectory analy
13	Air_Launch	Trajectory		Kiarash, Thomas	Air Launch traje
14	Deboost_Circular_Orbit	Trajectory	Thrust	Thomas	This method co
16	Subsonic_Wing_and_Blended_Body	Aerodynamics		Caitlin	Aerodynamics Ar
17	Transonic_Supersonic_Wing_and_Blended_Body	Aerodynamics	Lift	Caitlin	Hypersonic Res
19	GHV_Hypersonic_Airbreather_Vehicle	Geometry		Sam	OpenVSP mode

Figure 4-9 Methods Library Browser

The *Methods Library* is a database of system analysis methods accessible to the *Project Builder* for application in the synthesis tool. The database entry listings and new entry creation form are illustrated in Figure 4-9 and Figure 4-10 respectively. The Method Browser tab is the *Methods Library's* home screen. From here, a method can be deleted, modified, or added. A listing of all currently entered methods is given. The methods are listed alphabetically. Method ID numbers are unique and automatically generated on method creation. General method information is displayed including the discipline the method is associated to and a brief method description.

To create a new method, the user clicks the "New" button on the Method Browser tab. To view an existing method, the user clicks a cell corresponding to the desired method's row and selects "Modify".


New Variable

Name:

Author:

Description:

Discipline

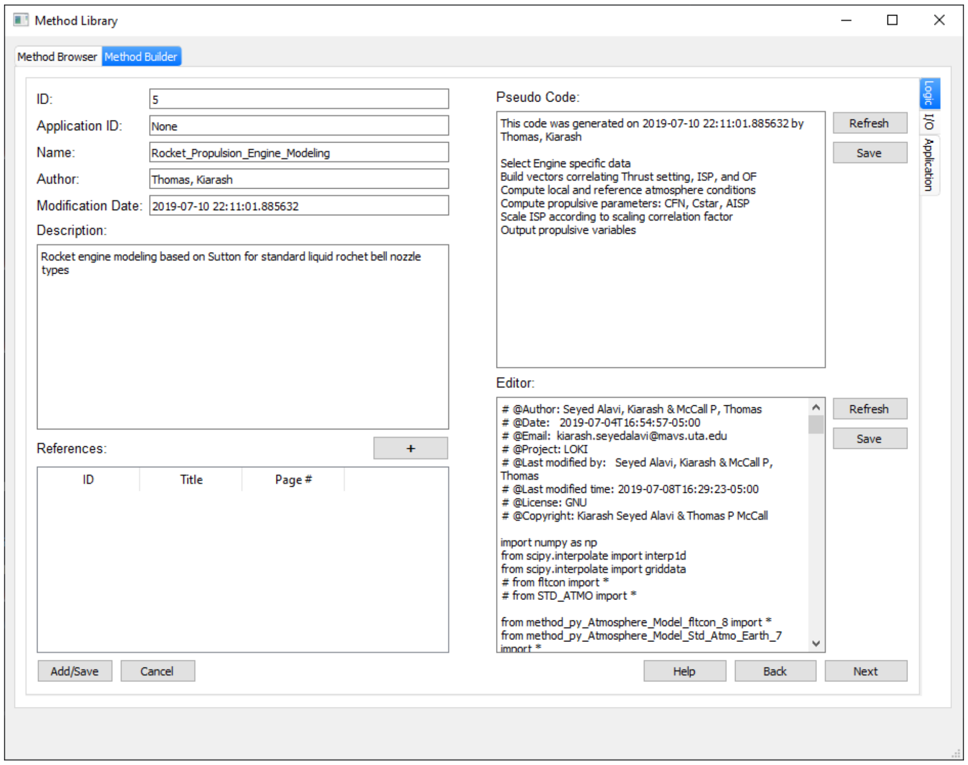
Primary:

Secondary:

Figure 4-10 New Method form

On selection of "New", the user is presented with a method entry form as shown in Figure 4-10. As evident in the figure, the user enters the method's general information such as a descriptive name and a general description. Additionally, the user selects the applicable primary and secondary disciplines, such as aerodynamics or propulsion, thereby correlating the method to a discipline. The selection of a primary discipline is required. On save, the new method information is saved to the back-end database and appropriate tables are created as well as blank method script files that are discussed later. On completion, the user is taken to the Method Builder tab—Figure 4-11—for further method specification. On modify, the user is directly shown the Method Builder tab.

The Method Builder is where the method is fully described, entered, and associated. This task is segmented into three subtabs in the Method Builder page labeled Logic, I/O, and Application; they are shown in Figure 4-11, Figure 4-12, and Figure 4-13 respectively. The Logic tab is organized into four fields. The left half of the page contains two fields for describing and documenting the method. This includes the general information entered on the method creation form and a field for selecting the references for the method. The user is to add the references to support the method. The documenting of the references allows for an easy identification of source material for later reference if necessary.



ID:	5
Application ID:	None
Name:	Rocket_Propulsion_Engine_Modeling
Author:	Thomas, Kiarash
Modification Date:	2019-07-10 22:11:01.885632

Description:

Rocket engine modeling based on Sutton for standard liquid rochet bell nozzle types

References:

ID	Title	Page #

Pseudo Code:

```
This code was generated on 2019-07-10 22:11:01.885632 by
Thomas, Kiarash

Select Engine specific data
Build vectors correlating Thrust setting, ISP, and OF
Compute local and reference atmosphere conditions
Compute propulsive parameters: CFN, Cstar, AISP
Scale ISP according to scaling correlation factor
Output propulsive variables
```

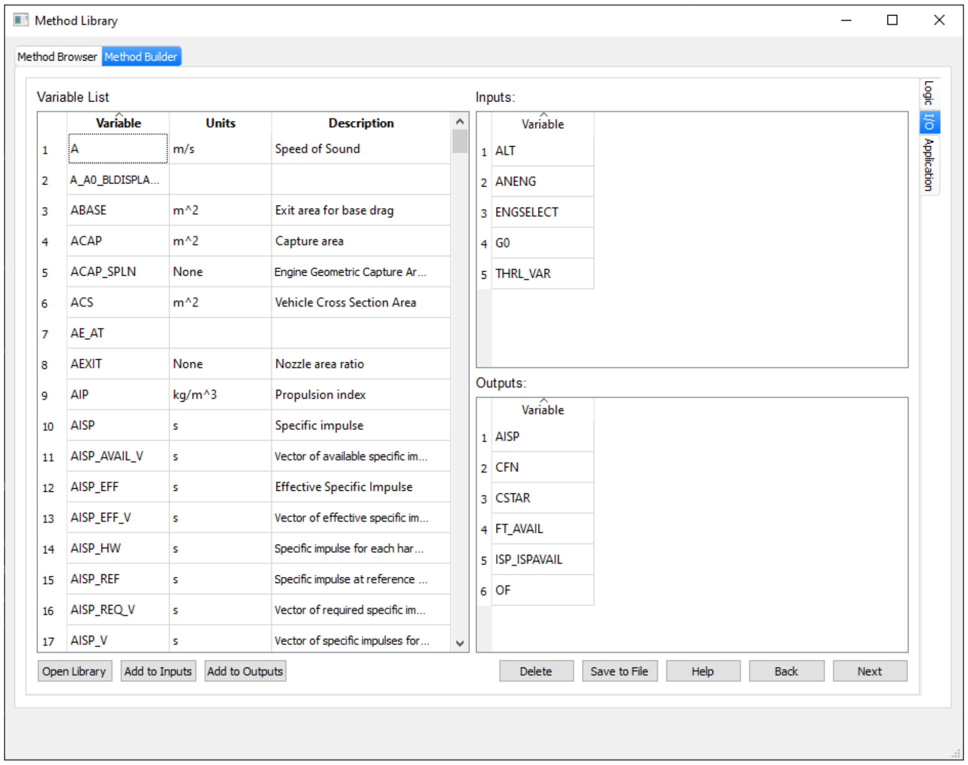
Editor:

```
# @Author: Seyed Alavi, Kiarash & McCall P, Thomas
# @Date:   2019-07-04T16:54:57-05:00
# @Email:  kiarash.seyedalavi@mavs.uta.edu
# @Project: LOKI
# @Last modified by:   Seyed Alavi, Kiarash & McCall P,
Thomas
# @Last modified time: 2019-07-08T16:29:23-05:00
# @License: GNU
# @Copyright: Kiarash Seyed Alavi & Thomas P McCall

import numpy as np
from scipy.interpolate import interp1d
from scipy.interpolate import griddata
# from fltcon import *
# from STD_ATMO import *

from method_py_Atmosphere_Model_fltcon_8 import *
from method_py_Atmosphere_Model_Std_Atmo_Earth_7
import *
```

Figure 4-11 Method Builder—Logic definition tab

The right half of the Logic tab's page consists of two text fields for the method's code documentation: "Pseudo Code" and "Editor". The "Pseudo Code" section is a text field to document—via pseudo code—the method script that is entered in the "Editor" text field below "Pseudo Code" field, in the bottom right corner. The "Pseudo Code" is saved to a text file in the back-end that is created on method creation. The "Editor" field is a text field as well. Within this field, the user enters the method. The user must write in the "Editor" field in proper python syntax (or paste a properly written script into the field). Similar to the "Pseudo Code" field and associated file, the “Editor” associated file and entries are similarly created and saved to the back-end, but as python script files. All method pseudo code and script files s are stored relative to the master folder as described in the previous section. Both files are named according to the method name, discipline, and ID generated on method creation. On entering the information and selecting next, all page information is saved to the back-end and the user is displayed the I/O tab—Figure 4-12.

The I/O page is the page in which the user identifies the method’s inputs and outputs. As is visible, the page is separated into two regions. The left half is a list of all variables currently stored in the system and is loaded from the *Variable Library’s* database. The right half of the page contains tables for displaying the selected input and output variables. To add a variable to the selection fields, the user must select the desired variable from the variable list and add them to either the inputs or the outputs list by clicking the appropriate button: “Add to Inputs” or “Add to Outputs”. On click, the currently selected variables in the variable list are added to the appropriate list according to the button activated.



Variable List

	Variable	Units	Description
1	A	m/s	Speed of Sound
2	A_A0_BLDISPLA...		
3	ABASE	m^2	Exit area for base drag
4	ACAP	m^2	Capture area
5	ACAP_SPLN	None	Engine Geometric Capture Ar...
6	ACS	m^2	Vehicle Cross Section Area
7	AE_AT		
8	AEXIT	None	Nozzle area ratio
9	AIP	kg/m^3	Propulsion index
10	AISP	s	Specific impulse
11	AISP_AVAIL_V	s	Vector of available specific im...
12	AISP_EFF	s	Effective Specific Impulse
13	AISP_EFF_V	s	Vector of effective specific im...
14	AISP_HW	s	Specific impulse for each har...
15	AISP_REF	s	Specific impulse at reference ...
16	AISP_REQ_V	s	Vector of required specific im...
17	AISP_V	s	Vector of specific impulses for...

Inputs:

	Variable
1	ALT
2	ANENG
3	ENGSELECT
4	G0
5	THRL_VAR

Outputs:

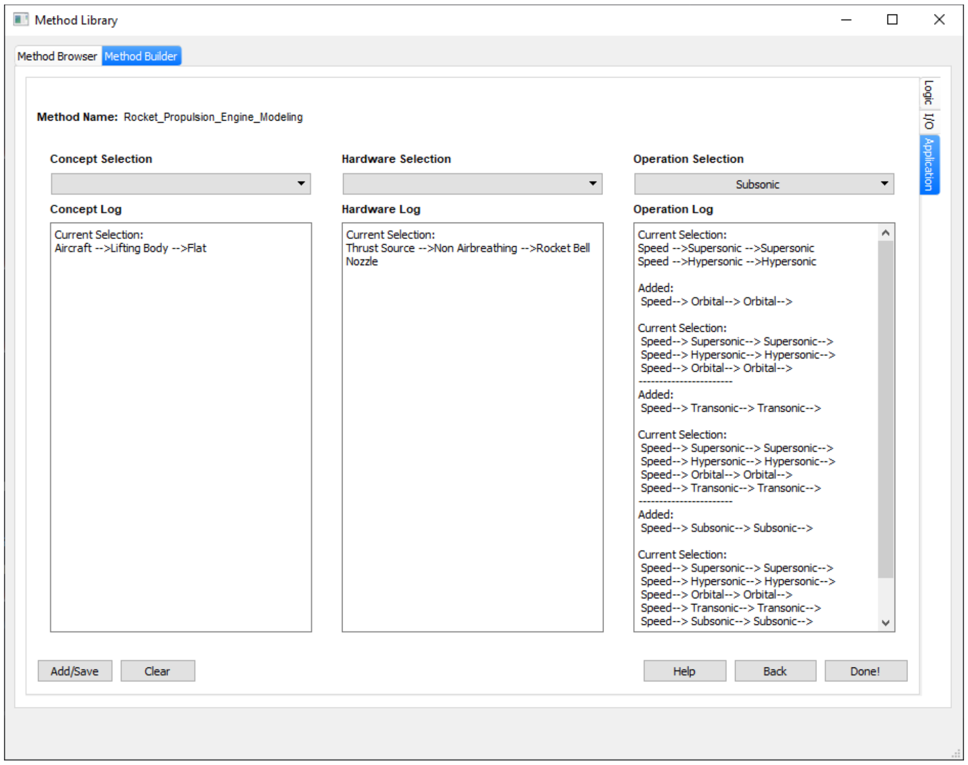
	Variable
1	AISP
2	CFN
3	CSTAR
4	FT_AVAIL
5	ISP_ISPAVAIL
6	OF

Figure 4-12 Method Builder–Input and Outputs (I/O) tab

If the desired variable does not exist, the user can click the “Open Library” button, which will open the *Variable Library*. Upon opening the *Variable Library*, the user shall add the desired variable, after which, it will be available and displayed in the I/O variable list.

Upon completion of specifying the input and output variables, the user clicks “Next”. On click of “Next”, two operations occur. First, the selected variables are saved to the back-end in the appropriate tables according to whether they are inputs or outputs. Second, the active tab changes to Application, the last tab.

The *Method Library's* Application tab—Figure 4-13—is where the user identifies the method's applicable conditions. The applicable conditions are those states in which the method is applicable and, therefore, are requirement conditions that should be met for correct method application. By selecting the appropriate condition field elements, the user will correctly define the applicability of the method. These options are selected from the drop-down menus above each text field. The text field displays the current selections and notifies the user on condition add or remove.



Method Name: Rocket_Propulsion_Engine_Modeling

Concept Selection

Concept Log

```
Current Selection:
Aircraft -->Lifting Body -->Flat
```

Hardware Selection

Hardware Log

```
Current Selection:
Thrust Source -->Non Airbreathing -->Rocket Bell
Nozzle
```

Operation Selection

Subsonic

Operation Log

```
Current Selection:
Speed -->Supersonic -->Supersonic
Speed -->Hypersonic -->Hypersonic

Added:
 Speed--> Orbital--> Orbital-->

Current Selection:
 Speed--> Supersonic--> Supersonic-->
 Speed--> Hypersonic--> Hypersonic-->
 Speed--> Orbital--> Orbital-->
----------------------
Added:
 Speed--> Transonic--> Transonic-->

Current Selection:
 Speed--> Supersonic--> Supersonic-->
 Speed--> Hypersonic--> Hypersonic-->
 Speed--> Orbital--> Orbital-->
 Speed--> Transonic--> Transonic-->
----------------------
Added:
 Speed--> Subsonic--> Subsonic-->

Current Selection:
 Speed--> Supersonic--> Supersonic-->
 Speed--> Hypersonic--> Hypersonic-->
 Speed--> Orbital--> Orbital-->
 Speed--> Transonic--> Transonic-->
 Speed--> Subsonic--> Subsonic-->
```

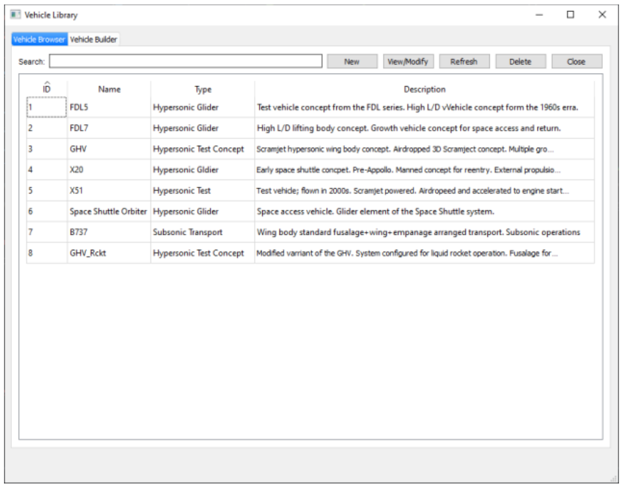
Figure 4-13 Method Builder—Application tab

There are three condition fields. They are Concept, Hardware, and Operation. The Concept field identifies the type of vehicle concept the method is applicable to, such as a TSTO launch vehicle or a flat-bottom lifting vehicle. Similarly, the Hardware field is the collection of decomposed system hardware components that assemble into a system-of-systems vehicle. Hardware includes such elements as the landing gear and type, the engine and type, lifting surfaces, etc. The Hardware are the physical components that assemble into the total system that the method models. The Operation field identifies the operating conditions in which the method is applicable. This would include, for example, man vs unmanned, subsonic vs supersonic, fuel and oxidizer type, etc.

After the completion of the application condition selections, the user clicks "Done." On doing so, the information selected is saved to the back-end database, the method addition or modification is complete, and the *Method Library's* Browser tab is displayed, where in, the process can be repeated for a new method or method modification, as necessary.

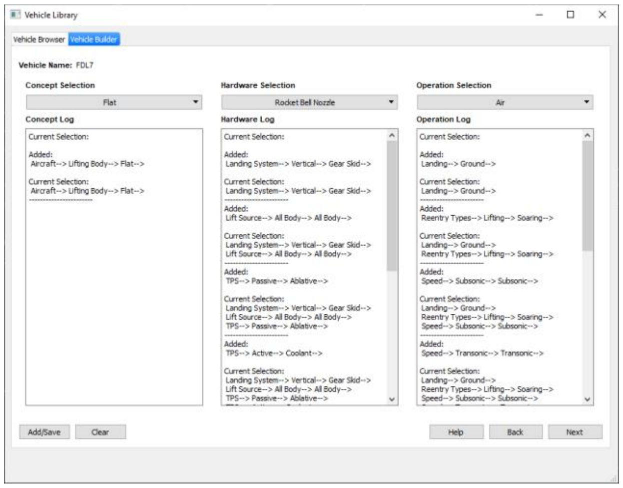
4.3.3 Vehicle Library

The *Vehicle Library* constitutes an interface and database for the creation and storage of vehicles that are to be employed within the *Project Builder*. The *Vehicle Library* interface is the means in which the user specifies the constituents composing the vehicle. Two tabs formulate the *Vehicle Library*. They are Vehicle Browser and Vehicle Builder, each depicted in Figure 4-14 and Figure 4-15 respectively.



ID	Name	Type	Description
1	FDL5	Hypersonic Glider	Test vehicle concept from the FDL series. High L/D vVehicle concept form the 1960s erra.
2	FDL7	Hypersonic Glider	High L/D lifting body concept. Growth vehicle concept for space access and return.
3	GHV	Hypersonic Test Concept	Scramjet hypersonic wing body concept. Airdropped 3D Scramject concept. Multiple gro...
4	X20	Hypersonic Gldier	Early space shuttle concpet. Pre-Appollo. Manned concept for reentry. External propulsio...
5	X51	Hypersonic Test	Test vehicle; flown in 2000s. Scramjet powered. Airdropeed and accelerated to engine start...
6	Space Shuttle Orbiter	Hypersonic Glider	Space access vehicle. Glider element of the Space Shuttle system.
7	B737	Subsonic Transport	Wing body standard fusalage+wing+empanage arranged transport. Subsonic operations
8	GHV_Rckt	Hypersonic Test Concept	Modified varriant of the GHV. System configured for liquid rocket operation. Fusalage for...

Figure 4-14 Vehicle Library—Vehicle Browser



Vehicle Name: FDL7

Concept Selection: Flat
Hardware Selection: Rocket Bell Nozzle
Operation Selection: Air

Concept Log:
```
Current Selection:

Added:
 Aircraft--> Lifting Body--> Flat-->

Current Selection:
 Aircraft--> Lifting Body--> Flat-->
-----------------------
```

Hardware Log:
```
Current Selection:

Added:
 Landing System--> Vertical--> Gear Skid-->

Current Selection:
 Landing System--> Vertical--> Gear Skid-->
-----------------------
Added:
 Lift Source--> All Body--> All Body-->

Current Selection:
 Landing System--> Vertical--> Gear Skid-->
 Lift Source--> All Body--> All Body-->
-----------------------
Added:
 TPS--> Passive--> Ablative-->

Current Selection:
 Landing System--> Vertical--> Gear Skid-->
 Lift Source--> All Body--> All Body-->
 TPS--> Passive--> Ablative-->
-----------------------
Added:
 TPS--> Active--> Coolant-->

Current Selection:
 Landing System--> Vertical--> Gear Skid-->
 Lift Source--> All Body--> All Body-->
 TPS--> Passive--> Ablative-->
```

Operation Log:
```
Current Selection:

Added:
 Landing--> Ground-->

Current Selection:
 Landing--> Ground-->
-----------------------
Added:
 Reentry Types--> Lifting--> Soaring-->

Current Selection:
 Landing--> Ground-->
 Reentry Types--> Lifting--> Soaring-->
-----------------------
Added:
 Speed--> Subsonic--> Subsonic-->

Current Selection:
 Landing--> Ground-->
 Reentry Types--> Lifting--> Soaring-->
 Speed--> Subsonic--> Subsonic-->
-----------------------
Added:
 Speed--> Transonic--> Transonic-->

Current Selection:
 Landing--> Ground-->
 Reentry Types--> Lifting--> Soaring-->
 Speed--> Subsonic--> Subsonic-->
```

Figure 4-15 Vehicle Library—Vehicle Builder

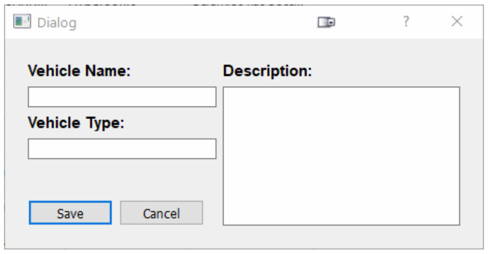


Figure 4-16 Vehicle Library—New Vehicle

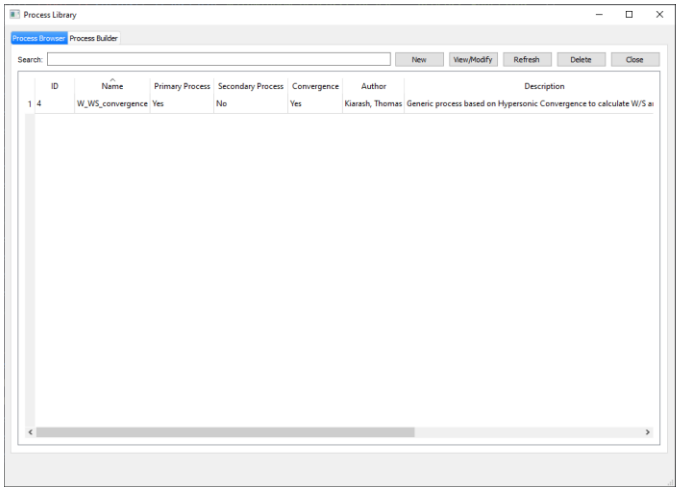
In opening the *Vehicle Library*, the user is presented with the Vehicle Browser. This browser's operation and layout is the same as all previously discussed browser pages. The user is presented a chronicle of all currently entered vehicles. From this page, the user can select to modify, create, or delete a vehicle. To modify a vehicle the user selects a cell in the desired vehicle's row and selects "View/Modify". At this point, the user will be displayed the "Vehicle Builder" window (discussed below). On the click of "New", the user is presented a vehicle initialization window as shown in Figure 4-16. As evident in the figure, the user enters a vehicle name, the vehicle type, and a vehicle description. This information displays in the Vehicle Browser's table of vehicles on continuation as a new vehicle entry. On completion and save, the Vehicle Builder tab displays.

The Vehicle Builder tab presents the interface in which the user specifies the specifics of the vehicle and, in doing, defines the vehicle. The Vehicle Builder page is displayed in Figure 4-15; the layout and operation are the same as that in the *Methods Library's* Application subtab. The user specifies the constituents composing, defining, and limiting the vehicle. This is accomplished by specifying the vehicle's Concept, Hardware, and Operation. The constituents are subsystem components or conditions that specify the total system. These constituents are the same elements within the condition fields of the *Methods Library*. If a vehicle has more than one of an element, the user need only select it once. On click of "Next", the information selected is saved and the user is returned to the Vehicle Browser tab; this completes a vehicle build.

4.3.4 Process Library

The *Process Library* is the interface where in an analytical process is defined and stored. A process can be either a secondary or a primary process. A secondary process is a process that occurs within or in the context of a primary process. A primary process is a process that can be executed independently or in conjunction with another defined system process (secondary process). A primary process governs the total system; a secondary process must exist within a primary process. The creation of each within the *Process Library*, follows the same procedure.

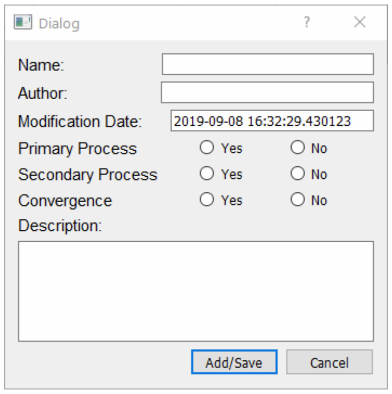
The *Process Library*'s approach is consistent with the other libraries of AIDRA-DSS. As in the other libraries, the *Process Library* comprises of two principal tabs: "Process Browser" and "Process Builder". The "Process Browser"—Figure 4-17—is the screen shown at library initialization. A table presents the user with all currently recorded processes. The user is presented with each process' identifying information. This includes the process' name, whether it is a primary or secondary process, whether or not convergence occurs within the process, and a process description. The identifying process information is entered on process creation, see Figure 4-18. From the browser page, the user can select a process for modification or deletion, or the user can initialize the creation of a new process. On the selection of a process and clicking "View/Modify", the user is taken to the Process Builder tab, Figure 4-19. On clicking "New" the new process entry form is presented, Figure 4-18.



	ID	Name	Primary Process	Secondary Process	Convergence	Author	Description
1	4	W_WS_convergence	Yes	No	Yes	Kiarash, Thomas	Generic process based on Hypersonic Convergence to calculate W/S a

Figure 4-17 Process Library—Process Browser

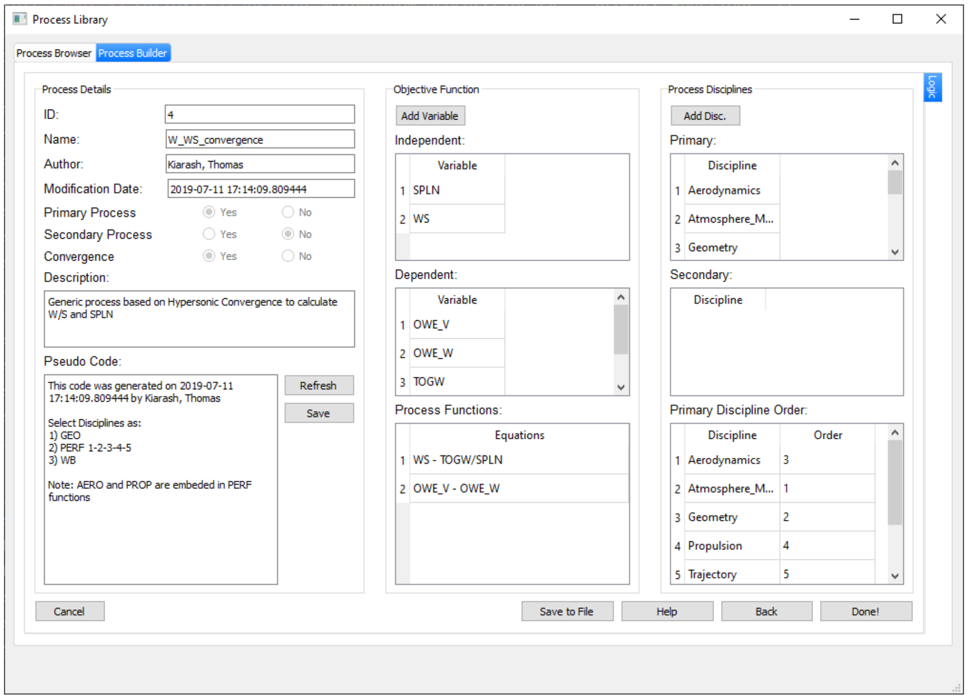
The new process initialization form is shown in Figure 4-18. On completion of this form, the new process is initialized within the system. To begin, the user enters the process identifying information: name, author, process type (primary or secondary), convergence class (yes or no), and a brief description. On the click of "Add/Save" the necessary tables for the new process are created in the back-end and the information is saved, thereby initializing the process. Additionally, this form closes and the user is displayed an active Process Builder tab.



Name:		
Author:		
Modification Date:	2019-09-08 16:32:29.430123	
Primary Process	Yes	No
Secondary Process	Yes	No
Convergence	Yes	No
Description:		

Figure 4-18 Process Library—New Process form

The Process Builder, Figure 4-19, is where the user defines or edits a process. The tab's page is setup in three columns. The first column constitutes the general information describing the process. This includes the sections: Process Details and Pseudo Code. The Process Details section is the process information added in the new process form; it is not editable. The Pseudo Code section has a text field for the user to specify in natural language the process. The text added here is saved in a text file in the back-end.



Process Details

ID:	4
Name:	W_WS_convergence
Author:	Kiarash, Thomas
Modification Date:	2019-07-11 17:14:09.809444
Primary Process	[✓ Yes] No
Secondary Process	Yes [✓ No]
Convergence	[✓ Yes] No

Description:

Generic process based on Hypersonic Convergence to calculate W/S and SPLN

Pseudo Code:

```
This code was generated on 2019-07-11
17:14:09.809444 by Kiarash, Thomas

Select Disciplines as:
1) GEO
2) PERF 1-2-3-4-5
3) WB

Note: AERO and PROP are embeded in PERF
functions
```

Refresh | Save

Objective Function

Add Variable

Independent:

	Variable
1	SPLN
2	WS

Dependent:

	Variable
1	OWE_V
2	OWE_W
3	TOGW

Process Functions:

	Equations
1	WS - TOGW/SPLN
2	OWE_V - OWE_W

Process Disciplines

Add Disc.

Primary:

	Discipline
1	Aerodynamics
2	Atmosphere_M...
3	Geometry

Secondary:

Discipline

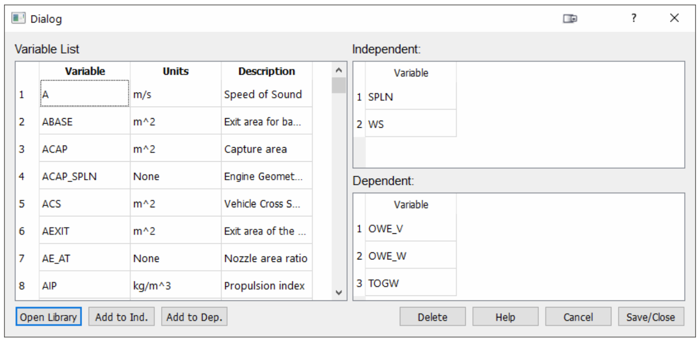
Primary Discipline Order:

	Discipline	Order
1	Aerodynamics	3
2	Atmosphere_M...	1
3	Geometry	2
4	Propulsion	4
5	Trajectory	5

Cancel | Save to File | Help | Back | Done!

Figure 4-19 Process Library—Process Builder

The second column contains the section Objective Function. If the process has a governing objective equation(s), in this section the user specifies it. The process of specifying the objective function has two parts. First, the user must add the independent and dependent function variables. They are added by clicking "Add Variable"; at which a form to select the variables opens, see Figure 4-20. The form has on the left a table of available variables to choose. On the right are two tables, one for independent variable listing and a second for dependent variable listing. To add to either table, and thereby make a variable an independent or dependent process variable, the user selects the desired variable from the variable list and then selects either "Add to Ind." or "Add to Dep." to add the variable to the independent or dependent variable list, respectively. When all necessary variables are added, the user clicks "Done", at which point the form is closed, the selections saved to the back-end, and the Process Builder tab's page is shown with the selected variables visible in the appropriate Primary or Secondary tables.



Variable List

	Variable	Units	Description
1	A	m/s	Speed of Sound
2	ABASE	m^2	Exit area for ba...
3	ACAP	m^2	Capture area
4	ACAP_SPLN	None	Engine Geomet...
5	ACS	m^2	Vehicle Cross S...
6	AEXIT	m^2	Exit area of the ...
7	AE_AT	None	Nozzle area ratio
8	AIP	kg/m^3	Propulsion index

Independent:

	Variable
1	SPLN
2	WS

Dependent:

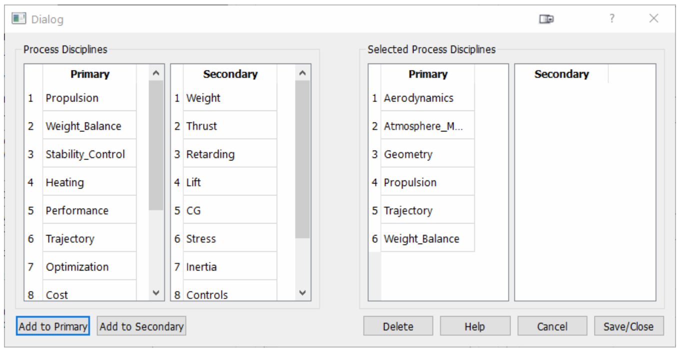
	Variable
1	OWE_V
2	OWE_W
3	TOGW

Open Library | Add to Ind. | Add to Dep. | Delete | Help | Cancel | Save/Close

Figure 4-20 Process Builder—Objective Function variable selection form

The second part of the objective function specification process is to enter the objective function itself. The objective function(s) are added in the Process Function table. On the selection of independent variables, the Process Function table row count is set to the number of independent variables. The user must type the objective functions into the newly created rows. Between all objective functions entered, all independent and dependent variables must be used at least once. The variable names must be entered as they appear in the Independent and Dependent tables. It is upon the user to verify accuracy in entry and that all variables have been used. Additionally, all objective functions should equate to zero. The system is current set only to solve for objective functions in this form.

The form's third column section is Process Disciplines. In this region, the user identifies the primary and secondary disciplines, and the primary discipline order of operation. The user first selects the disciplines and then specifies the order. To select the disciplines, the user clicks "Add Disc". On clicking, the discipline selection form, Figure 4-21, is presented.



Process Disciplines

	Primary		Secondary
1	Propulsion	1	Weight
2	Weight_Balance	2	Thrust
3	Stability_Control	3	Retarding
4	Heating	4	Lift
5	Performance	5	CG
6	Trajectory	6	Stress
7	Optimization	7	Inertia
8	Cost	8	Controls

Selected Process Disciplines

	Primary	Secondary
1	Aerodynamics	
2	Atmosphere_M...	
3	Geometry	
4	Propulsion	
5	Trajectory	
6	Weight_Balance	

Figure 4-21 Process Builder—Process Discipline Selection form

The form layout and the process of selecting disciplines, is similar to the variable selection process. The form is organized with the disciplines (primary and secondary) available on the left and those selected on the right. To add a discipline to the Selected Process Discipline tables, the user selects the variable from either the Primary or the Secondary tables under the Process Disciplines area and selects appropriately either "Add to Primary" or "Add to Secondary". The user cannot mix disciplines; that is, the user cannot add a primary discipline as a secondary discipline or a secondary discipline as a primary discipline. To add or modify the list of available disciplines, the user must adjust the appropriate table in the back-end database. On selecting the desired disciplines, the user clicks "Save/Close", at which point the user is presented with the Process Builder window, where in the previously selected disciplines are shown in the appropriate Primary and Secondary tables. To finalize the process build, the user need only award the discipline order. The discipline execution is serial. In the third and right most column, under Primary Discipline Order, the user is displayed a table populated with the primary disciplines. The user must add a numerical value correlating to the disciplines order in the overall process. The numbering must be integer based; the order will be executed in numerically increasing order. That is, the discipline with the lowest numeric value will occur first in the process execution; the value with the greatest numeric value will occurs last.

On completion of all form sections, the user completes the process creation and build by clicking "Done", at which point all entered information is appropriately written to the tables in the back-end and the user is returned to the Process Browser.

4.3.5 Project Builder

The *Project Builder* is the primary component of AIDRA-DSS. This system is the DSS environment. The *Project Builder* has seven discernable secondary components in addition to the standard library browser. These components are the tabs of the DSS GUI. The tab breakdown is illustrated in Figure 4-22, the order of operation is indicated by the numerals. The inspiration for segmenting the system into the specific scheme what is called "the standard to design ladder" as presented in [88]. Simply, the ladder is a symbolic representation of specific technical tasks that should be present in a technically rigorous process of design and evaluation. Each tab has a unique task to build towards the final deliverables and ultimately decision making. A summary of each tab's objectives is given below. This followed by an in-depth discussion of each tab as fabricated.

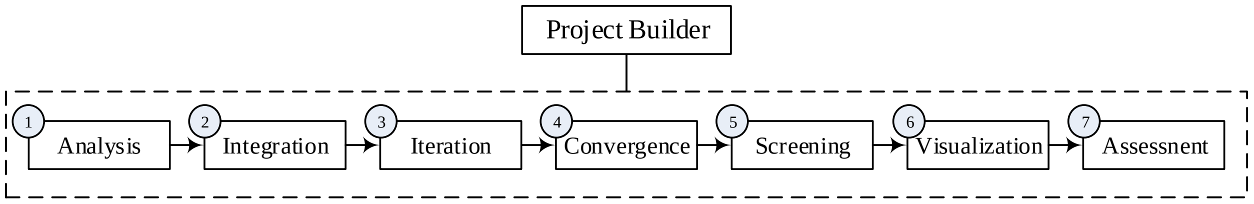


Figure 4-22 AIDRA-DSS Project Builder tab set

» Analysis: Selection of primary and secondary vehicles and corresponding trajectory segments

» Integration: Selection of architecture processes and assignment to the vehicle selections, selection of hardware-discipline methods, and association of hardware function to mission segment

» Iteration: Selection and specification of a method set for hardware requiring multiple methods per discipline, and establishment of vehicle trajectory

» Convergence: Review of architecture processes selections including convergence specifications, and specification of additional convergence criteria including the option for user specific solver selection and solver option specification

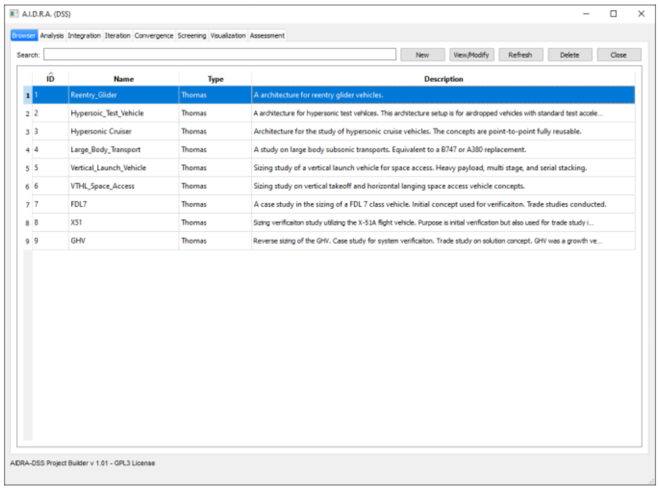
» Screening: Specification of the study as single or multipoint, specification of trade variables and ranges if a trade study, and specification of required input variable values

» Visualization: Selection of visualization materials to be generated to assist in design evaluation

» Assessment: Evaluation of study results for system accuracy and general design insights towards the design problem at hand

4.3.5.1 Project Builder Browser

As in the case of the libraries, the *Project Builder* also starts with a browser window. The style and operation are consistent with the other browsers. A table lists all currently started project builds. From this vantage point, the user can delete, modify, or begin a project. The operation is identical to the browser in the other libraries discussed earlier. The Project Builders browser tab and new project window are shown in Figure 4-23 and Figure 4-24respectively.

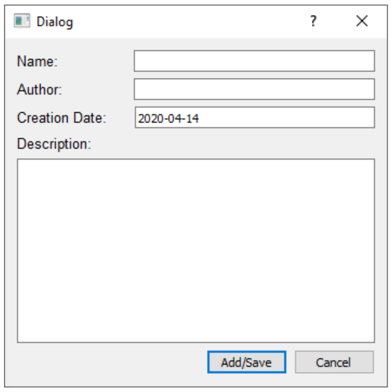


Search:

	ID	Name	Type	Description
1	1	Reentry_Glider	Thomas	A architecture for reentry glider vehicles.
2	2	Hypersoic_Test_Vehicle	Thomas	A architecture for hypersonic test vehilces. This architecture setup is for airdropped vehicles with standard test accele...
3	3	Hypersonic Cruiser	Thomas	Architecture for the study of hypersonic cruise vehicles. The concepts are point-to-point fully reusable.
4	4	Large_Body_Transport	Thomas	A study on large body subsonic transports. Equivalent to a B747 or A380 replacement.
5	5	Vertical_Launch_Vehicle	Thomas	Sizing study of a vertical launch vehicle for space access. Heavy payload, multi stage, and serial stacking.
6	6	VTHL_Space_Access	Thomas	Sizing study on vertical takeoff and horizontal langing space access vehicle concepts.
7	7	FDL7	Thomas	A case study in the sizing of a FDL 7 class vehicle. Initial concept used for verificaiton. Trade studies conducted.
8	8	X51	Thomas	Sizing verificaiton study utilizing the X-51A flight vehicle. Purpose is initial verification but also used for trade study i...
9	9	GHV	Thomas	Reverse sizing of the GHV. Case study for system verificaiton. Trade study on solution concept. GHV was a growth ve...

AIDRA-DSS Project Builder v 1.01 - GPL3 License

Figure 4-23 Project Builder—Browser

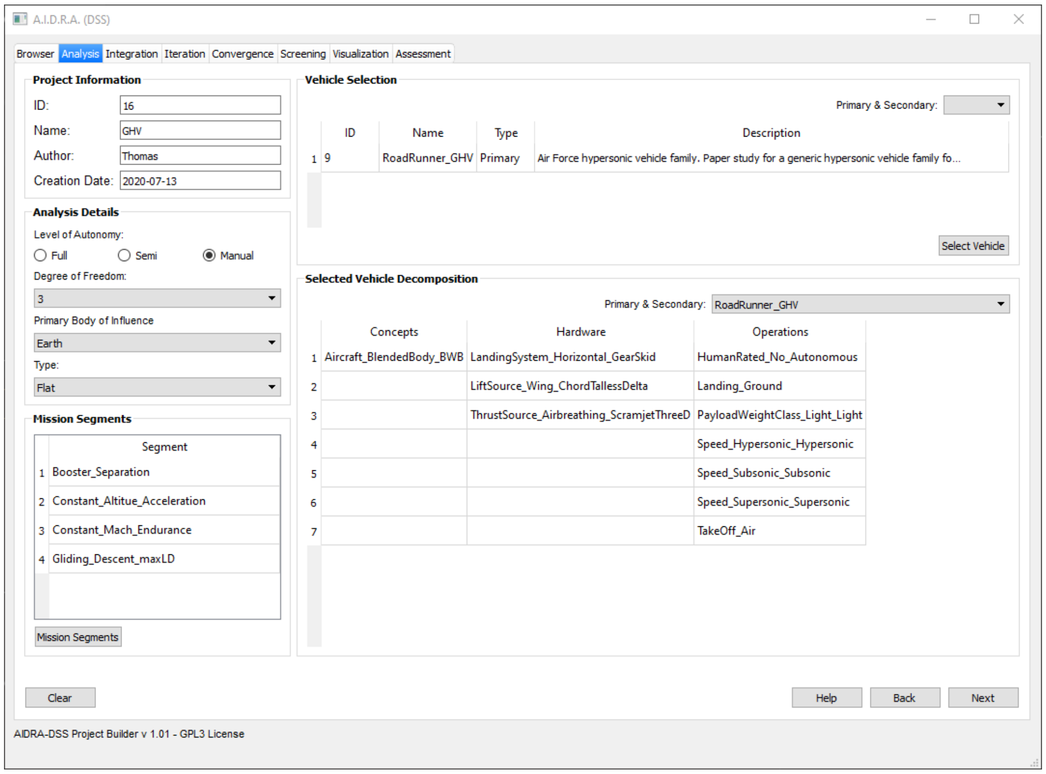


Name:	
Author:	
Creation Date:	2020-04-14
Description:	

Figure 4-24 Project Builder—New Project

4.3.5.2 Analysis

The <u>Analysis</u> tab presents the page wherein the user identifies the principal system(s) for analysis and the environment of analysis. The <u>Analysis</u> tab's content is shown in Figure 4-25. The page is separated into specific sections: Project Information, Analysis Details, Mission Segments, Vehicle Selection, and Selected Vehicle Decomposition.



Project Information

ID:	16
Name:	GHV
Author:	Thomas
Creation Date:	2020-07-13

Analysis Details

Level of Autonomy: Full Semi [✓ Manual]

Degree of Freedom: 3

Primary Body of Influence: Earth

Type: Flat

Mission Segments

	Segment
1	Booster_Separation
2	Constant_Altitue_Acceleration
3	Constant_Mach_Endurance
4	Gliding_Descent_maxLD

Vehicle Selection

Primary & Secondary:

	ID	Name	Type	Description
1	9	RoadRunner_GHV	Primary	Air Force hypersonic vehicle family. Paper study for a generic hypersonic vehicle family fo...

Selected Vehicle Decomposition

Primary & Secondary: RoadRunner_GHV

	Concepts	Hardware	Operations
1	Aircraft_BlendedBody_BWB	LandingSystem_Horizontal_GearSkid	HumanRated_No_Autonomous
2		LiftSource_Wing_ChordTallessDelta	Landing_Ground
3		ThrustSource_Airbreathing_ScramjetThreeD	PayloadWeightClass_Light_Light
4			Speed_Hypersonic_Hypersonic
5			Speed_Subsonic_Subsonic
6			Speed_Supersonic_Supersonic
7			TakeOff_Air

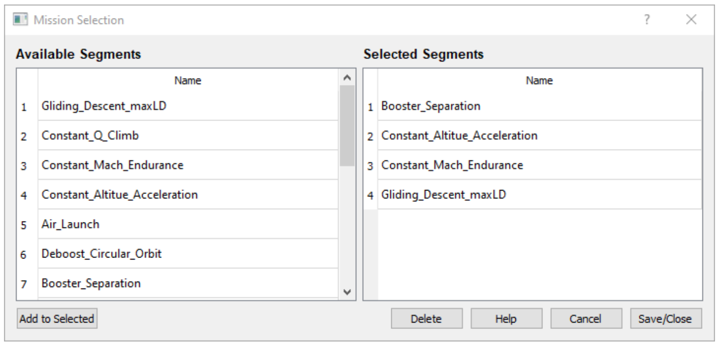
AIDRA-DSS Project Builder v 1.01 - GPL3 License

Figure 4-25 Project Builder—Analysis

The Project Information section, the top-left region of the page, contains the project specific information as generated and created in the new project form. This information is not editable and is repeated, in part, in the other tabs' page for reference.

Below the general project information, is the Analysis Details section. In this section, the user is required to select several analysis options. This includes the execution automation level (currently full and semi are not functional), the celestial body that governs the gravitational and atmospheric conditions, and the celestial body type assumption, that is, the case of flat, round, or spherical body assumption. These selections will set the environment of the vehicle's operation and method analysis type. These option selections will participate in the governing of the methods presented for selection in a later window.

Below the Analysis Details section, is the Mission Segments section. In this section, a table shows the mission segments (trajectory segments) selected for vehicle operation. To select the mission segments, the user clicks "Mission Segments" and the mission selection window opens, see Figure 4-26. In this window, the user selects the desired missions in the same manner as previously discussed for similar selection windows. The mission segments available directly correlate to the trajectory methods available. On save, the window is closed, the back-end database updated, and the mission segment selections displays in the Mission Segment section table.



Available Segments

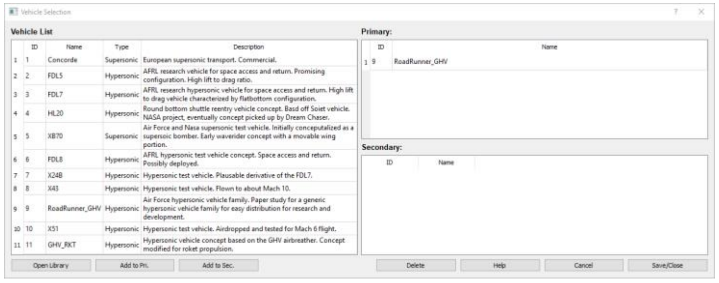
	Name
1	Gliding_Descent_maxLD
2	Constant_Q_Climb
3	Constant_Mach_Endurance
4	Constant_Altitue_Acceleration
5	Air_Launch
6	Deboost_Circular_Orbit
7	Booster_Separation

Selected Segments

	Name
1	Booster_Separation
2	Constant_Altitue_Acceleration
3	Constant_Mach_Endurance
4	Gliding_Descent_maxLD

Figure 4-26 Analysis—Mission Selection window

The section Vehicle Selection (top-right corner) displays the vehicle(s) selected for analysis. To select a vehicle(s), the user clicks "Select Vehicle", at which point, the vehicle selection window—Figure 4-27—displays. On the left side of the window, the vehicles available are shown. These are the vehicles from the *Vehicle Library*. The user must select a vehicle. Multiple vehicles can be selected; however, if multiple are selected, then there must be at least one primary vehicle. The user must correctly select the vehicles as primary or secondary. A primary vehicle is a vehicle that is independent but can consist of one or more secondary vehicles. A secondary vehicle is a vehicle or a system that acts like a vehicle (a distinguishable sub element such as a first stage in a multi-stage rocket) or is a distinguishable vehicle but is part of a total system that is considered itself as a vehicle. For example, the Falcon Heavy would be considered a primary vehicle consisting of multiple distinct secondary vehicles (the side stages, center core, and upper stage) that, in their



Vehicle List

	ID	Name	Type	Description
1	1	Concorde	Supersonic	European supersonic transport. Commercial.
2	2	FDL5	Hypersonic	AFRL research vehicle for space access and return. Promising configuration. High lift to drag ratio.
3	3	FDL7	Hypersonic	AFRL research hypersonic vehicle for space access and return. High lift to drag vehicle characterized by flatbottom configuration.
4	4	HL20	Hypersonic	Round bottom shuttle reentry vehicle concept. Basd off Soiet vehicle. NASA project, eventually concept picked up by Dream Chaser.
5	5	XB70	Supersonic	Air Force and Nasa supersonic test vehicle. Initially conceputalized as a supersoic bomber. Early waverider concept with a movable wing portion.
6	6	FDL8	Hypersonic	AFRL hypersonic test vehicle concept. Space access and return. Possibly deployed.
7	7	X24B	Hypersonic	Hypersonic test vehicle. Plausable derivative of the FDL7.
8	8	X43	Hypersonic	Hypersonic test vehicle. Flown to about Mach 10.
9	9	RoadRunner_GHV	Hypersonic	Air Force hypersonic vehicle family. Paper study for a generic hypersonic vehicle family for easy distribution for research and development.
10	10	X51	Hypersonic	Hypersonic test vehicle. Airdropped and tested for Mach 6 flight.
11	11	GHV_RKT	Hypersonic	Hypersonic vehicle concept based on the GHV airbreather. Concept modified for roket propulsion.

Primary:

	ID	Name
1	9	RoadRunner_GHV

Secondary:

ID	Name

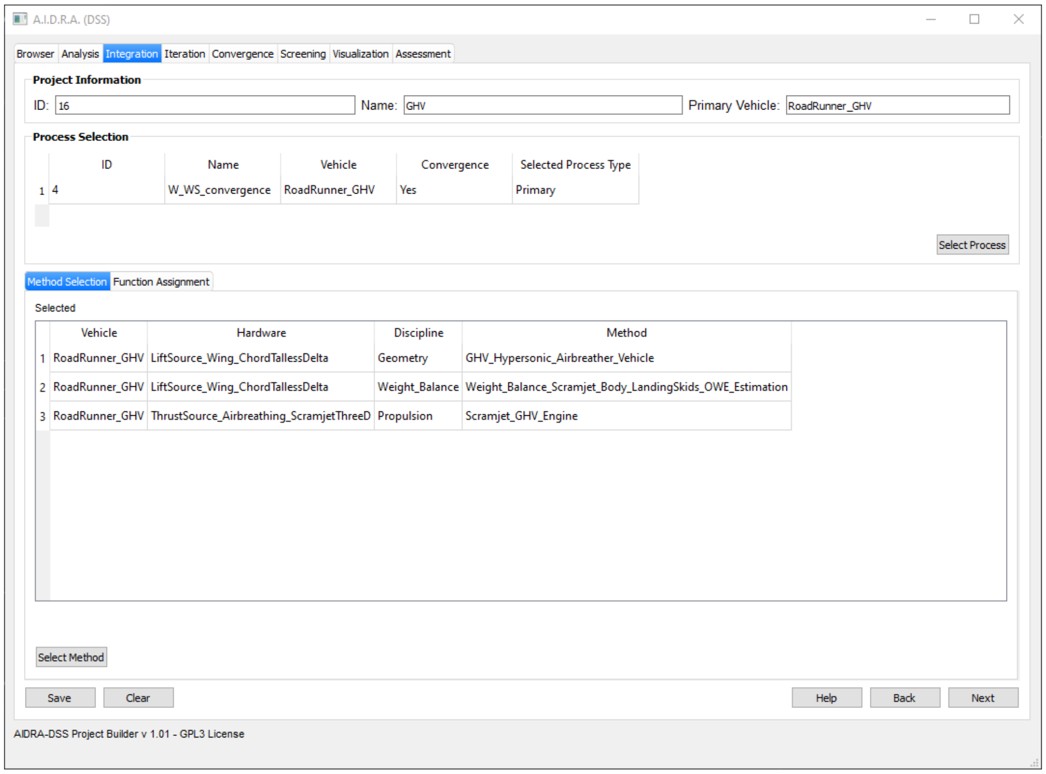
Figure 4-27 Analysis—Vehicle Selection window

own-right, can be treated as distinct vehicles with their own secondary missions and sizing processes. On completion of vehicle selection, the user clicks “save”, the data is added to the back-end, the window closes, and the vehicles are added to the Vehicle Selection table.

The Selected Vehicle Decomposition section displays the elemental constructs of the vehicles selected. The user can switch between the vehicles selected. The table displays the constructs selected during the vehicle build. The vehicle’s concept, hardware, and operation selections are shown. This is provided for self-review prior to moving to the next tab, Integration.

4.3.5.3 Integration

The Integration tab, Figure 4-28, contains three sections: Process Selection, Method Selection, and Function Assignment. These sections lead to the selection of the analysis’ processes, the selection of the analysis’ methods, and the association of hardware to function. The order of operation is to select the processes first, followed by the methods, and lastly the hardware-function assignment. Each is discussed next.



Project Information

ID: 16 Name: GHV Primary Vehicle: RoadRunner_GHV

Process Selection

	ID	Name	Vehicle	Convergence	Selected Process Type
1	4	W_WS_convergence	RoadRunner_GHV	Yes	Primary

Selected

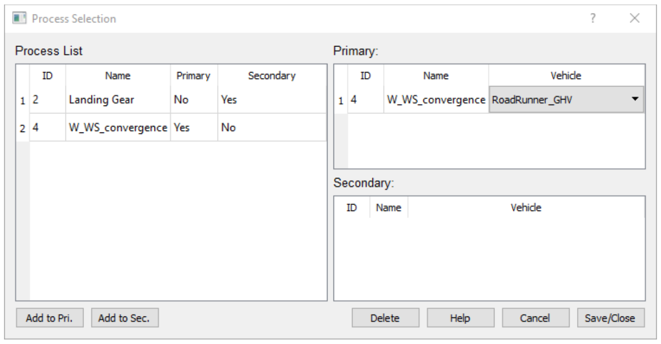
	Vehicle	Hardware	Discipline	Method
1	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Geometry	GHV_Hypersonic_Airbreather_Vehicle
2	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Weight_Balance	Weight_Balance_Scramjet_Body_LandingSkids_OWE_Estimation
3	RoadRunner_GHV	ThrustSource_Airbreathing_ScramjetThreeD	Propulsion	Scramjet_GHV_Engine

AIDRA-DSS Project Builder v 1.01 - GPL3 License

Figure 4-28 Project Builder—Integration

4.3.5.3.1 Process Selection

The user selects the analysis process in the Process Selection and Assignment window, Figure 4-29. Clicking the “Select Process” button will display the window. On the left side of the window is the Process List. The Process List is an itemization of the processes available for selection. The right side of the window contains two windows—Primary and Secondary. The primary and secondary terms correspond to the vehicle class not the process class. The process’ class (primary or secondary) is indicated in the Process List table. A process classification can be either secondary or primary and can be assigned to either a primary or a secondary vehicle. (Recall, a primary process is a process that governs the closure of a vehicle design and can but is not required to contain a sub-process. A secondary process is a process that operates within the bounds of a primary process.) To assign a process, the user selects the process in the Process List table and then clicks either “Add to Pri.” or “Add to Sec.” to add the process to the Primary or Secondary table, respectively. After selecting the process, the user assigns the process to a vehicle by selecting the desired vehicle from the drop-down window available in the Primary or Secondary table depending on the user’s selection. Only vehicles selected during the Analysis page operation will be available as options to the user. When the process selection and assignment to a vehicle is complete, the user selects “Save/Close” at which point the selections made are saved to the back-end database and the window closes. The process-vehicle selections made are displayed in the Integration page’s Process Selection table.



Process List

	ID	Name	Primary	Secondary
1	2	Landing Gear	No	Yes
2	4	W_WS_convergence	Yes	No

Primary:

	ID	Name	Vehicle
1	4	W_WS_convergence	RoadRunner_GHV

Secondary:

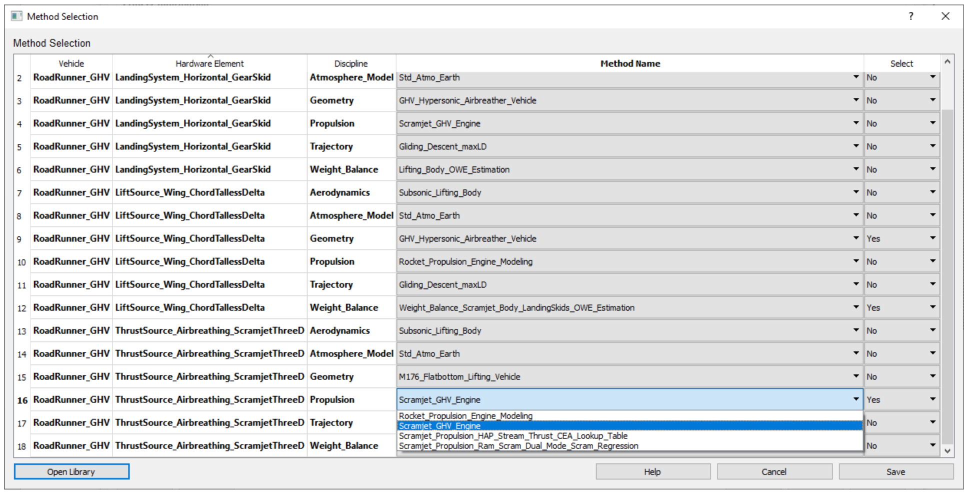
ID	Name	Vehicle

Figure 4-29 Integration—Process Selection and Assignment window

4.3.5.3.2 Method Selection

Method selection occurs through the Method Selection window—Figure 4-30. The window is accessed by selecting “Select Method” under the Method Selection tab. On button click, the window displays and is populated.

Methods are assigned according to vehicle-hardware-discipline association. The vehicle(s) previously selected and its hardware populates the Method Selection window. Each hardware has the option to be assigned a single method per process discipline (if more than one method is necessary, all hardware-discipline methods for that discipline should be assigned under the Iteration tab). The methods available per hardware per discipline display in a dropdown menu in the Method Name column. To assign a method and activate the analysis option, the user must select a method from the menu in the Method Name column and select “Yes” under the Select column. The user can decline a hardware-discipline analysis by selecting “No” under the Select column; in this case, the method displayed in the Method Name column is non-consequential. By reviewing the Select column’s entries, the user can review to what degree a given hardware is being considered in the analysis (hardware-discipline accountability). On completion of method assignment, the user selects “Save” and the user’s selections are saved to the back-end database, the window closes, and the vehicle-hardware-method selections are displayed in the Integration page's "Selected" table section under the Method Selection tab.


Method Selection
Method Selection
Rocket_Propulsion_Engine_Modeling
Scramjet_GHV_Engine
Scramjet_Propulsion_HAP_Stream_Thrust_CEA_Lookup_Table
Scramjet_Propulsion_Ram_Scram_Dual_Mode_Scram_Regression
Open Library
Help
Cancel
Save

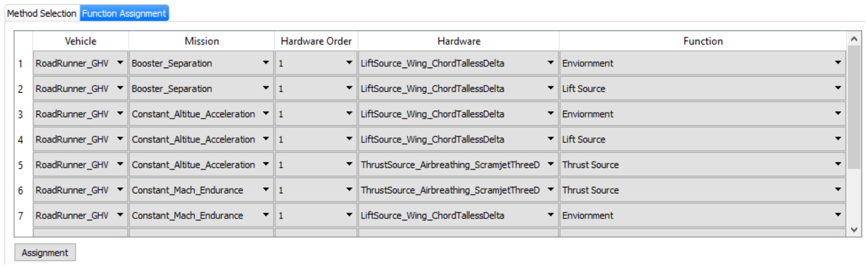
	Vehicle	Hardware Element	Discipline	Method Name	Select
2	RoadRunner_GHV	LandingSystem_Horizontal_GearSkid	Atmosphere_Model	Std_Atmo_Earth	No
3	RoadRunner_GHV	LandingSystem_Horizontal_GearSkid	Geometry	GHV_Hypersonic_Airbreather_Vehicle	No
4	RoadRunner_GHV	LandingSystem_Horizontal_GearSkid	Propulsion	Scramjet_GHV_Engine	No
5	RoadRunner_GHV	LandingSystem_Horizontal_GearSkid	Trajectory	Gliding_Descent_maxLD	No
6	RoadRunner_GHV	LandingSystem_Horizontal_GearSkid	Weight_Balance	Lifting_Body_OWE_Estimation	No
7	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Aerodynamics	Subsonic_Lifting_Body	No
8	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Atmosphere_Model	Std_Atmo_Earth	No
9	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Geometry	GHV_Hypersonic_Airbreather_Vehicle	Yes
10	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Propulsion	Rocket_Propulsion_Engine_Modeling	No
11	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Trajectory	Gliding_Descent_maxLD	No
12	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Weight_Balance	Weight_Balance_Scramjet_Body_LandingSkids_OWE_Estimation	Yes
13	RoadRunner_GHV	ThrustSource_Airbreathing_ScramjetThreeD	Aerodynamics	Subsonic_Lifting_Body	No
14	RoadRunner_GHV	ThrustSource_Airbreathing_ScramjetThreeD	Atmosphere_Model	Std_Atmo_Earth	No
15	RoadRunner_GHV	ThrustSource_Airbreathing_ScramjetThreeD	Geometry	M176_Flatbottom_Lifting_Vehicle	No
16	RoadRunner_GHV	ThrustSource_Airbreathing_ScramjetThreeD	Propulsion	Scramjet_GHV_Engine	Yes
17	RoadRunner_GHV	ThrustSource_Airbreathing_ScramjetThreeD	Trajectory		No
18	RoadRunner_GHV	ThrustSource_Airbreathing_ScramjetThreeD	Weight_Balance		No

Figure 4-30 Integration—Method Selection window

4.3.5.3.3 Function Assignment

The Function Assignment tab—shown in Figure 4-31—is the interface for the user to assign a vehicle's hardware a function mode (its purpose) and assign that functionality to a given mission segment. The function type is set in the Function column. A drop-down menu shows the available functions (Lift Source, Thrust Source, TPS, etc.). If multiple hardware provides the same function for the same mission, their order of operation (simultaneous or sequential) is assigned via the value set in the Hardware Order column. The order is in ascending order, that is, the lower value associated hardware function occurs first.

Additionally, this section indirectly sets the mission segments per vehicle. As such, the user must assure that all required mission segments per vehicle are associated. The mission options available are from the list selected in the Analysis tab. To add a row, and therefore a mission segment, select "Assignment". This will add a single row. Furthermore, note that there is a direct dependency between mission segment and hardware function; this means that for all mission segments the vehicle must have some hardware performing a function whether it be thrust, lift, thermal protection, or some other.



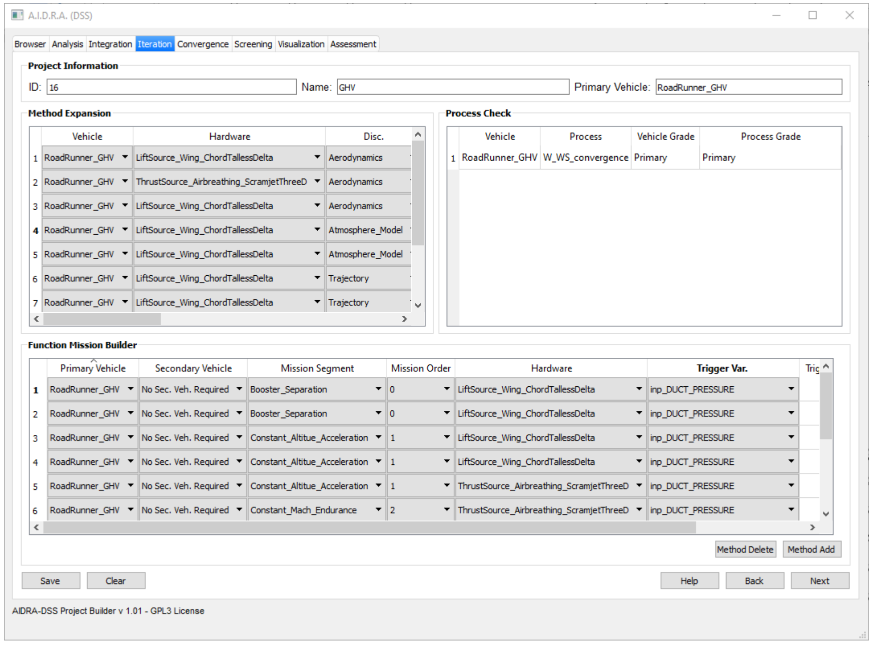
	Vehicle	Mission	Hardware Order	Hardware	Function
1	RoadRunner_GHV	Booster_Separation	1	LiftSource_Wing_ChordTallessDelta	Enviornment
2	RoadRunner_GHV	Booster_Separation	1	LiftSource_Wing_ChordTallessDelta	Lift Source
3	RoadRunner_GHV	Constant_Altitue_Acceleration	1	LiftSource_Wing_ChordTallessDelta	Enviornment
4	RoadRunner_GHV	Constant_Altitue_Acceleration	1	LiftSource_Wing_ChordTallessDelta	Lift Source
5	RoadRunner_GHV	Constant_Altitue_Acceleration	1	ThrustSource_Airbreathing_ScramjetThreeD	Thrust Source
6	RoadRunner_GHV	Constant_Mach_Endurance	1	ThrustSource_Airbreathing_ScramjetThreeD	Thrust Source
7	RoadRunner_GHV	Constant_Mach_Endurance	1	LiftSource_Wing_ChordTallessDelta	Enviornment

Figure 4-31 Integration—Function Assignment tab

By the completion of the Function Assignment tab, the vehicle has its mission segments specified, hardware per mission segment operation type and variable range defined, and, in the event of multiple hardware with the same function mode, the specification of operational order. Not defined however, is the mission order. The mission order is set during the Iteration page operation.

4.3.5.4 Iteration:

The *Iteration* tab—Figure 4-32—is the environment in which the user defines the parameters that refine the analysis process for proper code assembly. There are two primary objectives: (1) expand methods per hardware per discipline if necessary and (2) formulize the trajectory. These objectives are fulfilled by the interactions within the Method Expansion and Functional Mission Builder sections, respectively.



Project Information

ID: 16 Name: GHV Primary Vehicle: RoadRunner_GHV

Method Expansion

	Vehicle	Hardware	Disc.
1	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Aerodynamics
2	RoadRunner_GHV	ThrustSource_Airbreathing_ScramjetThreeD	Aerodynamics
3	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Aerodynamics
4	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Atmosphere_Model
5	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Atmosphere_Model
6	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Trajectory
7	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Trajectory

Process Check

	Vehicle	Process	Vehicle Grade	Process Grade
1	RoadRunner_GHV	W_WS_convergence	Primary	Primary

Function Mission Builder

	Primary Vehicle	Secondary Vehicle	Mission Segment	Mission Order	Hardware	**Trigger Var.**	Trig
1	RoadRunner_GHV	No Sec. Veh. Required	Booster_Separation	0	LiftSource_Wing_ChordTallessDelta	inp_DUCT_PRESSURE	
2	RoadRunner_GHV	No Sec. Veh. Required	Booster_Separation	0	LiftSource_Wing_ChordTallessDelta	inp_DUCT_PRESSURE	
3	RoadRunner_GHV	No Sec. Veh. Required	Constant_Altitue_Acceleration	1	LiftSource_Wing_ChordTallessDelta	inp_DUCT_PRESSURE	
4	RoadRunner_GHV	No Sec. Veh. Required	Constant_Altitue_Acceleration	1	LiftSource_Wing_ChordTallessDelta	inp_DUCT_PRESSURE	
5	RoadRunner_GHV	No Sec. Veh. Required	Constant_Altitue_Acceleration	1	ThrustSource_Airbreathing_ScramjetThreeD	inp_DUCT_PRESSURE	
6	RoadRunner_GHV	No Sec. Veh. Required	Constant_Mach_Endurance	2	ThrustSource_Airbreathing_ScramjetThreeD	inp_DUCT_PRESSURE	

AIDRA-DSS Project Builder v 1.01 - GPL3 License

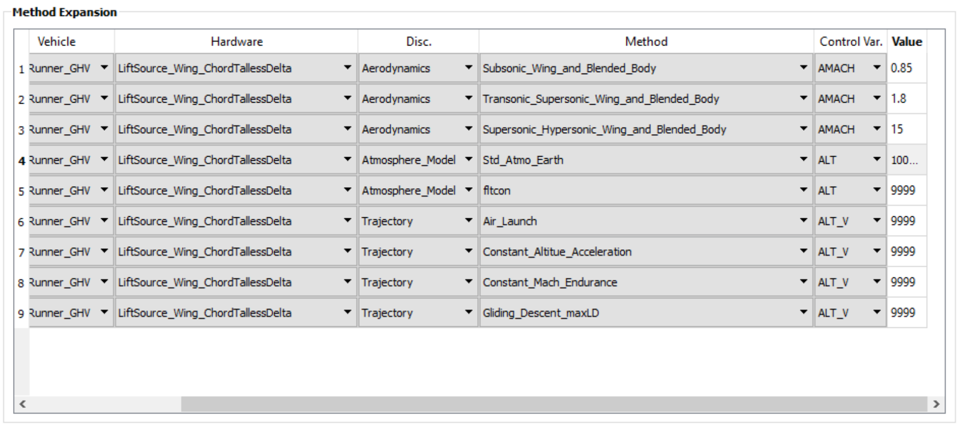
Figure 4-32 Project Builder—Iteration Page

A secondary objective of the Iteration page is to verify the vehicle-process selection. This is done by review of the information presented in the Process Check section. Here, the selected vehicle and associated process' grade are indicated (primary or secondary). All vehicles should have a process and the primary vehicle must have a primary process associated. The user should review the presented selections for correctness; this is a manual verification process.

4.3.5.4.1 Method Expansion

The Method Expansion section provides the user the option to add or assign multiple methods to a vehicle-hardware-discipline association. As visible in Figure 4-33, the Method Expansion area is a table populated with drop down menus for the user to associate a new method as in a similar manner to previously done. However, the user must now address a new condition, the conditionality of multiple methods per the same device. This is addressed through the selection of a control variable and variable value.

The control variable is a method variable that controls the operational execution of the multiple methods. The control variable is selectable from the method input variables. The methods execution is controlled by the value of the control variable as set in the Value column. For example, if there are three aerodynamic methods (subsonic, transonic, and supersonic) then a control variable could be the Mach number with control variable values of 0.85, 1.25, and 7. In this case, the subsonic method would execute so long as the Mach number is less than 0.85. The transonic method would execute for Mach numbers between 0.85 and 1.25. The supersonic method would execute for Mach number values greater than 1.25 and less than 7. In this way, the user controls the application range of a method in a multi-method set.

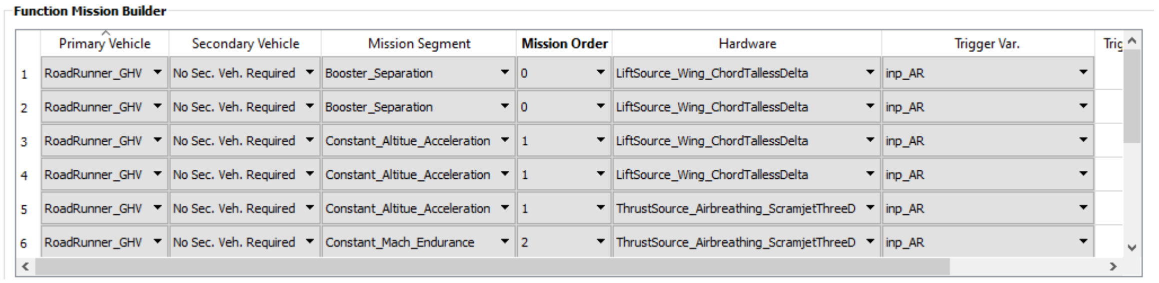

Method Expansion

	Vehicle	Hardware	Disc.	Method	Control Var.	**Value**
1	Runner_GHV	LiftSource_Wing_ChordTallessDelta	Aerodynamics	Subsonic_Wing_and_Blended_Body	AMACH	0.85
2	Runner_GHV	LiftSource_Wing_ChordTallessDelta	Aerodynamics	Transonic_Supersonic_Wing_and_Blended_Body	AMACH	1.8
3	Runner_GHV	LiftSource_Wing_ChordTallessDelta	Aerodynamics	Supersonic_Hypersonic_Wing_and_Blended_Body	AMACH	15
4	Runner_GHV	LiftSource_Wing_ChordTallessDelta	Atmosphere_Model	Std_Atmo_Earth	ALT	100...
5	Runner_GHV	LiftSource_Wing_ChordTallessDelta	Atmosphere_Model	fltcon	ALT	9999
6	Runner_GHV	LiftSource_Wing_ChordTallessDelta	Trajectory	Air_Launch	ALT_V	9999
7	Runner_GHV	LiftSource_Wing_ChordTallessDelta	Trajectory	Constant_Altitue_Acceleration	ALT_V	9999
8	Runner_GHV	LiftSource_Wing_ChordTallessDelta	Trajectory	Constant_Mach_Endurance	ALT_V	9999
9	Runner_GHV	LiftSource_Wing_ChordTallessDelta	Trajectory	Gliding_Descent_maxLD	ALT_V	9999

Figure 4-33 Iteration—Method Expansion section

4.3.5.4.2 Function Mission Builder

The Function Mission Builder section is the area in which the user defines the vehicle's mission. The Function Mission Builder is populated with the mission segments selected in the Integration page's Function Assignment tab. In this section, the user specifies the parent-child vehicle relationship, assigns a mission segment and order value to a vehicle, and specifies a trigger condition if necessary.


Function Mission Builder

	Primary Vehicle	Secondary Vehicle	Mission Segment	**Mission Order**	Hardware	Trigger Var.	Trig
1	RoadRunner_GHV	No Sec. Veh. Required	Booster_Separation	0	LiftSource_Wing_ChordTallessDelta	inp_AR	
2	RoadRunner_GHV	No Sec. Veh. Required	Booster_Separation	0	LiftSource_Wing_ChordTallessDelta	inp_AR	
3	RoadRunner_GHV	No Sec. Veh. Required	Constant_Altitue_Acceleration	1	LiftSource_Wing_ChordTallessDelta	inp_AR	
4	RoadRunner_GHV	No Sec. Veh. Required	Constant_Altitue_Acceleration	1	LiftSource_Wing_ChordTallessDelta	inp_AR	
5	RoadRunner_GHV	No Sec. Veh. Required	Constant_Altitue_Acceleration	1	ThrustSource_Airbreathing_ScramjetThreeD	inp_AR	
6	RoadRunner_GHV	No Sec. Veh. Required	Constant_Mach_Endurance	2	ThrustSource_Airbreathing_ScramjetThreeD	inp_AR	

Figure 4-34 Iteration Page—Function Mission Builder Section

The vehicle parent-child relationships are controlled through the Primary Vehicle and Secondary Vehicle column selections. If the user selected more than one vehicle during the operation of the Analysis tab, then here the user specifies the vehicle relationships. The primary vehicle or parent is selected in the Primary Vehicle column. All vehicles are listed as options. The secondary vehicle or child vehicle is specified in the Secondary Vehicle column. A secondary vehicle can be a primary vehicle as well. During the operation of the Analysis tab, if only one vehicle were selected, then the Second Vehicle column will display “No Sec. Veh. Required” and the user must not make any selection. All vehicles and their dependencies must be assigned mission segments.

For each mission segment, the user is required to assign a value for the Mission Order. The mission order defines the order of mission segment occurrence in the total mission. The value assigned expresses the order of operation; the lower the value the earlier the mission segment occurs. The mission order per vehicle is sequential. In the case of the multi-vehicle, the values need to coincide if the vehicles (primary and secondary) operate concurrently or as a single system. At the point of a multi-vehicle disintegration, the mission segment and order per vehicle does not need to agree. In this way, the user can define a system of vehicles that operates concurrently as a single system for a specified set of mission segments, but that can also operate disintegrated as independent individual systems—with unique mission segments—at a predetermined point in the total mission trajectory.

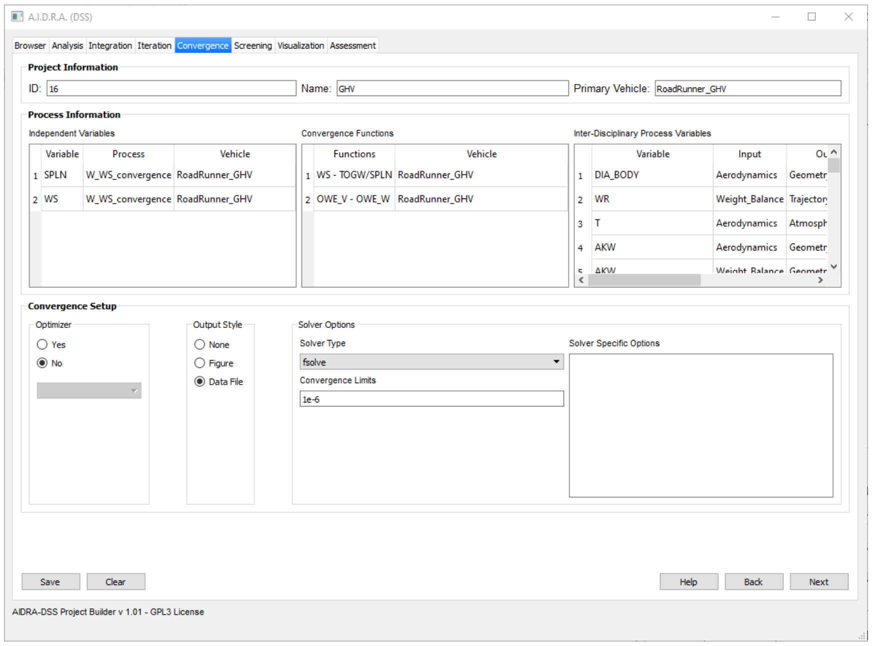
The trigger variable is a user-selected variable that specifies a variable dependency for the mission segment execution. The mission segment execution is controlled by the value of the trigger variable. In this way, similar to the case of multiple methods for single hardware, the user can set switches to control the mission segment analysis.

If a trigger variable is not required, then the user must set the Trigger Value entry to NA and set the Trigger Var. to any variable, the Trigger Var. is nonconsequential. In this case, all mission trajectory control will be accomplished through the standard method inputs for the mission segment. The inputs are set in the Screening tab.

4.3.5.5 Convergence

The Convergence tab—Figure 4-35—has two purposes. The purpose is to display the process information (as visible in the Process Information section) and, as visible in the Convergence Setup section, present the user with a means to control more directly the convergence execution.

The Process Information section presents the user with three information portals: Independent Variables, Convergence Functions, and Inter-Process Disciplinary Variables. These three portals are setup for system transparency and system review. Recall that a developmental objective is solution system transparency. As such, here the user is presented for review and edification, the independent variables of the process, the convergence functions of the process selected, and the process variables that are part of the multidisciplinary process (inter-disciplinary variables). The interdisciplinary variables account for the interconnectivity of the disciplinary analysis and changes with method selection. From this information, the user can track variable influence and the degree of discipline dependencies. This is significant for identifying and tracking potential design driving variables and allows for a means of assessment on method selection for multidisciplinary variable integration into the concept design. The Process Information section presents information; the user is required to review the information for impact, significance, and correctness.



Project Information

ID: 16 Name: GHV Primary Vehicle: RoadRunner_GHV

Process Information

Independent Variables

	Variable	Process	Vehicle
1	SPLN	W_WS_convergence	RoadRunner_GHV
2	WS	W_WS_convergence	RoadRunner_GHV

Convergence Functions

	Functions	Vehicle
1	WS - TOGW/SPLN	RoadRunner_GHV
2	OWE_V - OWE_W	RoadRunner_GHV

Inter-Disciplinary Process Variables

	Variable	Input	Ou
1	DIA_BODY	Aerodynamics	Geometr
2	WR	Weight_Balance	Trajector
3	T	Aerodynamics	Atmosph
4	AKW	Aerodynamics	Geometr
5	AKW	Weight_Balance	Geometr

Convergence Setup

Optimizer: Yes [✓ No]

Output Style: None Figure [✓ Data File]

Solver Options

Solver Type: fsolve

Convergence Limits: 1e-6

Solver Specific Options

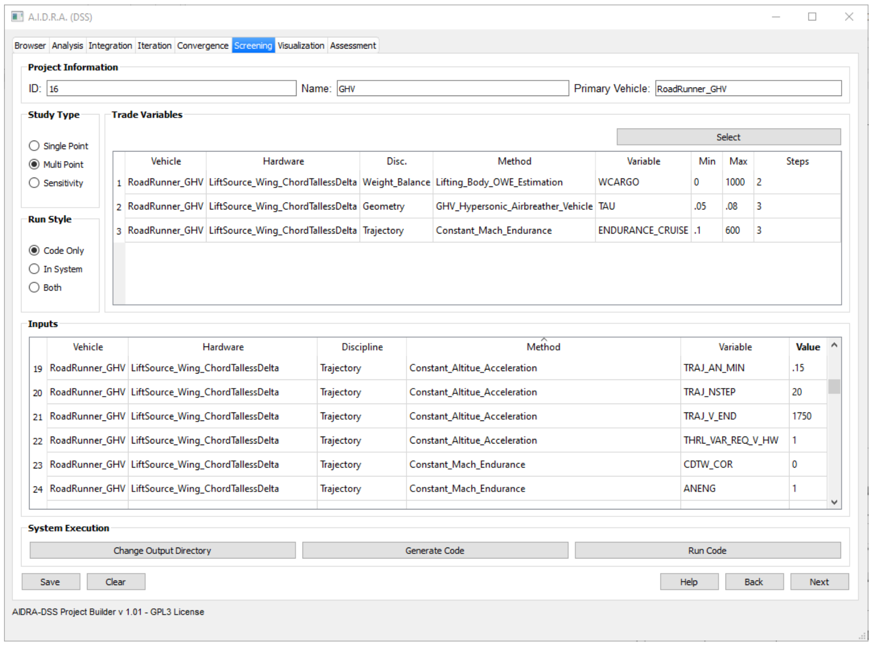
AIDRA-DSS Project Builder v 1.01 - GPL3 License

Figure 4-35 Project Builder—Convergence Page

The Convergence Setup section is the interface for refining the convergence process and user feedback. The user is able to select the convergence output style and specify solver options through the Output Style and Solver Options sections respectively. The output style refers to how the convergence output is handled in regard to the user; the user can select to receive no specific feedback, a data file of the convergence function outputs per iteration, or a real-time plot of the convergence data. The Solver Option section is where the user can specify a specific solver from a set of options, specify the numerical zero value for the solver and user specific solver options. The user specific options interface is a text input area; the text must be in proper python syntax for the solver selected.

4.3.5.6 Screening

The Screening tab—Figure 4-36—is a significant action location. Three critical tasks are accomplished: trade study specification, input variables' value specification, and system generation. The page's tasks are categorized horizontally into three rows corresponding to each task.


A.I.D.R.A. (DSS)

Browser | Analysis | Integration | Iteration | Convergence | Screening | Visualization | Assessment

Project Information

ID: 16 Name: GHV Primary Vehicle: RoadRunner_GHV

Study Type

Single Point [✓ Multi Point] Sensitivity

Trade Variables

	Vehicle	Hardware	Disc.	Method	Variable	Min	Max	Steps
1	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Weight_Balance	Lifting_Body_OWE_Estimation	WCARGO	0	1000	2
2	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Geometry	GHV_Hypersonic_Airbreather_Vehicle	TAU	.05	.08	3
3	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Trajectory	Constant_Mach_Endurance	ENDURANCE_CRUISE	.1	600	3

Run Style

[✓ Code Only] In System Both

Inputs

	Vehicle	Hardware	Discipline	Method	Variable	Value
19	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Trajectory	Constant_Altitue_Acceleration	TRAJ_AN_MIN	.15
20	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Trajectory	Constant_Altitue_Acceleration	TRAJ_NSTEP	20
21	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Trajectory	Constant_Altitue_Acceleration	TRAJ_V_END	1750
22	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Trajectory	Constant_Altitue_Acceleration	THRL_VAR_REQ_V_HW	1
23	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Trajectory	Constant_Mach_Endurance	CDTW_COR	0
24	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Trajectory	Constant_Mach_Endurance	ANENG	1

System Execution

AIDRA-DSS Project Builder v 1.01 - GPL3 License

Figure 4-36 Project Builder—Screening tab's page

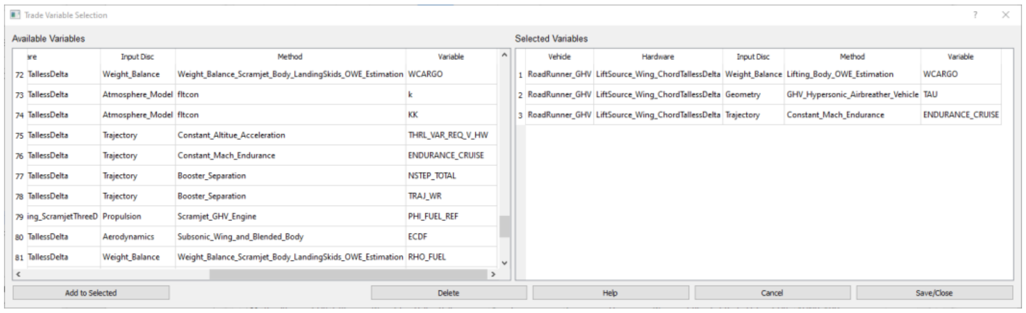
4.3.5.6.1 Trade Study Specification

The first task is trade study specification. Trade study specification occurs through the Study Type and Trade Variables sections selections and range specifications. The study type, selected in the Study Type section, can be selected as single point, multi-point, or sensitivity. The first two are self-explanatory. The sensitivity type is a type of multi-point but with a specific purpose of identifying variables of high influence on the design solution and does not allow user selection of trade variables.

For a multipoint study type, the trade study variable selection and specification occurs through the operation of the Trade Variables section. There are two steps to setting up the trade study: selecting the trade variables and identifying the trade variable range and step count.

The user must select the trade variables. The trade variables are selected through the Trade Variable Selection window—Figure 4-37—that is opened by clicking "Select" in the Trade Variables section. The Trade Variables Selection window presents the user with a list of available variables—the left-hand table. From this list, the user selects the trade variables desired and clicks "Add to Selected". On click, the variables selected are displayed in the Selected Variables section's table. When all trade variables are selected, the user clicks "Save/Close" at which point the data is saved and presented in the Trade Variables section's table in the Screening tab.

The selecting of the variables completes the first step; the second step is to specify the variable trade values. With the trade variables selected the user must enter the, the user enters the minimum, maximum, and desired data points between the minimum and maximum specified (including the boundary values). The user enters the values directly in the Trade Variables section's table.



Available Variables

	ıre	Input Disc	Method	Variable
72	TallessDelta	Weight_Balance	Weight_Balance_Scramjet_Body_LandingSkids_OWE_Estimation	WCARGO
73	TallessDelta	Atmosphere_Model	fltcon	k
74	TallessDelta	Atmosphere_Model	fltcon	KK
75	TallessDelta	Trajectory	Constant_Altitue_Acceleration	THRL_VAR_REQ_V_HW
76	TallessDelta	Trajectory	Constant_Mach_Endurance	ENDURANCE_CRUISE
77	TallessDelta	Trajectory	Booster_Separation	NSTEP_TOTAL
78	TallessDelta	Trajectory	Booster_Separation	TRAJ_WR
79	ing_ScramjetThreeD	Propulsion	Scramjet_GHV_Engine	PHI_FUEL_REF
80	TallessDelta	Aerodynamics	Subsonic_Wing_and_Blended_Body	ECDF
81	TallessDelta	Weight_Balance	Weight_Balance_Scramjet_Body_LandingSkids_OWE_Estimation	RHO_FUEL

Selected Variables

	Vehicle	Hardware	Input Disc	Method	Variable
1	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Weight_Balance	Lifting_Body_OWE_Estimation	WCARGO
2	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Geometry	GHV_Hypersonic_Airbreather_Vehicle	TAU
3	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Trajectory	Constant_Mach_Endurance	ENDURANCE_CRUISE

Figure 4-37 Screening Page—Trade Variable Selection window

4.3.5.6.2 Input Value Specification

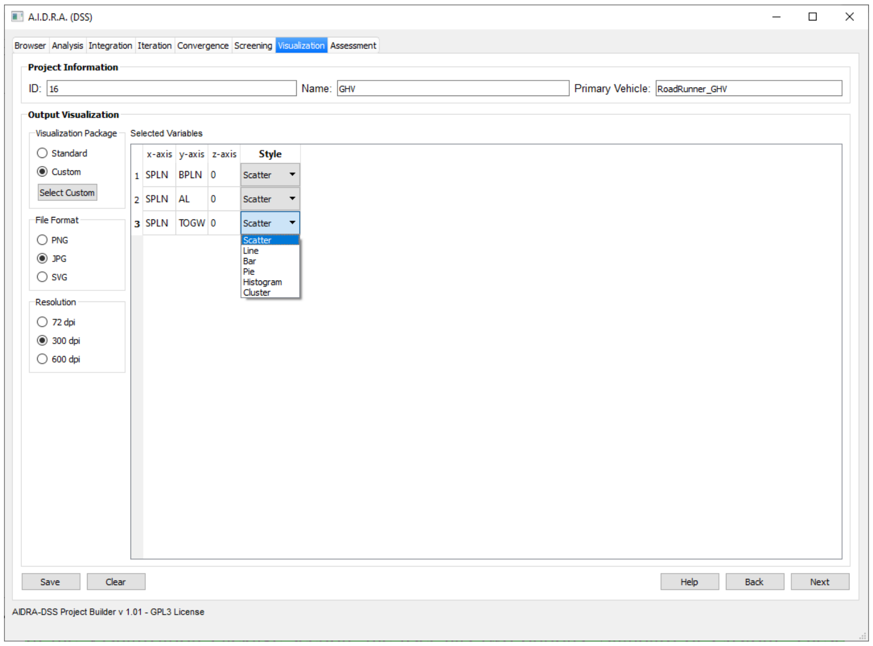
The second third of the Screening window comprises of a table of input variables, as determined by the system, where in the user is to specify the values. Through an evaluation of all methods selected and their place within the order of operations in the process selected, the system determines which variables require an initial value from the user. These variables are uniquely different from the interdisciplinary variables discussed previously in the Convergence tab. The user enters values for each variable at this interface. On the click of "next" or any of the generation buttons (discussed below) the input values entered by the user are saved to the backend database and are automatically entered into the synthesis script on generation. However, if the user clicks "back", the data is not retained, as each time the Screening window is displayed the required input variables are reevaluated and presented.

4.3.5.6.3 System Execution

The bottom third of the page contains the System Execution section. At this point, the user is presented with a principal option. The user is able to generate the code as standalone but that is not executed at the time of generation, or the user can generate the code and run it subsequently but with no tasked figure generation. At this point all necessary elements to generate the synthesis code have been specified. If the user chooses not to generate it at this point, it will be generated by default after the Visualization tab's page completion. Additionally, the user has the option to change the output folder. By default, the output folder is relative to the location of the GUI's script file as described in the earlier folder tree discussion. There is the option to not generate the code at this point if the user favors to set a desired result visualization scheme, which is set in the next tab—Visualization. If the user decides to generate and execute the synthesis generation at this point, then no result data analysis will be conducted automatically; a database of result data will be generated for later evaluation and analysis upon synthesis code execution.

4.3.5.7 Visualization

The Visualization page is the interface in which the user identifies the decision-making support figures to be generated. The page is shown in Figure 4-38. The user can specify the file type and image resolution in the File Format and Resolution sections respectively (located on the left-hand side of the page). The Visualization Package and Selected Variables sections are where in the user defines the figures to be generated.



Project Information

ID: 16 Name: GHV Primary Vehicle: RoadRunner_GHV

Output Visualization

Visualization Package: Standard [✓ Custom]

File Format: PNG [✓ JPG] SVG

Resolution: 72 dpi [✓ 300 dpi] 600 dpi

Selected Variables

	x-axis	y-axis	z-axis	Style
1	SPLN	BPLN	0	Scatter
2	SPLN	AL	0	Scatter
3	SPLN	TOGW	0	Scatter

AIDRA-DSS Project Builder v 1.01 - GPL3 License

Figure 4-38 Project Builder—Visualization Page

4.3.5.7.1 Visualization Package

In the Visualization Package section, the user has two options: Standard or Custom. These options control the variables and figure types that are to be generated. The user can adjust the standard package. Currently, the standard figures address study success (convergence iterations per trade and final convergence error per iteration), in addition to geometric and weight design variable depiction. Ideally, the standard package would contain a set of preselected figures that should address some standard design questions for a given problem relevant to the conceptual designer; this set could and would vary depending on the problem/vehicle type. Table 4-3 summarizes a standard figure package set. The Custom option is selected if the user wishes to specify the visualization output exactly.

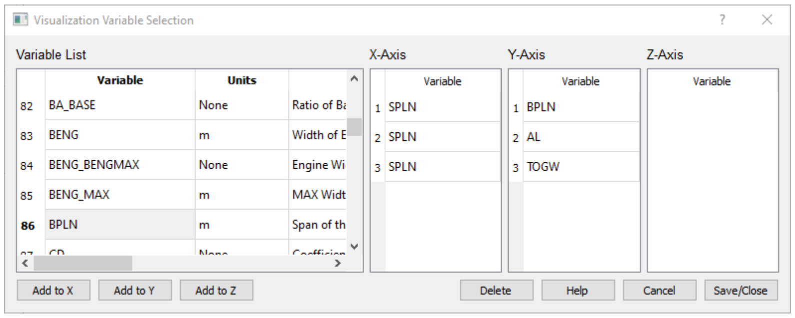
Table 4-3 Visualization Standard Package figure set

Field	**Description**	**Type**
Geometry/Weight	Standard sizing figure of S_{pln} versus $TOGW$; trade study inclusive	Scatter-Line
Geometry	Vehicle length versus span; trade study inclusive	Scatter-Line
Convergence	Depiction of convergence criteria per solver iteration; trade study inclusive; $f(Process\ independent\ variable, error\ function\ value)$	Scatter-Line
Study Summary	Presentation of all converged and non-converged points and execution error log check; rapid identification of non-solution iterations	Bar

To initiate the custom output option, the user clicks “Select Custom”, at which point the “Custom” radial option is set active and the variable selection window—Figure 4-39—displays. The window is separated into a table of variables available and the set of selected variables and their corresponding figure axis assignment. The user can plot up to three different variables. The user selects a variable from the “Variable List” and assigns it to an axis by clicking “Add to X”, “Add to Y”, or “Add to Z” referring to the corresponding X, Y, and Z axis, respectively. On completion of variable selection, the user clicks “Save/Close”; at this point the user’s selections are saved to the backend database and then displayed in the Visualization tab's Selected Variables section.

4.3.5.7.2 Selected Variables

The Selected Variables section presents to the user the variables to be visualized and the style of the visualization. Each row corresponds to a single figure. The variables for the x, y, and z axes are shown as selected in the Visualization Variable Selection form. The interaction required from the user is to select the figure type. Here the user is given a drop-down menu in the Style column, for each figure, where the user is to select the figure style. Style options include Scatter, Line, Bar, Pie, Histogram, and Cluster. The user must select one.



	Variable	Units	
82	BA_BASE	None	Ratio of Ba
83	BENG	m	Width of E
84	BENG_BENGMAX	None	Engine Wi
85	BENG_MAX	m	MAX Widt
86	BPLN	m	Span of th
87	CD	None	Coefficien

X-Axis

	Variable
1	SPLN
2	SPLN
3	SPLN

Y-Axis

	Variable
1	BPLN
2	AL
3	TOGW

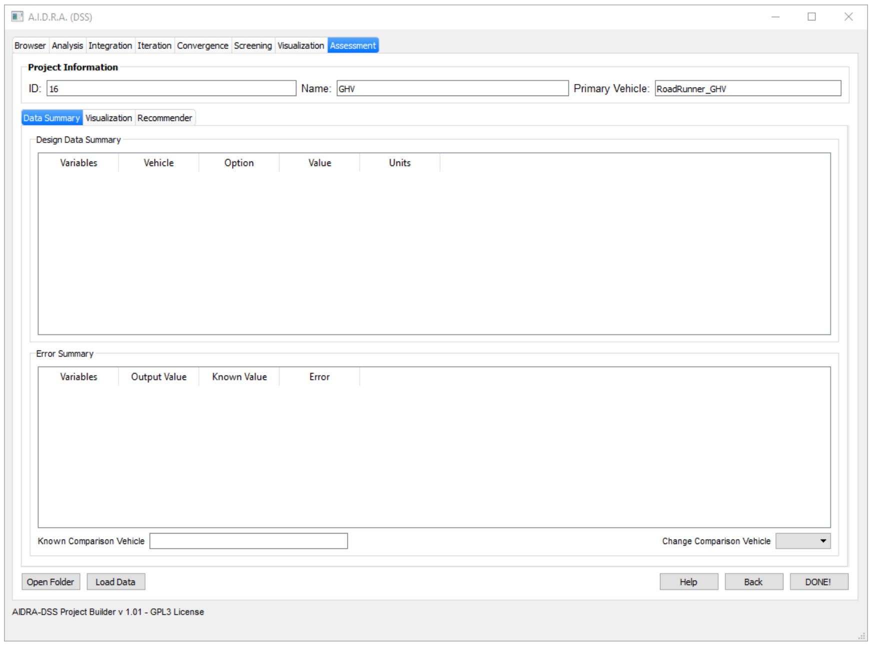
Z-Axis

Variable

Figure 4-39 Visualization—Figure Variable Selection window

4.3.5.8 Assessment

The Assessment tab, Figure 4-40, has the purpose of presenting the user with an interface for reviewing results for accuracy, reviewing results for design insights, and reviewing design recommendations. To accomplish this, the window contains three different tabs, each individually addressing a purpose: Data Summary, Visualization, and Recommendations.



ID: 16 Name: GHV Primary Vehicle: RoadRunner_GHV

Variables	Vehicle	Option	Value	Units

Variables	Output Value	Known Value	Error

AIDRA-DSS Project Builder v 1.01 - GPL3 License

Figure 4-40 Assessment page—Data Summary tab

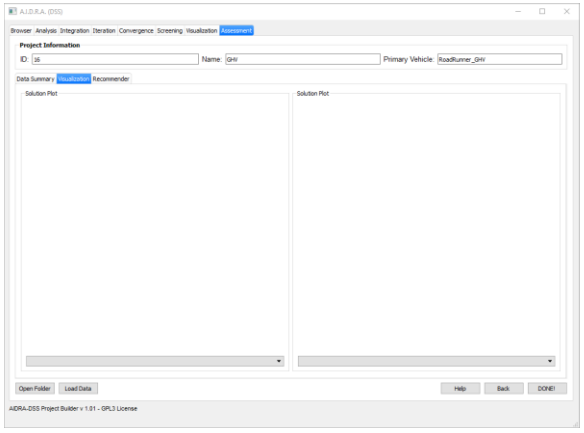
4.3.5.8.1 Data Summary

The Data Summary tab—Figure 4-40—is separated into two sections: Design Data Summary and Error Summary. The Design Data Summary is a section that contains a table that is populated by the primary design data generated by the code. The significant design variables and their values are shown. The Design Data Summary is a static display of the results for review by the user.

The Error Summary section is area in which principal design data is compared to a known vehicle's value. The percent difference between the known value and the output value are shown. The primary purpose of this section is to present an easy view for evaluating the accuracy of the system built. The user can select a specific vehicle for comparison by selecting it from the drop-down menu next to the "Change Comparison Vehicle" label. The current comparison vehicle name is displayed in the bottom left box at "Known Comparison Vehicle". The system will automatically select the nearest available vehicle available. If desired comparison vehicle is not available, then the user must add it to the database or perform the comparison in an outside environment.

4.3.5.8.2 Visualization

The Visualization tab, see Figure 4-41, displays the figures generated as previously specified in the Visualization page. The tab is separated into two sections; each section is a figure display area. Under each area is a separated drop-down menu. From the drop-down menus, the user can switch the figure displayed. The menu options are the figures found in the figure folder for the specific study. The user can change the default folder by selecting "Open Folder".



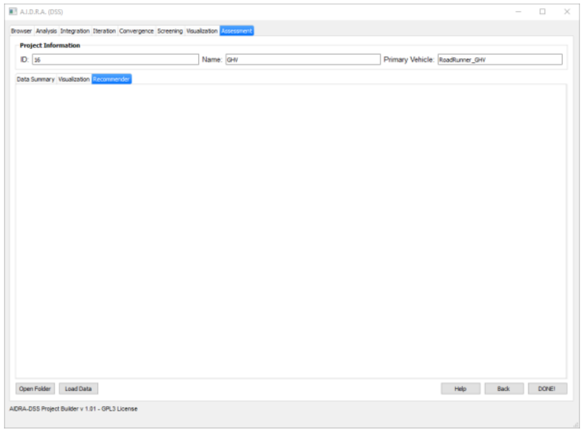
ID: 16 Name: GHV Primary Vehicle: RoadRunner_GHV

AIDRA-DSS Project Builder v 1.01 - GPL3 License

Figure 4-41 Assessment page—Visualization tab

4.3.5.8.3 Recommendations

The Recommendation tab—Figure 4-42—displays a set of design recommendations for the given project. This portion is underdevelopment and is to be a research and development area for another work. The purpose of this tab is to present the user with computer recommendations for the design problem. This can include the best design point, the effect of different hardware on the system, hardware combination recommendations, and more.



ID: 16 Name: GHV Primary Vehicle: RoadRunner_GHV

AIDRA-DSS Project Builder v 1.01 - GPL3 License

Figure 4-42 Assessment page—Recommendation tab

4.4 Back-End: Synthesis Assembler and Architecture

A general description of the back-end was given in section 4.1 *Description, Structure, and Core Components*, however, a more indepth discussion of the *Synthesis Assembler* comonent is necessary. The *Synthesis Assembler* is the element that, as the name implies, assembles the synthesis code. Given the problem's elements as defined during the operation of the *Project Builder*, such as the vehicle selections and decomposition, the processes, and the method selection, the *Synthesis Assembler* extracts the information from the systems databases and, using an assembly instruction algorithm, assembles the synthesis code with correct order of operation and initialization. The result is a unique and tailor-made synthesis code specific to the problem at hand. The *Synthesis Assembler's* output is a single aggregate file containing all necessary definitions and information required to execute the analysis.

4.4.1 Synthesis File Structure

The synthesis code itself, as generated, has a specific structure. Every synthesis architecture is assembled into a structure composing of specific algorithms, as necessary, which can be categorized by task. Each is identified and described below.

» Process Cost Function: a definition that is the primary call for the solver routine. The function contains a main analysis call and the objective function to be minimized. The primary output is the objective function(s) error in the correct form for the solver used.

» Solver Iteration and Call: a definition set that is the primary analysis driver. It contains the solver call and a routine for approximating the initial values or bounds for said solver. The solver itself is variable and dependent on the user. In the current environment, both a general nonlinear solver and an evolutionary global solver are used either individually or in tandem.

» Primary Disciplines Call: a definition containing the primary discipline function calls (aerodynamics, propulsion, etc.). They are called according to the order of operation set by the process selected. Interdisciplinary calls are handled appropriately as needed within the parent discipline method.

» Input Sheet Call: a definition that contains and defines the variables and any associated values of the input and outputs of each method.

» Mission and Hardware Definition: a definition that identifies the linkages and relevant information between the mission segments and the functional hardware per mission segment.

» Trade Study Setup: a definition that identifies the trade variables and values for the trade study. The related or affected vehicle, method, hardware, and discipline are identified as well.

» Data Process and Save: a set of definitions for preparing and saving the data to json database files.

» Variable Update Handling: a set of definitions that handle the duties of updating and concatenating the variable data.

» Hardware-Method Association: a definition that defines the linkages and constraints between the various hardware and the principal methods applicable per function and discipline.

» Multiple Method Resolve: a definition that, in the event a given hardware or discipline has multiple associated methods, identifies which method to use in a given situation and the necessary and/or current input values available to execute the identified correct method.

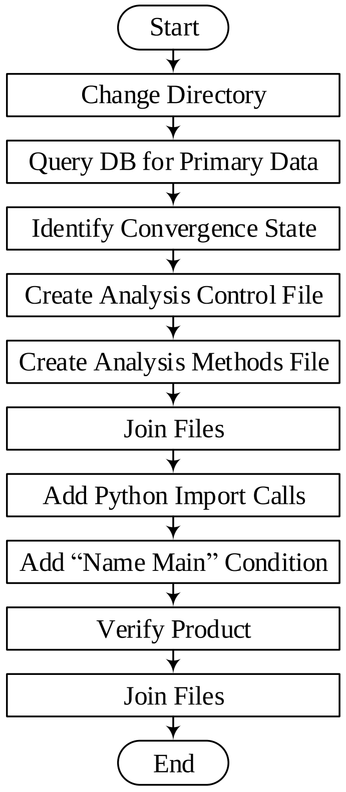
» Multiple Hardware/Method Variable Handling: code set that identifies how to handle data in the case that, for a given variable, the vehicle or hardware has multiple sources.

» Main Analysis Methods: a set of functions that are the engineering analysis methods that are called in the primary disciplines call block.

» Driver Code Block: this code block is a set of code (standard to all synthesis architectures) that initiates the program by calling the solver and iteration call definition (along with others). This is standardized code that exists after the *if __name__ == "__main__"* block of python code. This code handles the appropriate calls and setup of the problem dependent on the convergence and iteration case selected.

4.4.2 Synthesis File Generation Process

The code generation process to assemble the code is shown in Figure 4-43. The process is sequential. The process begins with the querying of the project database for the project variable definitions. These include the process(es), vehicle(s), and methods selected along with the method constraints, mission definition, and hardware-method-mission associations. Based on the process variable data, the *Process Library* is queried for the process information—primary disciplines, discipline order of operation, and objective functions. The input variables and values are extracted from the *Project Builder's* database; recall that the input and interdisciplinary variables were identified during the *Project Builder* operation and the user has entered the variable values. With the problem specific data available, the synthesis code is assembled.


Start
Change Directory
Query DB for Primary Data
Identify Convergence State
Create Analysis Control File
Create Analysis Methods File
Join Files
Add Python Import Calls
Add "Name Main" Condition
Verify Product
Join Files
End
Figure 4-43 Code assembly process

The code is assembled into the parts as described previously. A new code file is created. To it is added the principal analysis control definitions (Process Cost Function Solver, Solver Iteration and Call, Primary Discipline Call) in addition to the input values (Input Sheet Call) and trade study definition (Trade Study Setup). Additionally, the methods themselves are added. The methods are processed for trigger events for automated code insertion based on the dependent (interdisciplinary) method calls found within the source file. After processing, the updated methods are inserted into the file along with additional templated code control algorithms, method resolve algorithms and data processing, handling, and saving algorithms. On completion, the *Synthesis Assembler* has output a synthesis code that can be executed externally or internally of the DSS and is fully distributable with all uniquely necessary code included.

4.4.3 Synthesis File Generator Structure

The assembler code, the *Synthesis Assembler*, is a standard python script consisting of many functions. The functions can be categorized by application. The categories are summarized in Table 4-4. They, naturally, are similar to the synthesis code structure described previously, as they are responsible for generating the synthesis file.

Table 4-4 Principal function categories of the back-end's Synthesis Assembler

Category	**Description**
Utility Methods	Collection of supporting definitions not specific to any one condition
Directory Create	Definitions that identify root directory and create project subdirectory folder, as necessary.
File Control	Set of definitions for file name and path generation, and file copy to directory.
Import Inputs	Set of definitions for input sheet import and trade variable setup to proper form.
Data Extract	Set of definitions to query databases to extract primary data as selected and specified during GUI operation. Data is harnessed into useful form. Such data includes the process and methods selected, method limitations, mission definition, hardware-method-function-mission association, etc.
File Generator	Definition set to control synthesis code generation; calls to sub definitions for creation. Creates file in structure as described in 4.4.1 Synthesis File Structure.
Part Generator	Definition set that supports or executes specific subtasks within the file generation process or are code templates utilized in file generation. Various definitions generally fall within the code categories identified in 4.4.1 Synthesis File Structure.
Method Processing	A definition set for processing engineering methods for trigger events, import calls, and proper format. Definitions handle code injection for trigger events.

4.4.4 Selected Significant Algorithms

The *Synthesis Assembler* and resultant synthesis file comprise of many definitions. The *Synthesis Assembler* script is about 4000 lines, and the synthesis file is not limited to any length. There are several significant algorithms in both files. Many are shared in some form as one creates the other and the *Synthesis Assembler* is in part a library of templated code. Several significant definitions are selected for discussion and are summarized algorithmically. They are separated by location—synthesis generation (*Synthesis Assembler*) and synthesis / analysis file.

4.4.4.1 Synthesis Assembler

The overall approach to file generation according to the Synthesis Assembler is given in section 4.4.2 Synthesis File Generation Process. In this section, discussed specifically is the processing of the methods file for insertion into the assembled code. There are two points of discussion (1) method file processing in general and (2) trigger event processing.

4.4.4.1.1 Method File Processing

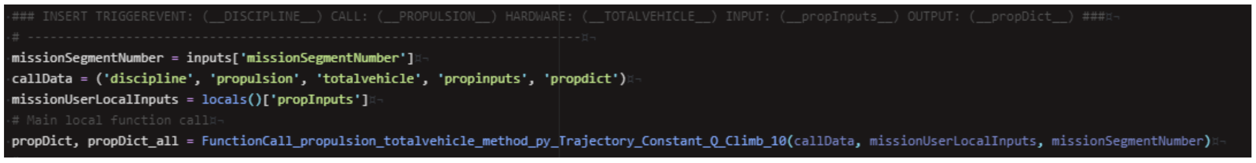
Figure 4-44 depicts the method processing process. The procedure begins with the identification of the methods required for the given vehicle's solution process. Each method is transcribed into a temporary methods collection file. During the transcribing process, each method's file is opened and read line by line. During the transcribing process, if a lines text satisfies a specific regex condition, the line is processed, and an event occurs. Two specific conditions searched for are the identification of a trigger event and the identification of an import call (python element). Other secondary processing occurs but it is not critical to this discussion as it reshapes the file into a desired layout and formatting style. If the line contains an import call, the import call is saved to a list that, on completion of all method processing, is filtered for unique imports, is joined with other required import calls specified elsewhere, and is inserted at the head of the main synthesis file. If a trigger event is identified, the event call is decomposed, processed, and the appropriate code is added to the method in the temporary file. The event is discussed in detail below. After all methods are processed, the temporary methods collection file is appended to the main synthesis file.

```
Algorithm Method File Process
Dependents: $f(mainAnalysis)$
Output: temporary methods file, appending to main synthesis file
1:  $for\ process$
2:      $methods\ \leftarrow get\ methods\ list$
3:          $for\ meth\ in\ methods$
4:              $line\ \leftarrow get\ method\ line$
6:              $if\ line\ is\ import\ line, store\ import\ call$
7:              $if\ line\ is\ definition\ line, store\ definition\ information,$ write to temp. methods file
8:              $if\ line\ is\ trigger\ event$
9:                  $triggerEventInfo\ \leftarrow process\ trigger\ event$
10:                 $Insert\ trigger\ evnet\ code\ according\ to\ triggerEventInfo\ into\ temp\ file$
11:             $if\ no\ special\ condition, write\ line\ to\ temp.file$
12:             $go\ to\ next\ line$
13: $Append\ temp.methods\ file\ to\ main\ synthesis\ file$
14: $Append\ unique\ import\ call\ list\ to\ head\ of\ synthesis\ file$
```

Figure 4-44 Method File Processing function process and layout

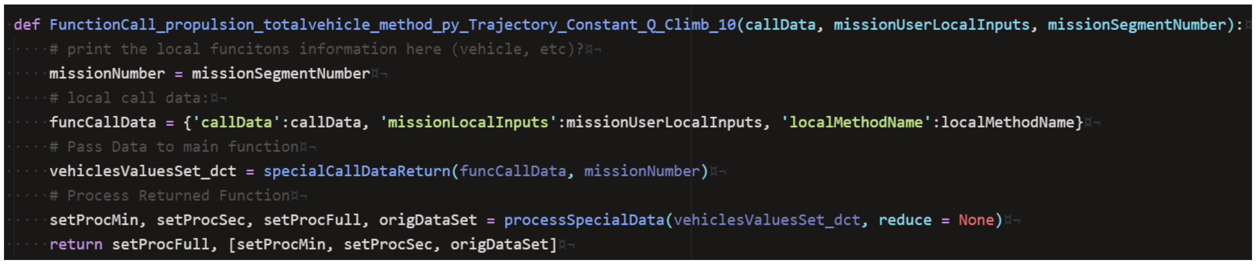
4.4.4.1.2 Trigger Event Processing

Each method file, during transcribing, is processed for trigger events. A trigger event is a line instructing for a certain code call to be inserted dynamically based on user selections. If a line contains a trigger event call, the event call is decomposed into its core components—event type, call discipline/function, call hardware, call local inputs, and call local output name. The type indicates if it is a function or discipline call. The call indicates the specific discipline or function of interest. The hardware term specifies whether the call applies to the total vehicle or a specific subsystem, and the inputs/outputs are the names within the source code that are required to access the output or input data for the call execution itself and the call results. Based on the event specifications, the correct code is inserted to call the correct code specified by the event and to create the correct function that is called. Figure 4-45 and Figure 4-46 are examples of the inserted event call code and the generated function that is called. Additionally, the generated function is dependent on a method resolution and variable processing function set, which is discussed in detail in the next section. The new functions and event call code generated per trigger are transcribed into the temporary method file as discussed previously.



```
### INSERT TRIGGEREVENT: (__DISCIPLINE__) CALL: (__PROPULSION__) HARDWARE: (__TOTALVEHICLE__) INPUT: (__propInputs__) OUTPUT: (__propDict__) ###
# ------------------------------------------------------------------------
missionSegmentNumber = inputs['missionSegmentNumber']
callData = ('discipline', 'propulsion', 'totalvehicle', 'propinputs', 'propdict')
missionUserLocalInputs = locals()['propInputs']
# Main local function call
propDict, propDict_all = FunctionCall_propulsion_totalvehicle_method_py_Trajectory_Constant_Q_Climb_10(callData, missionUserLocalInputs, missionSegmentNumber)
```

Figure 4-45 Example trigger event call and local inserted code



```
def FunctionCall_propulsion_totalvehicle_method_py_Trajectory_Constant_Q_Climb_10(callData, missionUserLocalInputs, missionSegmentNumber):
    # print the local funcitons information here (vehicle, etc)?
    missionNumber = missionSegmentNumber
    # local call data:
    funcCallData = {'callData':callData, 'missionLocalInputs':missionUserLocalInputs, 'localMethodName':localMethodName}
    # Pass Data to main function
    vehiclesValuesSet_dct = specialCallDataReturn(funcCallData, missionNumber)
    # Process Returned Function
    setProcMin, setProcSec, setProcFull, origDataSet = processSpecialData(vehiclesValuesSet_dct, reduce = None)
    return setProcFull, [setProcMin, setProcSec, origDataSet]
```

Figure 4-46 Example trigger event inserted function

4.4.4.2 Synthesis File

Section 4.4.1Synthesis File Structure summarized the principal function groups and general file structure. In this section, the principal algorithms are addressed in more detail. Specifically considered are the groups: Process Cost Function, Solver Iteration and Call, Primary Disciplines Call, Multiple Method Resolve, and Multiple Hardware/Method Variable Handling. These algorithms effectively form the spine of the analysis and solution finding process.

4.4.4.2.1 Process Cost Function

The process cost function is the function targeted by the numerical solver; it computes and returns the value of the process's objective functions given values for the independent process variables as shown in Figure 4-47. The function calls the main engineering analysis function, computes the objective functions' values, and returns the values. These values are used by the numerical solver to converge to correct independent variable values.

```
Algorithm: Process Cost Function (Solver Function)
Dependents: $f(mainAnalysis)$
Output: $e$
1:  $for\ var\ in\ indVarSet$
2:      $inputs[var] \leftarrow update\ var\ val\ given\ x0_{init}$
3:  $results = mainAnalysis(inputs)$
4:  $for\ var\ in\ (indVarSet, depVarSet)$
5:      $var \leftarrow extract\ var\ val\ from\ resutls[var]$
6:  $for\ objFunc\ in\ objectiveFunctionSet$
7:      $e_i \leftarrow objFunc(indVar, depVar)$
8:  return $e$
```

Figure 4-47 Solver function process and layout

4.4.4.2.2 Solver Iteration and Call

The Solver Iteration and Call consists of a function where in the Process Cost Function is called as necessary until problem resolution. Figure 4-48 illustrates the process. The process occurs per design or mission variable trade as shown by the first for loop. For each trade condition, the solver is executed for n number of attempts ($n_{attempt}$). With each attempt, a different initial condition is utilized. It was found, for the hypersonic case study addressed in Chapter 5 Verification and Application, that the success of the solver—convergence—could depend greatly on the initial condition used.

Several initial guess approaches were implemented and are usable. Approaches include a constant or random growth factor applied to the previous converged solution (a Monte Carlo type approach), an evolutionary algorithm, and a simple grid search for an appropriate initial guess. However, the standard approach is to begin each trade, with the previous converged state's values as the starting point of the initial guess for the independent process variable. The value is queried from the result database using the *initCondApprox* function. The independent variable values are adjusted according to the initial guess approach being utilized. Ultimately, with any approach used, the result is an initial guess that is used by a nonlinear solver to solve for the independent variable.

Upon satisfactory solver completion or expiration due to reaching the attempt limit and exhausting initial guess approaches, the result is either a converged or a not converged event. If convergence does occur, the indent variables' values solved for are passed into the main analysis and the execution results are returned. The results are processed for form and are saved to a Jason database file. If convergence does not occur, the final iteration result and solver state is saved for record keeping. The process repeats for the next trade state.

```
Algorithm Solver Iteration and Call
Dependents: f(initCondApprox, runEvolSolver, solverFunc, mainAnalysis, jsonifier, jsonSave)
Output: main analysis data and summary
1:   $for\ i\ in\ n_{trades}$ do
2:       $for\ j\ in\ n_{attempt}$ do
3:           $if\ i = 1$
4:               $x0_{init} = X0_{guess}$
5:           else
6:               $x_{init} \leftarrow initCondApprox()$
7:               $for\ var\ in\ x_{init}$
8:                   $x0_{init} \leftarrow var * initGrowthVar$
9:           $result \leftarrow solver(solverFunc, x0_{init})$
10:          $if\ evalSolve = true\ and\ result.conv = False$
11:              $result \leftarrow runEvolvSolver(solverFunc)$
12:              $x0_{init} = result$
13:              $result \leftarrow solve(solverFunc, x0_{init})$
14:          $if\ result.conv == True$
15:              $data = mainAnalysis()$
16:          $conditionedData \leftarrow jsonifier(data)$
17:          $jsonFile \leftarrow jsonSave(conditionedData)$
```

Figure 4-48 Synthesis solver iteration process

4.4.4.2.3 Primary Disciplines Call

The Primary Disciplines Call, or main analysis, is rather strait forward. Figure 4-49 shows algorithmically the approach. The function is responsible for executing the engineering analysis as prescribed by the process. The analysis function is uniquely generated for each synthesis architecture created according to the process, method, and vehicle/hardware selections made in *Project Builder*. However, in most cases the main analysis is a linear sequence of discipline calls. The analysis for most cases follows that depicted—a standard analysis process for a single vehicle. For each vehicle's primary process and the disciplines required the analysis executes per hardware as required. The result is a dataset containing the input and output of each analysis method executed.

```
Algorithm Primary Disciplines Call
Dependents: $f(analysis\ methods)$
Output: $globalData$
1:  $for\ discipline\ (disc)\ in\ primary\ process$
2:      $for\ hardware\ (hard)\ in\ vehicle\ (veh)$
3:          $if\ vehicle\ hardware\ associated\ to\ discipline$
4:              $for\ method\ (meth)\ in\ disc\ per\ veh\ hard$
5:                  $inputs \leftarrow get\ inputs,\ f(veh, hard, disc, meth)$
6:                  $data \leftarrow method(inputs)$
7:                  $globalData \leftarrow update\ with\ local\ data$
8:  return $globalData$
```

Figure 4-49 Main Analysis function process and layout

4.4.4.2.4 Multiple Method Resolve and Data Processing

There are situations in which there is more than one method per hardware or function. To determine the appropriate method and necessary inputs, a method resolve routine is necessary. The processes to determine, execute, and process the results of a multiple method or multiple hardware case is illustrated in Figure 4-50 through Figure 4-52. These processes correlate to three functions. The three functions are referred to as the Method Resolve, Special Call Data Return, and Process Special Data. These methods are contained within the Multiple Method Resolve, and Multiple Hardware/Method Variable Handling groups discussed previously. Each is discussed next.

The Method Resolve implements a process to determine the appropriate method to execute given multiple methods associated to a hardware for a given discipline or function. Upon execution, the result is the correct method for the given situation. The correct method is determined by the method's type (driver method or not), the number of methods, and the methods' control variable's value versus the variable's current value. The process is depicted below. The function returns the determined method's name and function handler. The input (methodDataSet) is a list of relevant methods, their data, and their handles. In the greater scheme, the Method Resolve is called within the Special Call Data Return function, which is discussed next.

```
Algorithm Method Resolve
Dependents:
Input: methodDataSet
Output: $methodUsePointer, methodUseName$
1:  $names, pointers, contVar, contVarVal \leftarrow decompose\ methodDataSet$
2:  $if\ length(methods) == 1\ and\ conVar\ != None\ and\ conVar\ != Driver$
3:      $methodUse = methods[0]$
4:  $elif\ driver\ method\ in\ methods$
5:      $if\ count(drivermethod) > 1$
6:          $raise\ exception$ "$more\ than\ 1\ driver\ method$"
7:      $else$
8:          $i \leftarrow control\ method\ index$
9:          $methodUse = methods[i]$
10: $elif\ length(methods) > 1\ and\ driverMethod\ not\ in\ methods$
11:     $while\ j < length(methods)$
12:         $if\ j == length(methods) - 1$
13:             $if\ currentControlVarVal \leq controlVarVal$
14:                 $methodUse = methods[j]$
15:             $elif\ currentControlVarVal > controlVarVal$
16:                 $raise\ exception$: "$Current\ value\ exceeds\ methods\ ranges$"
17:                 $methodUse = None$
18:             $else$
19:                 $raise\ exception$: "$No\ method\ meets\ constraints$
20:                 $methodUse = None$
21:         $elif\ j == 0$
22:             $if\ currentControlVarVal \leq controlVarVal \rightarrow methodUse = methods[j]$
23:             $else\ j += 1$
24:         $else$
25:             $if\ controlVarVal[j-1] < currentControlVarVal \leq controlVarVal$
26:                 $methodUse = methods[j]$
27:             $else$
28:                 $j += 1$
29: return $methodUsePointer, methodUseName$
```

Figure 4-50 Method Resolve function process and layout

Special Call Data Return

The Special Call Data Return is the function set that identifies and executes the method given a trigger event. The functions general structural procedure is shown in Figure 4-51. The Method Resolve is called in this function. The principal output is the data generated from the resolved method. The inputs are the trigger event data and the current variable data set at parent method execution. The process executes based on the trigger events data: event type, event option call, hardware call, specified local inputs, and specified output name. Per vehicle and per functional hardware as prescribed by the call option and hardware, the procedure identifies the appropriate method for the given state of the methods' constraining variable and method types. The global inputs, for the identified correct method, are updated with the specified local input variable values. The updated values are inserted into the method, the method executes, and the results are returned.

```
Algorithm 2 Special Call Data Return
Dependents: $f(methodResolve)$
Input: trigger event information, inputData
Output: $data$
1:  $type, opt, hard, input, output \leftarrow triggerEventData$
2:  $disc, func \leftarrow func2discMap(opt), disc2funcMap(opt)$
3:  $for\ veh\ in\ vehicleSet$
4:    $hardwareSet \leftarrow\ missionFunc2hardwareMap(mission, func)$
5:    $for\ hard\ in\ hardwareSet$
6:      $methodSetData \leftarrow allMethsData[veh][hard][disc]$
7:      $where\ methodSetData\ of\ type\ [name, pointer, controlVar, controlVarVal]$
8:      $if\ currentMethod\ in\ methodSetData \rightarrow pop\ method$
9:      $locMethInputs \leftarrow get\ current\ method\ input\ values\ from$ inputData
10:     $locInputs \leftarrow get\ updated\ variable\ values\ from\ trigger\ event\ data$
11:     $inputSet \leftarrow update\ locMethInputs\ with\ locInputs$
12:     $methodPointer, methodName \leftarrow call\ methodResolve(methodSetData)$
13:     $data \leftarrow call\ methodPointer(inputSet[methodName])$
14:     $return\ data$
```

Figure 4-51 Special Call Data Return function process

Process Special Data

Process Special Data processes the data returned by Special Call Data Return prior to passing the data back to the parent function that had initiated this cycle of events. The data for each function source as related to the hardware and vehicle is returned as a set of data identifying the variable values by hardware, vehicle, and total vehicle. In this way, all data states are available to the parent analysis file for use. The necessity of a data processing event arises due to the potential for multiplicity of vehicle or hardware and the necessity of situational awareness of the variable for proper total variable value determination, for not all variable's total is the simple sum of the individual variable's values ($\sum T = T_{total}\ but \sum I_{sp} \neq I_{sp_{total}}$). The simplest case is the single vehicle and single hardware. In this case, the minimal condition (the single hardware's functional effect) is the total condition (the total vehicles functional effect on that variable). Figure 4-52 shows the process of the Process Special Data function. Each variable is treated according to its type—simple average, simple sum, weighted average, or special case. The type and associated handling rule must be identified for each variable.

```
Algorithm Process Special Data
Dependents: $f(mainAnalysis)$
Output: processedData, originalData
1:  $var\ data\ set \leftarrow gather\ similar\ varialbes\ per\ hardware\ data\ set\ per\ vehicle$
2:  $for\ each\ vehicle\ harware\ set\ and\ var\ in\ vehicle\ hardware\ set\ do$
3:      $check\ var\ for\ special\ conditions\ (weighted\ average, sum, etc.)$
4:      $process\ variable\ according\ to\ special\ condition$ and store in dictionary
5:  $for\ each\ veh\ and\ hardware, process\ var\ subsets\ into\ total\ system, local\ system, and\ local\ hardware\ sets$
6:  return processedData, originalData
```

Figure 4-52 Process Special Data function process and layout

4.5 Chapter Summary

4.5.1 General Summary

In this chapter, a presentation of the principal research objective—development of a decision support framework for the CD phase—was given. The system is referred to as AIDRA-DSS. The overall system was discussed. The general file structures, both front-end and back-end, were presented. An in-depth presentation of the front-end and back-end was given. Identified and discussed were the support libraries (*Reference Library, Method Library, Process Library, and Vehicle Library*) as well as the primary DSS environment, the *Project Builder*, and principal components and approach of synthesis code generation through the *Synthesis Assembler*. The *Project Builder* is the principal element that the libraries support. A primary deliverable of the system is a synthesis architecture.

AIDRA-DSS's primary directive is the assembly, documentation, and standardization of sizing architecture generation. The goal is transparency and accountability in the development of sizing toolsets. AIDRA-DSS is a semi-automated tool-of-tools. Through a code assembly platform, given the user's specifications, the system assembles base components into a functional architecture for a given problem. Its primary purpose is the assembly of methods into a sizing toolset to better help in decision-making. This is model-based engineering, with a capacity to model any hardware's effect and contribution to any discipline within the design process, to assist in design evaluation and decision-making. Each sizing toolset is specifically generated to solve the problem at hand.

With the system discussed, the next objective is to demonstrate proper system functionality and potential. The next chapter will demonstrate the functionality of the system by presenting a case study in both single point verification and multi-point trade study.

4.5.2 Contribution Statement

» Developed and presented a unique generic synthesis assembly tool founded on principles of a vehicle-of-vehicle concept and problem definition by vehicle-hardware statement.

Chapter 5 Verification and Application

Having specified the system concept and implementation, the next requirement is to demonstrate functionality and application. The research conducted and elaborated on in this document is of course of two parts. There is the tool developed and there is the output of the execution of the tool and its utilization. Functionally, the system's outputs are both the synthesis code generated and the output of the synthesis code. The correctness of the synthesis code results depends directly on the correctness of the code assembly.

Proper system code assembly is verified by manual inspection of code assembly and, more significantly, is mostly inferred from correct output upon assembled synthesis system execution. As of now, there is no automated or computerized intelligent verification of proper code assembly aside from the assembly code executing without error. For the purposes of the following discussion, the synthesis codes were manually checked for proper assembly. On inspection, all codes were assembled as algorithmically specified. With this understanding, the condition of code assembly is considered properly executed. Therefore, the criterion of manual verification of synthesis code assembly based on user GUI selections is found to be complete. The remainder of this chapter evaluates the code assembly by consideration of correctness of code output and demonstrates system application.

Inference of correct assembly and demonstration of system application is accomplished through the execution of a verification study and a trade study. The verification study and the trade study are the subject addressed in this chapter. The problem setup and results of the case studies selected are systematically presented in the following sections. Addressed first is a general description of the problem and solution approach. The verification and case study are on the topic of hypersonic reusable vehicle demonstrators.

5.1 Problem Statement

The execution study is separated into two parts—verification and application demonstration through a trade exploration. The verification step is critical in establishing the correctness of GUI to synthesis code operation and execution. The trade exploration takes the verification step a step further. It serves as not only a system test and verification for the multipoint system execution case, but is also a demonstration of the systems utility to a larger problem that is relevant. For the purposes of this research, the objective is to demonstrate system operability for the single-vehicle and the single process case.

5.1.1 System Verification

The verification and validation is presented in two parts. The approach can be broken into the consideration of the single point and the multi-point cases. Both are conducted to verify proper system execution. However, for the purposes of the discussion of verification, the single point case is considered the primary focus here as it demonstrates the key system component of individual vehicle execution on which any multi-point case is based upon. That is, the multi-point case is an expansion of the single case (repeated execution of the single point case for a breath of varied input values).

Single point verification occurs by executing the system for a known control vehicle and comparing the resulting design output to known vehicle design variables' values (legacy verification data). System execution correctness is inferred based on the output versus known variable comparison. Several vehicles are selected as representative cases. They are selected such that the AIDRA-DSS system must execute several scenarios in which the process components must be varied. Control vehicles are both, real world production or test vehicles, and concept case study vehicles from other documented project (paper) studies.

The problem of verification is approached through a systematic buildup. To establish verification, there is the establishing of the vehicle selections, the vehicles' missions, the vehicles' synthesis process, and the synthesis methods that culminates in the result evaluation and, subsequently, satisfactory verification establishment. Each step is addressed in the following chapter sections.

5.1.2 Trade Study

The trade study is an expansion of the single point verification case. The trade study entails the variation of assumptions or input variables to arrive at many solutions that are presented to the user as a space of solutions. The solution space is there to assist in the evaluation of the solutions' responses to variations in design variables. Figure 5-3 illustrates simplistically a trade study as multiple cases of a single point design case, Figure 5-2, with variation in a design variable.

The trade study serves two purposes. First, in terms of system operability, it demonstrates the iterative multi-point analysis functionality. Second, it presents an opportunity to study a problem through the exploration of a solution space. In the multi-point case, it is possible to evaluate selections in vehicle concept, configuration, hardware, and operational conditions. In the case presented in this chapter, the single point verification vehicles are used as baseline concepts to explore a hypersonic solution space.

The trade study identifies, synthesizes, and evaluates a representative baseline set of hypersonic test vehicle concepts in terms of the consideration of carrier vehicle constraints. Figure 5-1 illustrates considerations for carrier vehicle constraints through the illustration of the X-24C and B-52 combination. Baseline configurations' solution topographies are identified through the evaluation of various vehicle operational requirements; as such, a trade matrix is identified. The multi-disciplinary study results are constrained with carrier payload mass and geometry limitations. The multi-disciplinary results provide physical insights into near-term hypersonic test vehicle design variable relation to the carrier vehicle requirements.

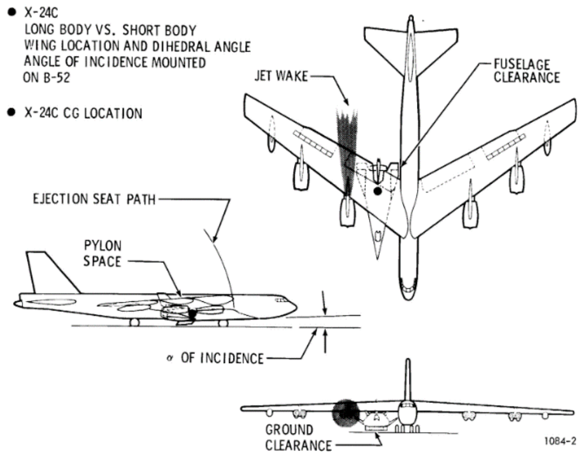


Figure 5-1 Illustration of X-24C test vehicle and B-52 carrier vehicle constraints considerations [146]

The trade study case is similarly built up as the single point verification case. Due to significant overlap between the two cases, both are addressed concurrently. As such, the multi-point case is likewise documented through a systematic buildup addressing the establishment of the vehicle selections, the vehicles' missions, the vehicles' synthesis process, the synthesis methods, and the additional identification of a trade matrix, which finalizes in result presentation and discussion.

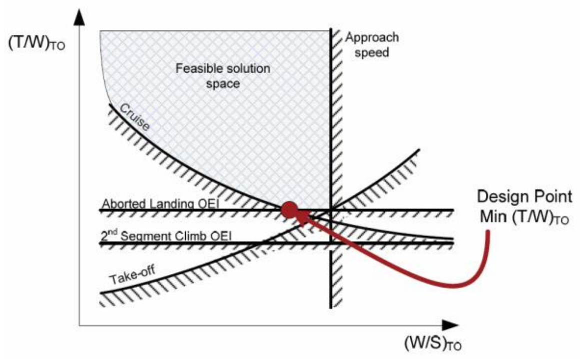


Figure 5-2 Example of the classical performance matching diagram design point[85]

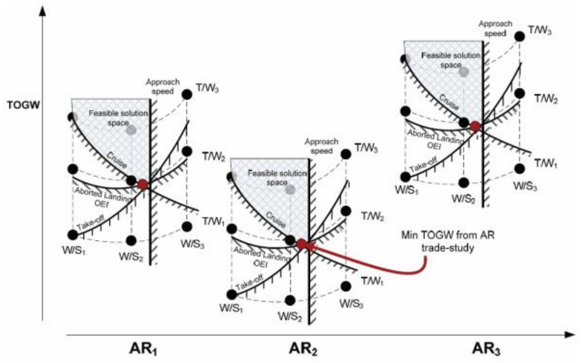


Figure 5-3 Trade study illustration visualized by a set of performance matching diagram[85]

5.2 Vehicle Selection

The vehicle concept and configurations selected for study are categorized by verification case and multi-point trade study case. The vehicle selection of each is considered.

5.2.1 Verification Case

The verification vehicles include a mix of concept vehicles and flown vehicles. The vehicles are hypersonic test vehicles. The hypersonic test vehicles selected are the USAF AFRL Road Runner Generic Hypersonic Vehicle (GHV) [147] and X-51A [148]. Each vehicle represents a different concept, blended-body versus all-body respectively. A range of vehicle concepts has been selected in order to ensure that the methods for each discipline will have to change, thereby testing for proper code generation. Note that the vehicles selected are high-speed (hypersonic). Low speed (subsonic) vehicles could equally have been used. However, hypersonic systems are a current research and development area of relevance to many governments and agencies. As such, the topic serves as a relevant case study demonstration. Additionally, the trade study case is for hypersonic systems, so the verification by hypersonic systems supports the trade study cases as well. Under this condition, the system is tested for proper operation and synthesis code generation. Table 5-1 identifies each vehicle and summarizes each vehicle's general classification.

Table 5-1 Verification study vehicles

Name	Class	Organization	Mach Range	Summary
GHV	Hypersonic Test	*Air Force*	$M = 6$	Blended-Body waverider with 3D inlet and nozzle air-breathing scramjet powered cruise vehicle concept
X-51A	Hypersonic Test	*DARPA* *Air Force*	$M = 6 +$	All-body 2D scramjet powered hypersonic test vehicle

With the vehicles identified, the geometric and sizing variables for each vehicle are tabulated in Table 5-2. Additional discussion of each vehicle can be found in Appendix A Case Studies Expanded.

Table 5-2 Vehicle sizing and general parameter values [147, 148]

	Units	GHV(1x)	GHV(2x)	GHV(3x)	GHV(4x)	GHV(5x)	X-51A
τ	-	0.0735	0.0698	0.0678	0.0674	0.0657	0.2075*
V_{tot}	m^3	0.563	1.494	2.713	4.111	5.642	0.705*
S_{pln}	m^2	3.888	7.705	11.699	15.496	19.457	2.26*
l							
Overall	m	4.468	6.319	7.739	8.936	9.991	4.267
Fuselage	m	4.313	6.100	7.471	8.627	9.645	-
d	m	0.479	0.677	0.829	0.957	1.070	0.584
b	m	1.488	2.104	2.577	2.975	3.327	0.702
W_{GTOW}	N	5430	11586	19386	27894	36456	6690
W_{fuel}	N	1099**	3493	6658	10331	10331	1241**
W/S	N/m^2	1397	1504	1657	1800	1874	2960*

* estimate
** usable fuel plus approximate non usable (launch weight less cruiser operating weight)

The verification vehicles are illustrated below. The GHV and X-51 are shown in Figure 5-4, Figure 5-5 respectively.

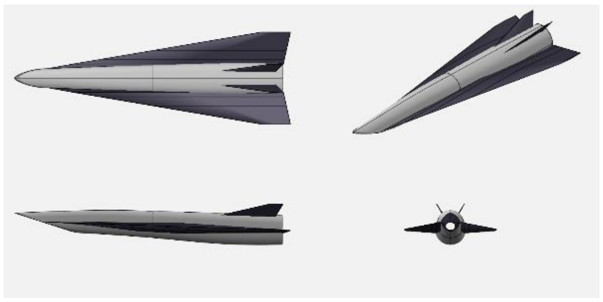


Figure 5-4 GHV

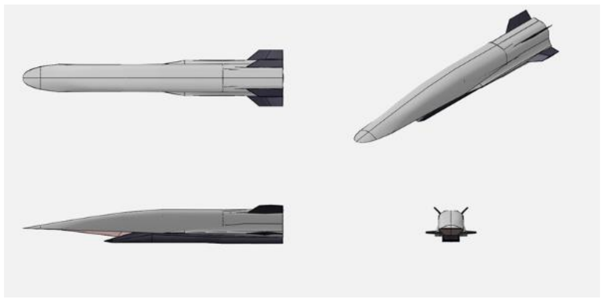
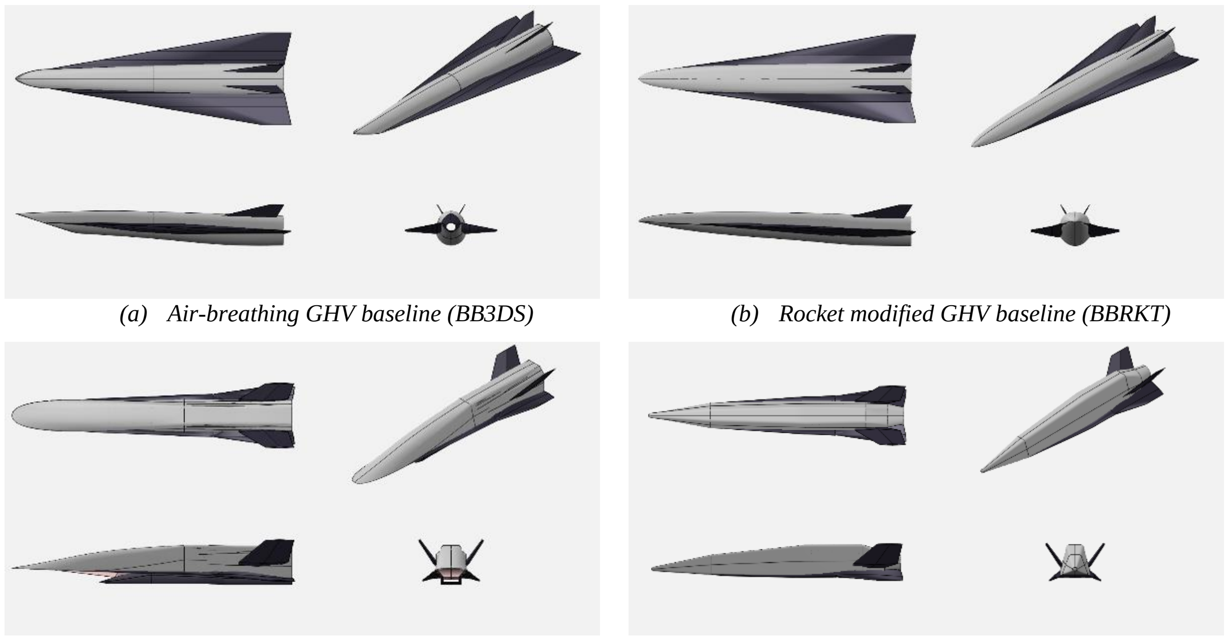


Figure 5-5 X-51.

5.2.2 Trade Study Case

The vehicles from the verification case form a baseline vehicle set for the trade study case. As in the verification case, the all-body (AB) and the blended-body (BB) concept types are represented. Furthermore, the hypersonic test vehicle case is expanded to consider both airbreathing and non-airbreathing systems. The GHV and X-51 are used both in their initial airbreathing concept and in a modified non-airbreathing concept state. Figure 5-6 visualizes each of the vehicle concepts. The blended-body concept is represented by the GHV vehicle class concept. The X-51 concept represents the all-body class. Note that the all-body's principle lift generating source is its own body and, as such, is also referred to as a lifting-body (LB).

The GHV concept is an air-breathing blended-body vehicle. To address the rocket powered blended-body concept (BBRKT), the GHV vehicle is transformed into an enclosed fuselage rocket concept, Figure 5-6(b). The vehicles fuselage intake is closed off and a rocket system added. The X-51 concept is the baseline for the AB airbreathing concept (AB2DS)—Figure 5-6(c). A rocket class AB baseline (ABRKT)—Figure 5-6(d)—is modelled after the FDL-7 and McDonald Douglas Model 176 and MRS, which predate the X-51 but share many similarities in configuration and concept. The X-51's configuration's outer mold line is very similar to the FDL-7's and Model 176's configuration, but with the addition of a spatula nose and an underslung 2D scramjet in place of the rocket propulsion system. Additionally, for low-speed landing, the AB concepts have an internal swing-wing included (historically included in the FDL-7, Model 176, and MRS as well). All systems concepts were originally designed for high Mach number operations. Note that the objective herein is not to drive to an optimal vehicle configuration, but rather to realize a general solution space. The vehicle concept perturbations are a representative spectrum incorporating both near-term and mid-term propulsion systems.



(a) Air-breathing GHV baseline (BB3DS) (b) Rocket modified GHV baseline (BBRKT)

(c) Air-breathing X-51 baseline (AB2DS) (d) Rocket-powered Model-176 baseline (ABRKT)

Figure 5-6 Multi-point trade study baseline vehicles.

5.3 Processes Definition

In this section, the synthesis process implemented for each case study is discussed. Each case study follows the same general process. The process implemented does not very and its variation should not be misconstrued with the variation in methods selected from case-to-case. A process coordinates the methods, and the methods can be variable while existing within the operation of the same process. The methods are independent of the process and are discussed in a later section. Presented in this section are the multi-disciplinary synthesis process, the convergence processes, and the process to solution space formation.

5.3.1 Multi-Disciplinary Synthesis Process

The multi-disciplinary synthesis core process is shared between both the single-point and multi-point case studies. Figure 5-7 illustrates the process. The dash-dot area sections the synthesis core process. Both studies share this convergent process. It is discussed in detail below. The multi-point process utilizes the same core synthesis process; however, appended to it is an iterative feedback loop as indicated. In the multi-point case, the single point case is executed repeatedly to identify a set of solutions; it is a trade study. In this process, the single point case is repeated with different configurations, concepts, hardware, or operational conditions. All conditions of the single point case's processes remain; that is, the disciplinary execution, order of operation, and convergence approach are maintained.

The synthesis process for all case studies implemented is converging. Convergence is synonymous with the phrase: "closing the design." A converging process is one in which some objective function or functions are satisfied in an iterative manner. Through this process, some design variables value is searched for until a predetermined condition (objective function) is met. A non-converging process is one in which no process objective function is met. In such a condition, the design is not iterated to satisfy a predetermined design condition. In the non-converging process, the resulting vehicle solution point would be considered not closed. The processes used here are converging; therefore, all design points indicated herein are closed designs.

5.3.2 Convergence Process Description

The process is a series of steps. There are two primary parts: the disciplinary analysis and the convergence loop. The disciplinary analysis exists within the convergence loop as indicated in Figure 5-7. The convergence loop contains the analysis block and, in an iterative process of analysis block execution, seeks convergence criteria satisfaction through analysis input variable variation. The convergence iteration variables are planform area (S_{pln}) and wing loading (W/S_{pln}). The study sizing methodology is a weight and volume-based convergence process. The process employed here considers the total vehicle volume required given the weight estimate. The approach is adapted from references [85, 86].

The disciplinary analysis begins with the assumption of a baseline vehicle and mission profile. A key geometric parameter—the vehicle volume coefficient ($\tau = V_{tot}/S_{pln}^{1.5}$)—is held constant for each convergence cycle. (The variation of τ allows for a volumetric scaling of the vehicle rather than a simple pictorial scaling, which is more appropriate for hypersonic vehicles.) It is an input into the geometry method and directly defines the other geometric parameters given the vehicle configuration. With a geometric definition in place, the discipline specific analysis modules execute. They are executed in the following order: aerodynamics, propulsion, trajectory, and finally weight and volume. The aerodynamic and propulsion modules can be executed either: (1) in the sequential series to generate aerodynamic and propulsion lookup maps for the vehicle at different operating conditions, or (2) they can be called directly within the trajectory methods. In the case studies executed here, the aerodynamic and propulsion disciplines are called directly in the trajectory methods. The trajectory module utilizes the vehicle's aerodynamic and propulsion data in the analysis of the vehicle's trajectory. From the trajectory analysis, the vehicle's performance parameters are determined, including the required weight ratio along the flight path. The weight and volume module updates the weight and volume of the vehicle based on the trajectory module's output. On completion of a sequence of disciplinary module-based analysis execution, the instance of analysis is complete. However, the overall vehicle has not necessarily converged.

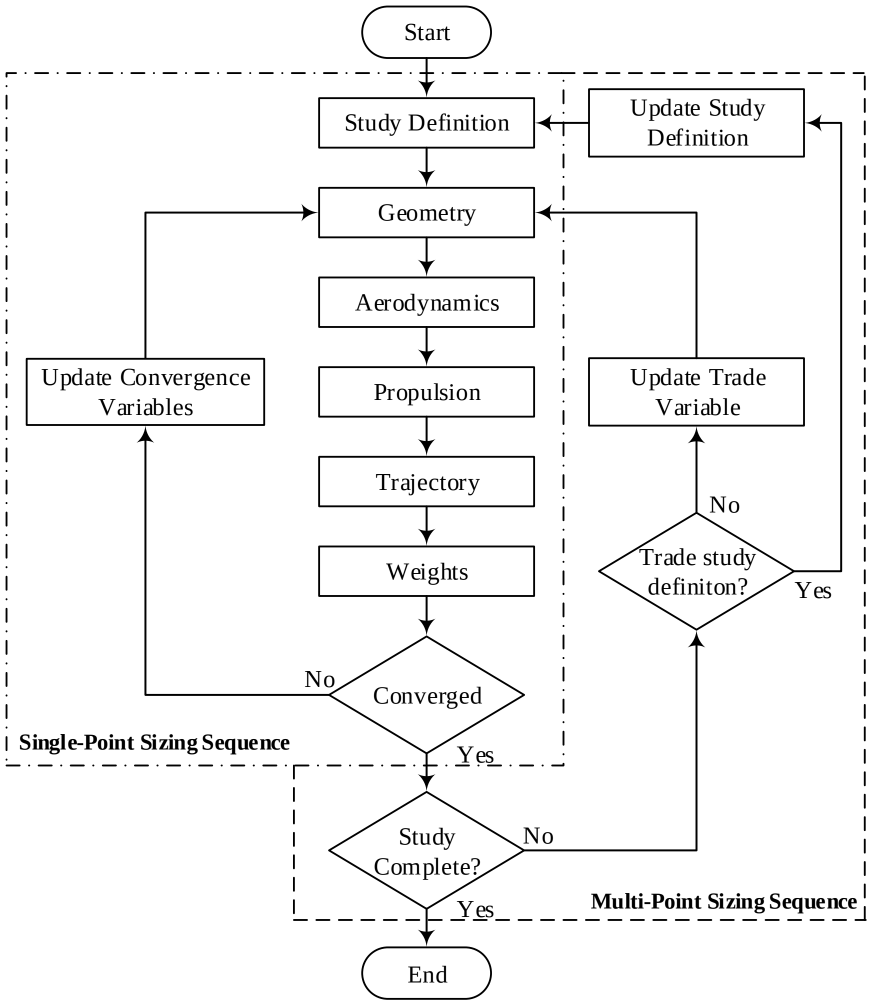


Figure 5-7 Sizing process, both single-point and multi-point sequence

After the analysis block's execution, the objective functions are tested for solution convergence. Two objective functions are minimized simultaneously, see equations (5.1) and (5.2). The first objective function, Eq. (5.1), is a function of operating weight empty by weight analysis (OWE_w) and operating weight empty by volume analysis (OWE_V). This function closes the vehicle's weight and volume requirement simultaneously. The second objective function, Eq. (5.2), is a function of wing loading (W/S_{pln}), planform area (S_{pln}), and takeoff gross weight ($TOGW$). In this function, the TOGW and planform area closes through an iteration of the wing loading. The convergence process is complete when each cost function equates to zero. In other words, the vehicle is said to be converged when its solution point is mathematically acceptable because weight and volume converge with $OWE_W = OWE_V$ and $\left(W/S_{pln}\right) = TOGW/S_{pln}$. If the objective functions are not satisfied, planform area and wing loading are iterated, and the sequence repeats until both cost functions are minimized simultaneously. For further discussion of this approach and the metrics used in the equations see Czysz [149], Coleman [85], and Gonzalez [86].

Objective Function 1:

$$OWE_V - OWE_W = 0 \tag{5.1}$$

Objective Function 2:

$$\left(\frac{W}{S}\right)_{guess} - \frac{TOGW}{S_{pln}} = 0 \tag{5.2}$$

Weight Budget:

$$OWE_W = OEW + W_{pay} + W_{crw} \tag{5.3}$$

$$OEW = \frac{I_{str} \cdot K_w \cdot S_{pln} + C_{sys} + W_{cprv} + T/W \cdot WR/E_{TW} \cdot \left(W_{pay} + W_{crw}\right)}{1/(1+\mu_a) - f_{sys} - T/W \cdot WR/E_{TW}} \tag{5.4}$$

Volume Budget:

$$OWE_V = \tau \cdot \frac{S_{pln}^{1.5} \cdot (1 - k_{vv} - k_{vs}) - \left(V_{pcrw} - k_{crw}\right) \cdot N_{crw} - W_{pay}/\rho_{pay}}{k_{ve} \cdot WR \cdot T/W + (WR - 1)/\rho_{ppl}} \tag{5.5}$$

5.3.3 Process to Solutions Space Formation (Multi-Point Case)

The subject of this section is the practical implementation of the processes into the development of the multi-point case's principal deliverable, the solution space visualization. This is a frequently misunderstood process.

A solution space is simply a locus of single point designs that can be visualized. An examiner formulates and visualizes a solution space to assist in evaluating trade options and solution behavior in a multidisciplinary environment. The formation of the solution space is the multi-point case's process in action. To understand better the process and, more importantly, the outcome, Figure 5-8 illustrates the process to arrive at the solution space definition. In this illustration, the X-51 type lifting-body 2D scramjet concept configuration and the GHV blended-body 3D scramjet concept configuration are used to give example process context. The example considers a three variable trade scenario for each vehicle: cruise time, volume coefficient (τ), and payload mass.

Figure 5-8 illustrates the stepwise processes to the population of a solution space. This illustrates pictorially the operational results of the process shown in Figure 5-7. Considering Figure 5-8, a solution space is a locus of single point designs, the manifestation of which begins with an initial set of design point solutions. This first consideration is illustrated in Figure 5-8(a). Each point is, in this case, a converged solution for a given set of inputs. The process to arrive at each individual solution is the execution of the single point case's process. The resulting solutions are mapped onto a plot, forming a simple solution space. In this example, each point plotted corresponds to a different mission cruise time. The cruise time is indicated alphanumerically next to each point. This set of points forms a mission cruise time trade for a given vehicle's (X-51 class AB) payload weight, and τ. Similarly, the cruise trade is executed for a new τ, as illustrated in Figure 5-8(b). The diagonal lines, highlighted by the callouts for the values of τ, are lines of constant vehicle τ, with the maximum value (minimum slenderness) appearing on the left and the minimum value (maximum slenderness) on the right.

Continuing with Figure 5-8 (c), another trade variable is introduced. The activities discussed for figures (a) and (b) are repeated but with a new input variable condition (the trade variable). In this example, trades in vehicle payload mass are conducted. The results are added to the plot. As such, this one diagram now illustrates three different trades. Each separately bounded and shaded solution space corresponds to a different payload mass mission of varying cruise endurance and geometric parameter τ. Any number of hardware or operational trade variables could be introduced to expand the solution space, revealing additional design behavior of the concept and configuration selected.

Finally, as illustrated in Figure 5-8(d), the trade option is expanded to include the vehicle concept and configuration. All activities discussed in Figure 5-8(a)-(c) formulation are executed again for the new concept and configuration. In this example, a vehicle of the GHV type is introduced to the trade matrix. Now, represented in this individual figure, is the solution space as given by the trade variables—cruise time, τ, and payload—for both a X-51 class and GHV class vehicle, capturing the behaviors of an all-body 2D scramjet vehicle versus a blended-body 3D scramjet vehicle. At this point, to derive further information from the illustration, constraints could be added to the figure.

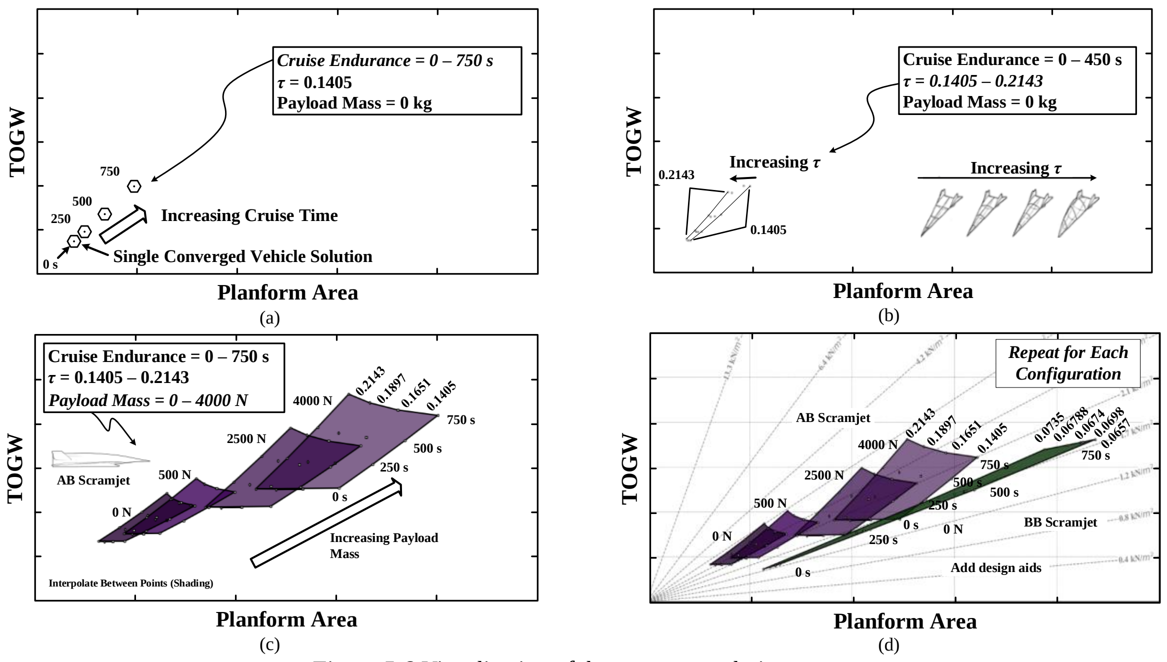


Figure 5-8 Visualization of the steps to a solution space

5.4 Missions Definition

Addressed in this section are the various missions that occur within the verification and multi-point case studies. The section concludes with the synopsis of the vehicles' mission compositional segments and flight conditions.

There are two distinct mission profiles for the hypersonic test vehicle. Both are characterized by being air launched. The discriminating feature is whether an expendable or integrated propulsion system accelerates the vehicle to the primary mission start condition (hypersonic cruise). The air-breathing configurations are limited to the expendable rocket booster scenario. A combined or dual cycle concept is not considered. The non-air breathing rocket configurations are not limited; both mission launch scenarios (i.e. external expendable boost system and internal reusable boost system) are applied to them. All scenarios start with an airdrop condition at Mach 0.8 and 12.2 km (40 kft.), and a horizontal gliding recovery at a landing site.

5.4.1 Expendable Booster Profile

The expendable external booster profile is a profile characterized by the state in which the vehicle's on-board propulsion system does not accelerate the vehicle to propulsive operational conditions. Rather, an external device—an expendable booster rocket—accelerates the vehicle to a condition in which the on-board propulsion can operate and take over as the primary propulsive system. In the non-combined cycle propulsion system, this assistance is necessary, as the scramjet and ramjet are not able to start at the subsonic airdrop conditions.

Figure 5-9 illustrates the external expendable booster mission profile. The vehicle is airdropped from a carrier vehicle at 12.2 km (40 kft) and Mach 0.8. On release, the vehicle is boosted to the test starting condition, the point for onboard propulsive operation at 22.96 km (75 kft) and Mach 4.5. After expenditure, the external booster separates, and the primary vehicle continues to accelerate at constant altitude until it reaches the design cruise Mach number. Acceleration occurs by means of the onboard propulsion system. After accelerating to the test cruise Mach number, the vehicle executes a constant Mach cruise of some duration. On completion of the cruise segment, the vehicle performs a gliding descent to the landing point. The conditions at which each event occurs can vary as a trade variable. Cruise time specifically is a trade variable considered.

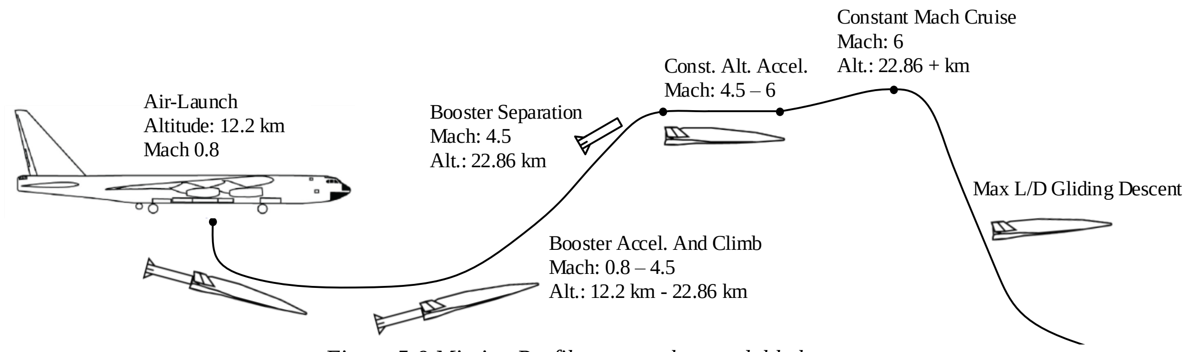


Figure 5-9 Mission Profile: external expendable booster

5.4.2 Internal Booster Profile

Unlike the expendable external booster case, the internally boosted case does not have an external fall away propulsive system; rather the full mission profile is powered through the onboard propulsion system. Given that the vehicle is launched at subsonic conditions, this flight scenario is limited to only the rocket-powered vehicles. Illustrated in Figure 5-10 is the mission scenario profile. The mission starts with an airdrop condition. After carrier vehicle release, the hypersonic vehicle accelerates to its cruise condition by means of the integrated onboard rocket system. The acceleration phase comprises of a constant altitude acceleration to a dynamic pressure of 89.3 kPa (a dynamic pressure that correlates to Mach cruise condition at cruise altitude) followed by a constant dynamic pressure climb to the cruise condition. Upon achieving the desired cruise condition—altitude and Mach number—the vehicle ceases acceleration and executes a constant Mach cruise for a predetermined cruise time. Upon completion of the cruise segment, the engine is shutoff and the vehicle glides to a landing condition. This mission trajectory profile mimics the profile of the externally boosted scenario.

As there is no drop-away external boost system, the test vehicle is its own accelerator. The integrated onboard main engine powers acceleration and cruise. Significantly, this mission sizes the vehicle to include the propulsive capacity previously provided by the external expendable booster. In this regard, the all-rocket vehicles can be potentially fully reusable.

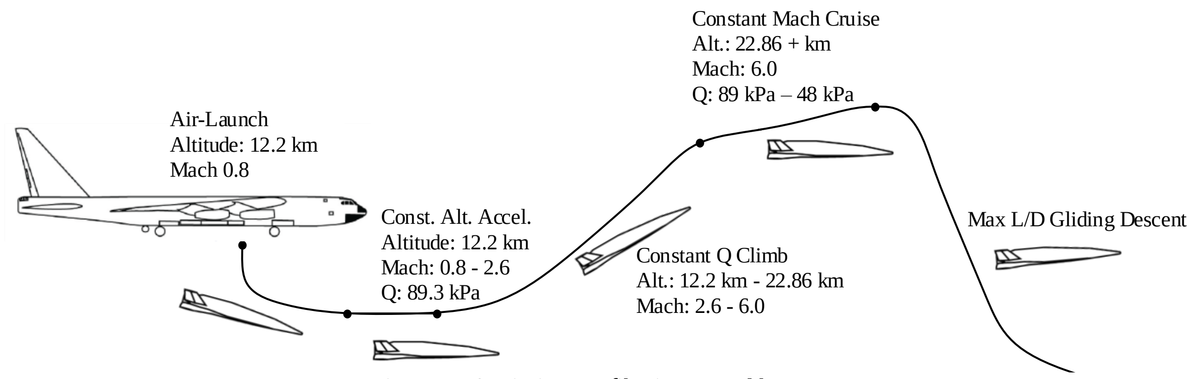


Figure 5-10 Mission Profile: integrated booster

5.4.3 Vehicle Mission Segment and Summary

With an understanding of the two mission scenarios considered, the mission profiles and conditions of each vehicle are summarized in this section. As stated, the total mission profiles comprise of individual mission segments. The mission segments correspond to specific flight conditions and methods (the methods are discussed in section 5.5 Methods). It is, in part, for this reason that the missions are decomposed into their primary constituents. For clarity and convenience, the mission segments comprising the total mission for each vehicle are indicated in Table 5-3. Both the verification and trade study cases are indicated.

Table 5-3 Vehicle Mission Segments toward total mission profile

	Verification		Trade Study					
	GHV	X-51	BB3DS	AB2DS	ABRKT		BBRKT	
Mission Type					Non-Int.	Int.	Non-Int.	Int.
External Boost Launch to Cruise	●	●	●	●	●		●	
Internal Boost Launch to Cruise						●		●
Mission Segments								
Gliding Descent	●	●	●	●	●	●	●	●
Constant Mach Cruise		●	●	●	●	●	●	●
Constant q Climb	●					●		●
Constant Altitude Acceleration	●	●	●	●	●	●	●	●
Air Launched	●	●	●	●	●	●	●	●

Table 5-3 indicates each vehicle’s mission segments, it does not, however; indicate the flight conditions at each mission segment. Table 5-4 and Table 5-5 provide the mission segment flight conditions for the verification and trade study cases, respectively. Note that for the BBRKT and ABRKT integrate boost type case, the table columns do not correspond to mission order. The constant altitude acceleration segment occurs prior to the boost segment as described previously.

Table 5-4 Verification vehicles’ mission segment flight conditions

Vehicle	**Mission Type**	**Start Condition**		**Booster Acceleration and Climb**		**Internal Propulsive Acceleration**		**Constant Mach Cruise**		**Gliding Descent**	
		Alt	Speed	Alt	Speed	Alt	Speed	Alt	Speed	Alt. Start	Alt. End
GHV	Boost Launch to Cruise	15	0.8	20.6	4.5	20.6 to 24.2	6	24.2+	6	24.2+	0
X-51	Boost Launch to Cruise	15	0.8	18.3	4.5	18.3	6	18.3+	6	18.3+	0

All speed in Mach Number
All Alt. in km

Table 5-5 Trade study vehicles’ mission segment flight conditions

System	**Boost Type**	**Start Condition (Airdropped)**		**Booster Acceleration and Climb**		**Acceleration**		**Constant Mach Cruise**		**Gliding Descent**	
		Alt	Speed	Alt	Speed	Alt	Speed	Alt	Speed	Alt. Start	Alt. End
BB3DS, AB2DS	External	12.2	0.8	22.9	4.5	22.9	6	22.9+	6	22.9+	0
BBRKT, ABRKT	External	12.2	0.8	22.9	4.5	22.9	6	22.9+	6	22.9+	0
	Integrated	12.2	0.8	22.9	6	12.2	2.6	22.9+	6	22.9+	0

All speed in Mach Number
All Alt. in km

5.5 Methods Selection

Continuing with addressing the problem setup, this section addresses the methods used for the verification case study and the trade study case. As the trade study case uses the verification vehicles as baselines, many methods are shared between cases. The methods are addressed in a top overview approach; for details on select methods see the noted references.

5.5.1 General Method Overview

A listing and general overview of the principal methods employed are presented in Table 5-6. The methods are categorized by discipline. The disciplines are Geometry, Aerodynamics, Propulsion, Trajectory, and Weight and Volume. Each discipline consists of at least one method and may contain more than one. Not all methods are applicable simultaneously nor necessary to a single vehicle. The method-vehicle associations are given in section 5.5.2 Method Application Summary.

Table 5-6 Summary of methods applied

Discipline	Methods	Description	Reference
Geometry	FDL-7/Model-176, GHV, GHV modified, and X-51 baseline geometries	Geometry analytical relations and look up table modules with data populated by configurations created in NASA openVSP	[147, 148, 150]
Aerodynamics	Subsonic, Transonic, Supersonic, Hypersonic (blended-body and lifting-body)	Empirical McDonald Douglas aerodynamic relations for estimating lift-to-drag ratio $(L/D)_{max}$, lift curve slope C_{L_α}, induced drag factor L', and zero lift-drag coefficient C_{D_0}	[85, 151]
Propulsion	Rocket Performance	Off and on design point analytical relations for determination of I_{sp} and thrust available, T	[152]
	2D Scramjet Performance	Off and on design point analysis incorporating stream thrust analysis and CEA based fuel properties to determine I_{sp}, thrust available T, and fuel flow rates	[153]
	3D Scramjet Performance	Custom method derived from the GHV's propulsion system	[147]
Trajectory	Const. Alt. Acceleration Const. q Climb Gliding Descent Constant Mach Cruise Air Launch / Booster Separation	Numerical method for small flight path angle atmospheric flight.	[154]
Weight & Volume	Transatmospheric vehicle sizing	A set of empirical and analytical relations for the identifying of weight and volume of the vehicle and its subsystems	[85, 149, 155]

5.5.2 Method Application Summary

With the methods available presented, now the methods applied per vehicle is considered. Since the vehicles share many trajectory segments and since the methods are very generic, many of the methods are used across the vehicle spectrum. Table 5-7 shows the methods per vehicle breakdown. The filled bullet indicates the application of the method to the given vehicle. Note that the geometry method tool is the same across all vehicles; however, the individual method module's data is different per vehicle. The tool (openVSP and supporting script) is used to populate the data necessary for the individual geometry method module.

Table 5-7 Methods per vehicle application summary

	Verification		Trade Study					
	GHV	X-51	BB3DS	AB2DS	ABRKT		BBRKT	
Geometry					Non-Int.	Int.	Non-Int.	Int.
openVSP	●	●	●	●	●	●	●	●
Aerodynamics*								
Subsonic	●	●	●	●	●	●	●	●
Transonic	●	●	●	●	●	●	●	●
Supersonic	●	●	●	●	●	●	●	●
Hypersonic	●	●	●	●	●	●	●	●
Propulsion								
Rocket Performance					●	●	●	●
2D Scramjet Performance		●		●				
3D Scramjet Performance	●		●					
Trajectory								
Gliding Descent	●	●	●	●	●	●	●	●
Constant Mach Cruise	●	●	●	●	●	●	●	●
Constant q Climb	●					●		●
Constant Altitude Acceleration		●	●	●	●	●	●	●
Air Launch	●	●	●	●	●	●	●	●
Weight and Volume								
Transatmospheric Sizing	●	●	●	●	●	●	●	●

* A different module is used for each speed regime depending on if a BB or an AB

5.6 Trade Matrix

A trade matrix is established for the trade study case. There is no trade matrix for the verification case; no vehicle properties are traded, rather, the goal is to arrive at the given vehicles within reasonable error. As such, the trade matrix discussed herein is in regard to the trade study case only.

The trade study case is an exploration of the air-launched reusable hypersonic test vehicle solution space. The examination is for the growth vehicle case. That is, the vehicles trades are to include increasing capability to identify how the vehicle size varies with capability variance. The reader could consider the vehicles sized similar to those of hypersonic missiles of varying capability. The trade matrix is given in Table 5-8. Observe that the concepts themselves are a trade. For each concept, the mission variables are traded, specifically cruise time and payload. Additional trades per concept configuration include geometric volume coefficient τ and propulsion system fuel (hydrogen and kerosene). The trade matrix indicates the range evaluated; however, note that not all points converge under the convergence criteria specified, which in itself can be informative. See Appendix A Case Studies Expanded for an account of the non-converged and converged trade points.

Table 5-8 Trade study trade matrix

Vehicle Tag	Baseline Vehicle	Propulsion System	Boost Type	Fuel Type	Tau Range	Payload (N)	Endurance Cruise (s)
BBRKT	GHV	Liquid Rocket	External	H_2 / RP-1	0.09 - 0.12	0 – 4000	0 – 300
BBRKT	GHV	Liquid Rocket	Internal	H_2 / RP-1	0.09 – 0.12	0 – 4000	0 – 300
BB3DS	GHV	3D Scramjet	External	Ethylene	0.0657 – 0.0735	0	0 – 750
ABRKT	MODEL 176	Liquid Rocket	External	H_2 / RP-1	0.1405 – 0.2143	0 – 5000	0 – 500
ABRKT	MODEL 176	Liquid Rocket	Internal	H_2 / RP-1	0.1405 – 0.2143	0 – 5000	0 – 500
AB2DS	X-51 / MODEL 176 Scram	2D Scramjet	External	JP-7	0.1405 – 0.2143	0 – 4000	0 – 750

5.7 Results: Single Point Verification Case

The verification case was executed with satisfactory conclusion. The verification case implemented the synthesis sizing code as established in the preceding chapter sections. Table 5-9 presents the sizing variable results. Both the calculated value and the percent error ($\% \, Error = 100 \times (Actual - Calculated)/Actual$) to the known value are given. Through the verification case execution, the methods have been calibrated as well. The percent error has been reduced by calibrating the method to better arrive at the known vehicle sizing variables' values. The X-51 and GHV error values are all within 5% of the known values with the majority below 1%. This error is acceptable at the early conceptual design stage where in speed to evaluate the largest possibility of solutions concepts is paramount.

Table 5-9 Verification case's sizing variables' value and percent error

	GHV 1X			GHV-5X			X-51A		
Parameter	Actual	Calculated	% Error	Actual	Calculated	% Error	Actual	Calculated	% Error
Tau, τ	0.0735	0.0735	0.0	0.0657	0.0657	0	0.2074**	0.2074	-
Planform Area, (S_{pln}, m^2)	3.88	3.895	-0.189	19.45	19.469	-0.062	2.26**	2.266	-0.271
Total Volume, (V_{tot}, m^3)	0.563	0.565	-0.368	5.642	5.644	-0.034	0.705**	0.730	-3.587
Length, (l, m)	4.468	4.471	-0.068	9.991	9.996	-0.048	4.267	4.273	-3.167
Span, (b, m)	1.488	1.491	-0.189	3.327	3.333	-0.206	0.702	0.7029	-0.134
Takeoff Gross Weight, (TOGW, N)	5430	5552	-2.247	36456	36238	0.598	6690	6689	0.016
Wing Loading, ($W/S_{pln}, N/m^2$)	1397	1425	-0.022	1874	1861	0.694	2960	2952	0.270

*usable fuel

**estimate

With the execution of the synthesis modules and satisfactory arrival at minimal percent error from the calculated value to the known value, the verification and calibration case study is considered complete. From the execution of the synthesis code and arrival at satisfactory sizing variable results, it is inferred that the AIDRA-DSS system executes correctly for the single vehicle case. The verification case illustrates correct system execution and instills confidence in the system and the methods selected for the vehicles considered. The next task is to consider the trade study case.

5.8 Results: Trade Study Case

The results of the trade study are presented in the following sections. The results are considered individually according to boost scenario and propulsion type. The discussion closes with the presentation of the total solution space with all scenarios shown with carrier vehicle constraints considered. Note that the solution spaces presented include marked solutions (design points); however, the reader should be aware that the problem at hand does not necessitate a unique solution. The solutions presented here are not necessarily 'optimal' solutions. For the given design trade combination, there could be any number of plausible and practical solutions. Here, a map of the solutions arrived at are presented and probable areas of viable solutions are indicated.

5.8.1 External Booster

The externally boosted concepts include both airbreathing and non-airbreathing concepts. For clarity, the solution space of each is presented separately. Figure 5-11 presents the airbreathing vehicles' solution space. Figure 5-12 and Figure 5-13 presents the non-airbreathing vehicles' solution space for the RP-1 and H_2 based systems, respectively. For reference, in both figures, the wing-loading of 3,413 N/m^2 is highlighted. This wing-loading corresponds to the landing design requirement of the X-24C [146] (a test vehicle further along in the FDL family vehicle evolution). Additionally, note that these solutions are for the cruiser only, the total stack is considered in a following section.

The AB solutions converge along lines of higher wing-loading than those of the BB. The AB airbreathing solutions converge along the 2,500 N/m^2 wing-loading line and are bracketed by the 3,300 N/m^2 and 1,650 N/m^2 lines. The BB solutions converge along approximately the 1,750 N/m^2 wing-loading line and are bracketed by the 1,900 N/m^2 and 1,200 N/m^2 lines. The wing-loading is directly relatable to the stall speed and structural loads. Given that the AB solutions converge along higher wing-loadings, the AB will have higher approach speeds unless supplemented. Historically this configuration type (AB) has been equipped with a secondary retractable lifting device for landing. It is for this reason that all AB concepts considered include an integrated deployable low-speed wing. At lower wing-loadings, the BB concepts do not require additional lifting support, overall representing an advantage.

The AB solutions for 0 to 4000 N payload and 0 to 750 second cruise time are indicated. In general solution convergence occurs with ease and as such the problem is well behaved. The solution areas expand vertically with mission variable trade, increasing in TOGW with minimal expansion (comparatively) in the planform area per τ. This is the volumetric design behavior and advantage of the AB configuration.

The BB solutions for 0 N payload are also shown. Unlike the AB solutions, the BB solutions expand significantly both vertically and horizontally with mission variable trade, increasing in both TOGW and planform area. Note that the single zero payload cases span the entire planform solution length as the AB solutions for 0 – 4000 N payload. It is noted however that the BB solutions are initially under sized and require scaling to closer match known design points despite the calibration of the methods. During operation, note that in some cases design point solutions are non-unique; it is possible that additional smaller scale solutions can be found. It is warranted that further investigation be conducted to verify if reduced vehicle size solutions do exist. As such, due to the scaling factor, the higher endurance design points appear to suffer from overestimating TOGW when compared to the reference GHV tool calibration vehicles in this region. In regard to the general behavior of the solutions, the expansive nature in both planform area and TOGW with increasing endurance is not unexpected given that the type of configuration does not share the same geometric efficiency advantage of the AB. However, the BB does maintain a lower wing loading given its trend towards larger planform area.

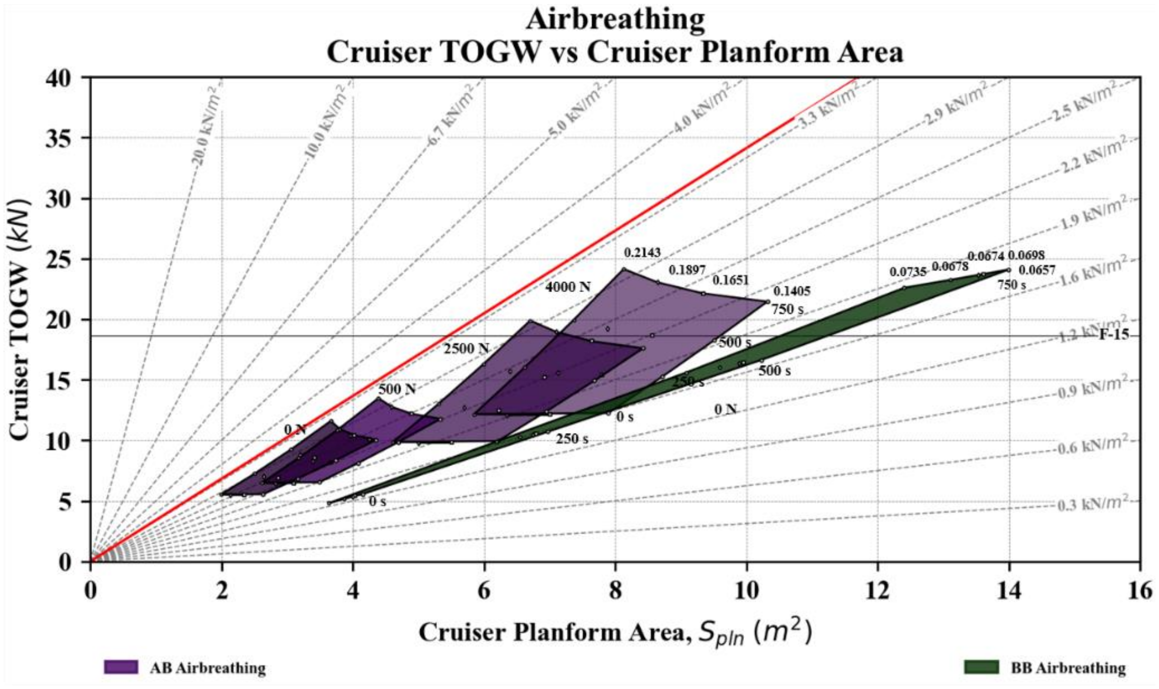


Figure 5-11 Boosted airbreathing hypersonic vehicle solution space: TOGM vs. S_{pln}

Considering the non-airbreathing case of the kerosene (RP-1) and hydrogen (H_2) systems as shown in Figure 5-12 and Figure 5-13 respectively, the solutions in both cases overlap each other significantly. Please note, the differences in payload and mission endurance. It is worth noting at this point that the BB configuration do suffer from significant convergence issues, that is difficulty in finding solutions, especially as compared to the AB case. The reader will notice differences in trade variable ranges and points of no solutions for certain trade combinations (such as the $\tau = 0.12$ at 4000 N and 300s for the H_2 case in Figure 5-13). The AB case does not suffer as significantly, though there are converged points that significantly exceed the solution trends as in the point in Figure 5-13 for $\tau = 0.2143$ and 5000 N. In regard to the RP-1 and H_2 solutions, they both fall within the wing-loadings of $1{,}300\ N/m^2$ and $4{,}200\ N/m^2$. The RP-1 solution field indicates that the AB concept can offer a lighter and smaller solution for approximately all mission design point cases, whilst the BB is showing only some possible advantage in size at the 0 payload and 300 s endurance point. Overall, the AB growth to mission requirement increases was at a lower rate than the BB. Additionally, and not unsurprisingly, due to having to carry oxidizer on board, all solutions as compared to the air-breathing case, are greater in TOGW and S_{pln}.

Significantly, the fuel type directly impacts the vehicle's TOGW and S_{pln}. For both, the AB and BB, the H_2 cases offer only increased S_{pln} and TOGW for the same mission. Additionally, the H_2 system grows very rapidly in both TOGW and S_{pln}, at a rate much greater than the RP-1 based system per change in mission variable. The hydrogen-based BB grows rapidly in size, an entirely undesirable behavior for the case at hand when evaluating of the trade space for carried demonstrator vehicles. As such, per the solutions gained, the hydrogen fuel poses no benefit.

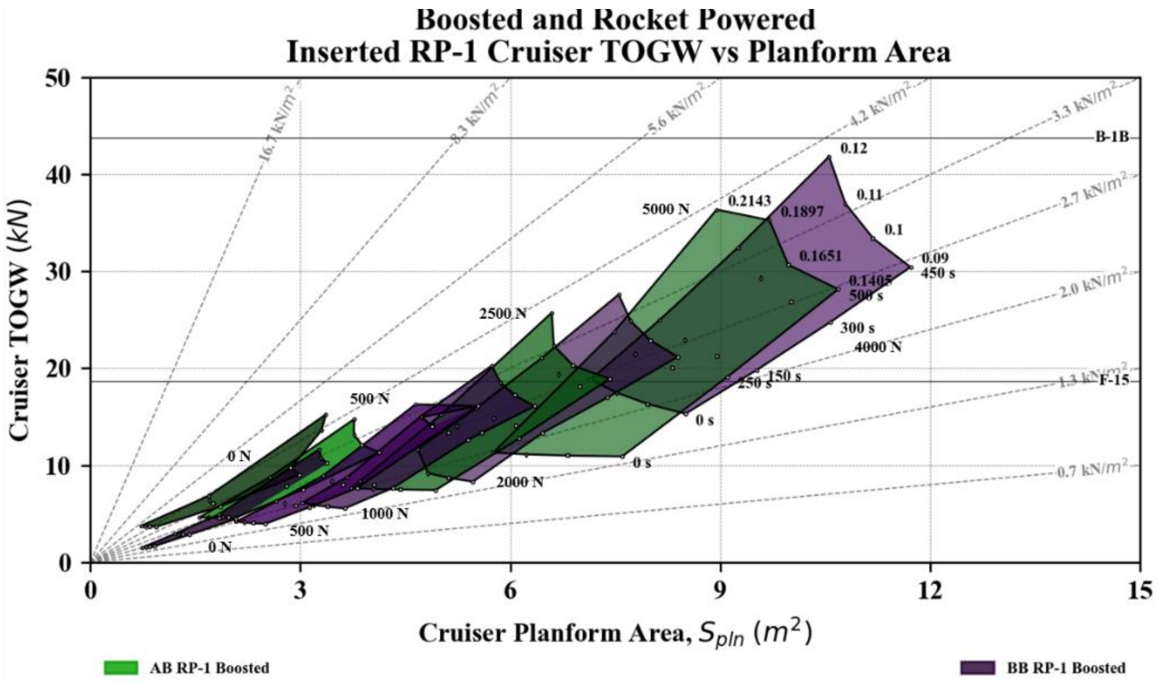


Figure 5-12 Boosted non-airbreathing hypersonic vehicle solution space RP-1: TOGM vs. S_{pln}

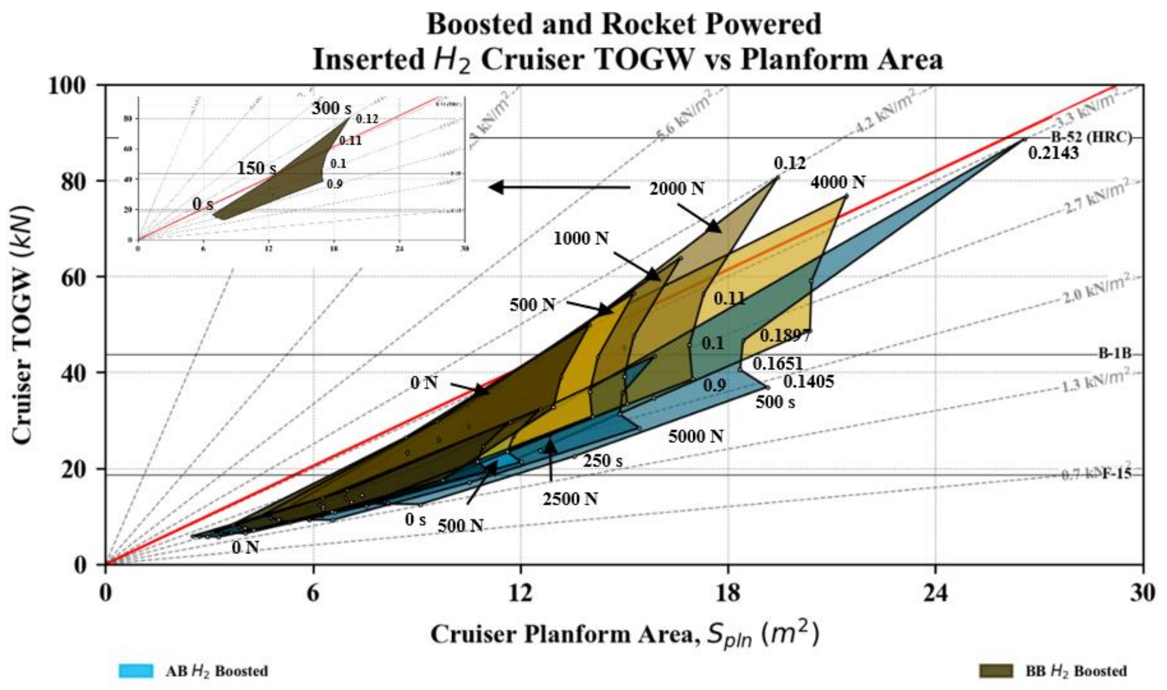


Figure 5-13 Boosted non-airbreathing hypersonic vehicle solution space H_2: TOGM vs. S_{pln}

5.8.2 Trade Study Solution Space: Launch Stack & Carrier Constraints

The previous section considers the cruiser case specifically. With the following, the total stack is considered. The launch stack comprises of the booster, inter-stage, and flight vehicle. The stack is sized based off the X-51A's stack. The total stack is considered in terms of the carrier vehicle constraints. At first, is considered the integrated versus boosted rocket system is considered. This is followed by a general consideration of payload capacities and of geometric constraints where available.

The evaluation of external versus onboard acceleration capacity systems indicates, that the inclusion of full acceleration capacity system integration is not beneficial when judged by design point $TOGW$ and S_{pln}. The figure below illustrates the two cases for the AB case. Note, that the fully integrated systems prove to be a much more difficult problem to solve, in particular for the BB case. Solutions indicate, not unexpectedly, the integrated cases are both heavier and larger than their comparative non-integrated case. However, there are solution points in which the integrated RP-1 system is more advantageous in both S_{pln} and TOGW than the same mission design point for the inserted hydrogen case. There are some cases for the zero cruise time in which the integrated system solutions are less in both weight and area than the equivalent inserted system, which may indicate that the booster rocket is oversized for the case, or that the onboard propulsive system is more efficient then the accelerator motor. Naturally, the integrated hydrogen case has no advantages over any other solution point. As such, in the evaluation of the design points by $TOGW$ and S_{pln}, the integrated solutions show no advantage to the externally boosted vehicle except in the case of an integrated RP-1 based solution being chosen over a hydrogen-based insert solution.

However, if the criterion includes full reusability, the integrated case naturally satisfies the criteria whereas the vehicle accelerated by expendable systems does not. Generally, for the same mission, the integrated systems are approximately twice the $GTOW$ and two-to-three times larger in S_{pln}.

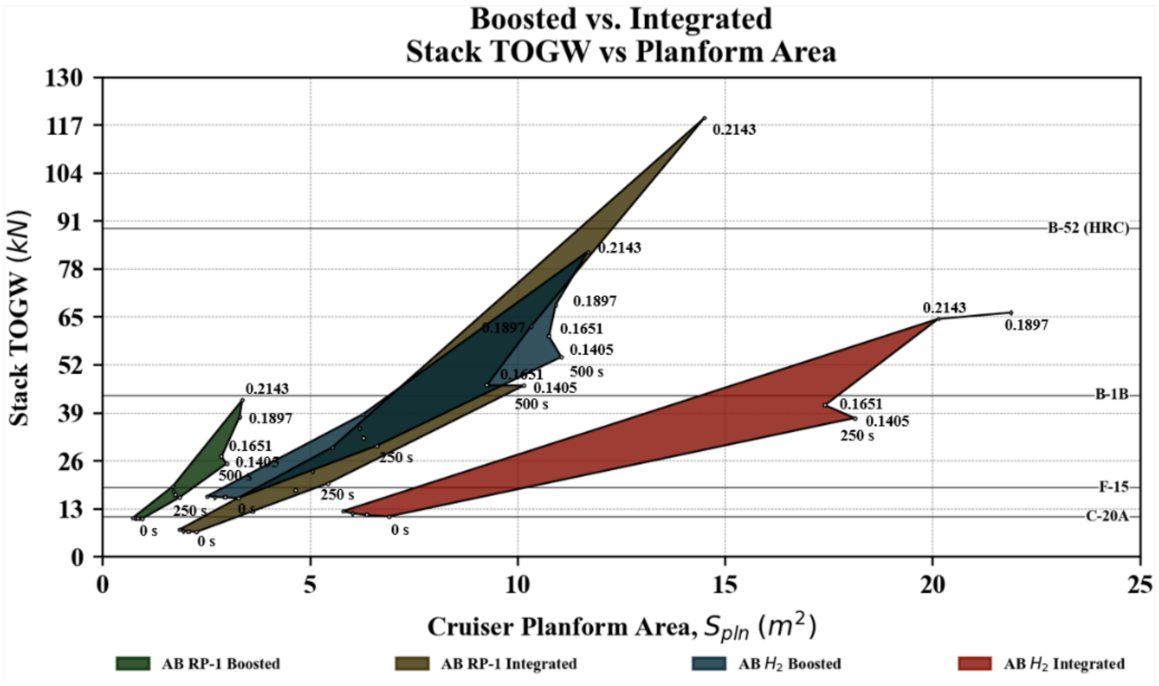


Figure 5-14 Integrated non-airbreathing hypersonic vehicle solution space: TOGM vs. S_{pln}

In Figure 5-15 and Figure 5-16, the launch stack for the AB and BB are presented, illustrating the solution space in regards to TOGW and S_{pln} versus known carrier vehicle payload limits. The pylon hard-point payload limits for several classical vehicles including the B-52 (HRC), F-15, and B-1B in addition to the Gulfstream C-20A, are indicated. Each system has either been used as a launch platform for test systems or is being fitted to carry hypersonic systems. Limits are based on publicly available hard point information or known carried hardware such as the 600 gallon fuel tank for the F-15 midline hardpoint. The selected vehicles represent the lower, middle, and upper limits of generally available carrier aircraft relevant to the hypersonic deployable system. Not included but equally plausible launch platforms include launch vehicles and their components, such as the Falcon 9 or Minotaur.

Considering the carrier vehicle constraints, it is observable that the majority of the solutions arrived at for the inserted vehicles with payloads of $\leq 5{,}000\ N$ and cruse times of $\leq 750\ s$, are within the payload limits of the B-52 with upgraded hardware. The principal exception only being hydrogen-based systems. The integrated solutions would exceed the B-52's capacity quickly as payload increases or cruise time beyond $250 - 500s$ depend on the fuel. However, almost all solutions far exceed the capacity of the C-20A and F-15. The solution field applicable to the F-15 are the AB and BB low-end zero to 500 N payload case up to potentially 250s cruise time of the RP-1 boosted design class and possibly a minimal performance airbreathing system. The B-1B offers a potential launch platform that can address approximately a third to half the solution space identified. The max mission requirements considered do tax the considered carrier vehicle payload capacities. For a growth test vehicle concept, vehicles of a mission requirement greater than that selected, would rapidly exceed the B-52 limits and would require a new launch platform besides the classical systems for hypersonic test systems application.

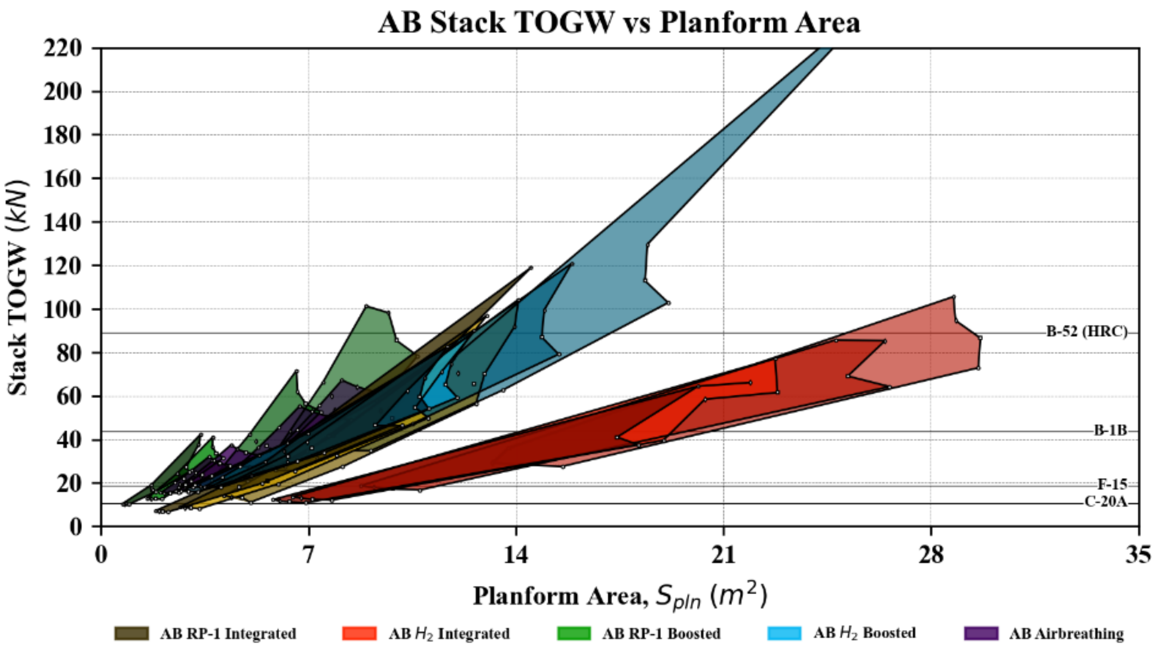


Figure 5-15 AB full stack payload constrained solution space: TOGM vs. S_{pln}

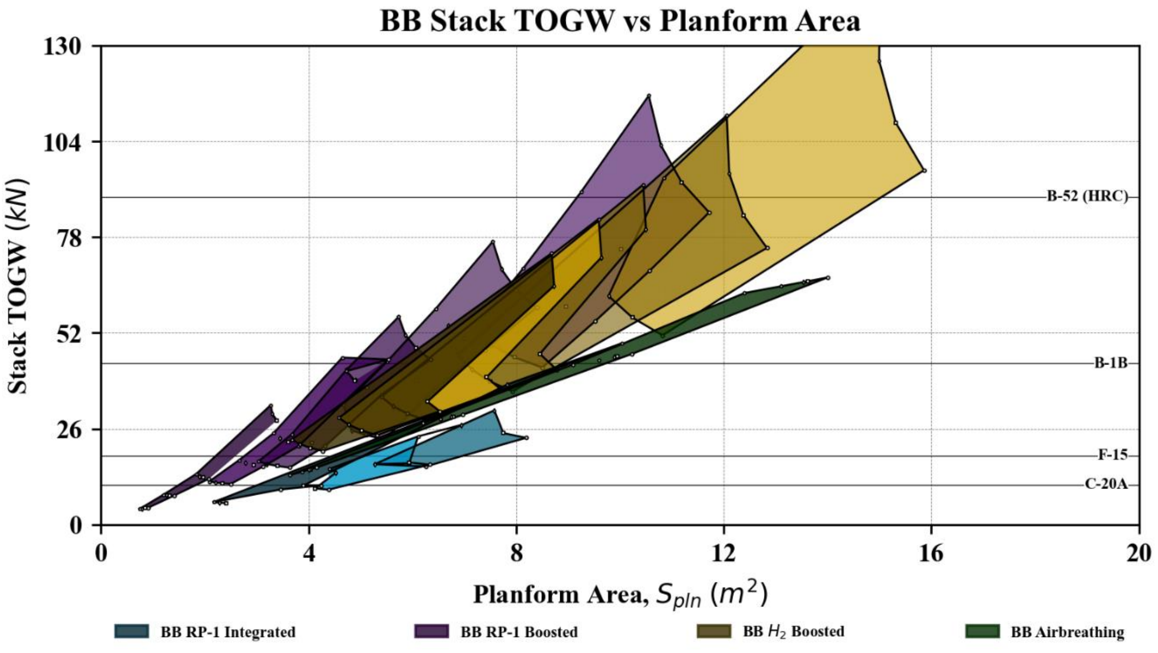


Figure 5-16 BB full stack payload constrained solution space: TOGM vs. S_{pln}

Figure 5-17 and Figure 5-18 illustrates the solution space in regard to the geometric dimensions of launch stack overall length (l) and span (b) versus known carrier vehicle payload geometric limits by BB and AB respectively. As in the previous figure, the B-52 (HCR), F-15, B-1B, and CA-20 are used for payload limits. (The geometric limitations of each vehicle are defined by the payload placement location. Limitations are set based on such parameters as landing gear location, inboard distance between fuselage and engine nacelle, jet wake, and payload CG location.) All vehicle solutions are represented in the figure. The practical solution spaces collapse down significantly when considering length and span. As evident from the figure, the solutions' span values are below the first limiting vehicles span constraint—C-20A. In regard to overall length, many do exceed the limits of F-15 and even the B-52. However, note that the accelerator was assumed to be a single linear component; the accelerator could be potentially divided and placed in parallel along the sides of the cruiser vehicle in order to reduce overall length at potentially the cost of span. Unlike the payload weight limit, in the length limits case, the integrated systems are indicated to have an advantage over the boosted systems, being of less length. Generally speaking, all feasible or likely AB and BB design choices fall within the geometric limits of the B-52. Only select BB solutions exceed the B-52 length limits, those being impractical hydrogen-based systems. In the case shown, the limiting factor is the stack or vehicle length. However, this likely could be solved through division of the accelerator into smaller elements fastened to the vehicle's fuselage rather than tail end.

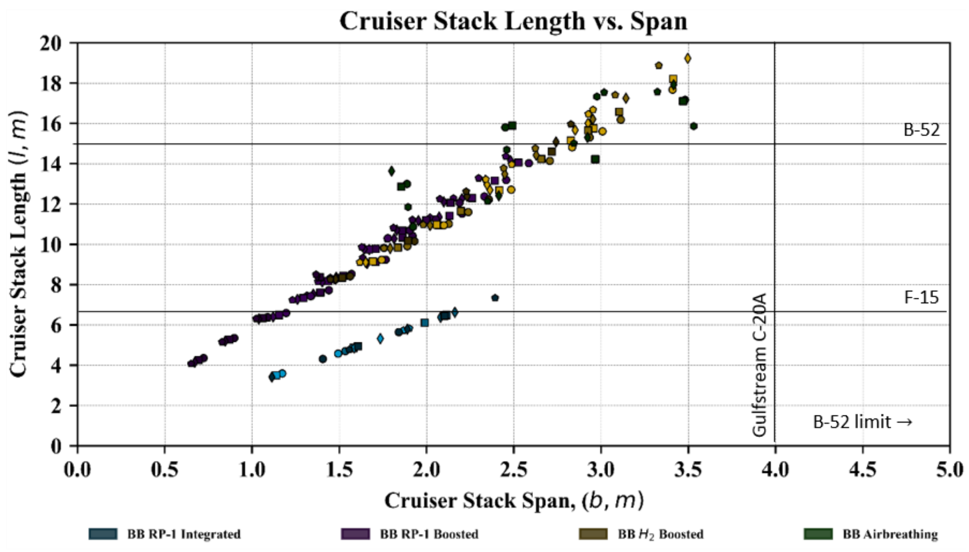


Figure 5-17 BB vehicle geometric constrained solution space: l vs. b

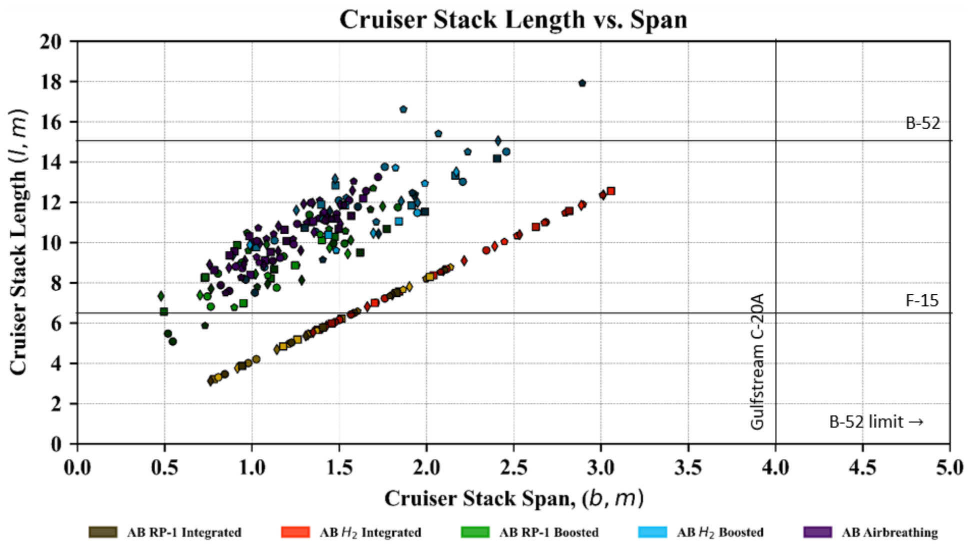


Figure 5-18 AB vehicle geometric constrained solution space: l vs. b

5.9 Conclusion

5.9.1 Study Summary

The principal intent of this chapter has been to demonstrate system functionality and application. Functionality and applicability are shown through a verification case study and a trade study. For system verification, two vehicles have been selected to be sized and the results compared against each other. The vehicles are the X-51A and GHV. These vehicles represent the hypersonic all-body and blended-body vehicle classes. The vehicles selected include both a flown test vehicle and a paper concept study. The vehicles are entered into the AIDRA-DSS system. Through user operation of the system, the result of the user selections via the user interface represents the generation of unique synthesis codes, each addressing a specific vehicle. The vehicles, processes, methods, and mission conditions for each have been presented. Upon synthesis code generation, each has been executed. The result is a satisfactory arrival at sizing results very near in value to the control vehicles known sizing variables' value. The percent error is within 5% for all sizing variables evaluated (most below 1%). The execution of the support system, the execution of the resulting synthesis tools, and the evaluation of the results indicate that the tool works properly. It is inferred that the system operates, that it uniquely assembles new synthesis codes, and that it executes the codes correctly as confirmed by the correct outcomes of the verification case.

Following the verification case, the systems applicability and functionality is demonstrated through the execution of a trade study. Using the vehicle concepts from the verification case as baseline vehicle concepts—with the addition of a rocket powered BB and AB concept—a trade study has been executed. Trade variables include configuration and concept (the vehicle base lines representing airbreathing and non-airbreathing blended-body and all-body concepts), payload, mission scenarios, and fuel types. The trade study not only demonstrates system functionality in terms of a multi-design point study, but also demonstrates system applicability. Through the range of design trades, a solution space for hypersonic test vehicles is assembled and visualized. The solutions are contextualized through the consideration of carrier vehicle geometric and weight constraints.

5.9.2 Study Conclusions

In regard to the trade study itself, a few concluding statements can be made. First, for all cases the AB solutions tended to have the advantage in in planform area, length, span, and even generally weight as compared to the BB counterparts. The BB solutions tend towards being larger in planform area for the same mission. Furthermore, the BB solution areas, for both airbreathing and non-airbreathing concepts, expand with performance demand significantly more so than the AB cases. Regarding the integrated systems, they show no advantage in overall weight, minor potential advantages in overall length are identified. Moreover, the integrated system represents a fully reusable system and for this reason could be more desirable than the other concepts for certain research objectives. Lastly, considering the carrier vehicles, for the missions and performance requirements selected, the B-52 could carry most probable near term systems identified. However, the trades considered result in solutions that approach the limits of the B-52 and as such, for any concepts that exceed those considered here in, an alternative launch platform or fully self-sustained concept would be required. From the study, it can be concluded that for a growth test hypersonic program of reusable systems, the carrier vehicle options are adequate for near term research but are limited and necessitate being considered in program planning far beyond near term. However, in consideration of the solutions as weapon or small-scale test systems and not growth research and development systems, then vehicles of the payload class of the B-1 and B-52 can carry several small systems simultaneously.

With the execution of these case studies, it has been shown that the system operates as expected for the single process and single complex vehicle synthesis assembly and execution scenario. Additionally, with the completion of the case studies, the capacity for trade study is demonstrated. In conclusion, all verification and demonstration tasks as outlined at the start of this chapter have been completed successfully.

5.9.3 Contribution Statement

» Verified single point and multi-point functionality of a new synthesis design tool.

» Solution space identification and visualization of all-body and blended-body airbreathing and non-airbreathing reusable hypersonic air-launched vehicles for 0-750s cruise time at varying payload, fuel type, and trajectory design.

Chapter 6 CONCLUSION

This document is concluded with the consideration of a research summary, a research contributions summary, and a consideration of areas of research and development for future enhancement of the concepts and system presented here in.

6.1 Research Summary

The principal development and deliverable of the research presented herein has been the generic synthesis decision support environment as a precursor to an artificial intelligence design and research assistant (AIDRA-DSS). *AIDRA-DSS* was developed in Python with an executable GUI written in QT. It is a framework that allows engineers to design and size any vehicle through a generic synthesis assembly approach. Additionally, the system is not limited to aerospace and, as long as the designer carries the proper methodologies, a vehicle can be assembled. The ideology behind *AIDRA-DSS* is a versatile system that can size and prototype vehicles in a fast-paced design environment. Giving the user the ability to compose a vehicle from different elements, *AIDRA-DSS* creates a tailored sizing code based on the user-designated requirements.

The systems functionality and applicability has been demonstrated successfully through the execution of a verification case and a trade study. The verification case considered the GHV and X-51A. Representing the blended-body and all-body configurations, these concepts and configurations were used as baseline vehicles for the trade study. The trade study evaluated air launched airbreathing and non-airbreathing concepts for consideration as reusable hypersonic vehicle research and development platforms. Trade variables include configuration and concept, payload, mission scenarios, and fuel types. Through the range of trade conditions, a solution space for hypersonic test vehicles was assembled and visualized. To assist in evaluating concepts and gleaning information from the results in pertinence to the carrier vehicle, the solutions were placed into context with carrier vehicle geometric and weight constraints. From the study, it can be concluded that for a growth test hypersonic program of reusable systems, the carrier vehicle options are limited and necessitate being considered in program planning. The vehicle solutions indicate plausible requirements for future carrier vehicles. Additionally, the all-body has shown superior solution regions in terms of total weight and size, both of which are critical for carrier vehicle consideration.

With the execution of the two case studies, it has been shown that the system operates as expected for the single process and single complex vehicle synthesis assembly and execution scenario. Additionally, with satisfactory execution of the case studies, the capacity for system handling of said trade studies is demonstrated.

The development of the system has had two purposes. First, advancement toward modular design synthesis assembly infusion into cognitive systems or other AI frameworks. This is the driving motivation of this research. In this respect, the purpose of the system is to develop further expertise and a baseline environment to test complex vehicle automated synthesis architecture synthetization that would be easily adaptable into a greater cognitive system. The second purpose is to serve as a useful engineering environment that arrives the user at an applicable synthesis solution toolset, based on user selections, to solve a given problem by providing standard feedback and decision aiding platforms. The second objective one could consider as an intermediate objective to provide immediate system utility while driving towards the greater objective of a cognitive design and research assistant.

Reiterating the statements of Chapter 1, fundamentally, the motivation for this research has been to explore the advancement of toolsets for the decision maker and designer operating at the earliest planning and design phase of an aerospace vehicle or program. The significance of the decisions made at these early phases cannot be overstated. The level of success of a product is dependent on the quality of the underlying early forecasts, requirement definitions, technology selections, and initial concept and configuration selections. This research has been but one-step towards a greater goal; the system itself could have additional immediate advancements through follow on research and development. Some foreseen and suggestable areas of development are discussed in section 6.3 Research and Development for Future Enhancement.

6.2 Research Contribution Summary

The research contributions are summarized below. The contributions are in the areas system concept specification, system development, system demonstration, and trade study execution.

- » Specification for a design decision support system environment concept for application to aerospace vehicle design, with an approach emphasis for vehicle-of-vehicle design.
- » Development of a modular and automated synthesis assembly toolset in the framework of a transparent and user-friendly decision support environment.
- » Demonstration of environment functionality through a verification case study.
- » Identification of a reusable hypersonic demonstrator class solution space for all-body and blended body vehicles of both airbreathing and non-airbreathing type.

The research undertaken has developed software to directly assist the early conceptual design phase and even the activities of the pre-design phase, notably the program planning and road-mapping activities that can include but are not limited to technology portfolio planning and requirements identification. As such, principal tool development and its application does focus on the parametric sizing phase, itself presenting the initial sub-phase of the conceptual design process. Any design naturally follows a refining processes and, as such, demands that the synthesis software increase its analysis fidelity successively as well. To accommodate this natural event of refinement through increasing analysis fidelity, the system presented would approach the problem through the generation of multiple architectures of varying analysis fidelity. As such, the fidelity of analysis is variable, and the tool and approach are applicable to more than one design event within the design process.

There are several additional advantages to the system's approach as identified and developed. First, through a transparent process of a dynamic method and analysis process definition and selection sequence, a user can implement a design analysis process that directly reflects the needs and requirements of the product at hand, as each product can have different criteria and design initiation avenues. For example, in the classical design approach, a basic geometric definition is the first step. However, it may not be necessary that a design initiates with this particular discipline nor that a process be limited to certain disciplinary areas and sequences. The disciplines represent an analysis type and the user is free to add any discipline to create and apply a new process as necessary. Thus, the synthesis architecture process is moldable to arrange the order of analysis as necessary to address the unique problem being solved. Modules can be added as needed to address the necessary analysis including cost models, life support models, radar cross-section analysis models, thermal heating models, etc. In summary, the process is user definable and not limited to a specific disciplinary analysis module application nor order of operation. However, the user tends to apply engineering best practice and knowledge in the design of a process to ensure correctness and peer-acceptability in application and design.

The modular process definition underscores an additional advantage to this approach and application, that is the ability to concurrently evaluate dissimilar concepts and configurations conceived to address the same problem. A classical design problem is the inability to rapidly compare uniquely different aerospace design configurations for the same mission in a timely manner. The automated modular approach presented, through a library of various processes and methods, permits the rapid assembly of architectures of consistent fidelity that each address different design concepts allowing for the comparison of potentially very dissimilar solutions on equal design evaluation footing.

A final notable benefit of the automated modular synthesis approach within an easy to use and transparent user interface, is the savings in time. Time in two regards; first time in regard to engineer training and time in regards to analysis deliverables. A key to learning is exposure. A rapid architecture generation capacity—of a generic type—allows for increased engineer exposure to various design processes, analysis methods, and aerospace concepts and configurations. The automated approach can directly contribute to an enhanced learning environment through which a novice engineer can rapidly gain exposure and design understanding. A synthesis architecture generation system of a transparent nature that operates through synthesis assembly automation by means of a modular design process and methods library, allows for the rapid introduction and exposure of inexperienced personnel to the labyrinth of available knowledge and the design processes of the institution. This subsequently permits increased design exposure and general experience, such that the experience and knowledge available can be more readily directed and passed on to the novice designer. Similarly, the same advantage is applicable to the university environment, where student exposure to the actual evaluation and comprehension of the design and the value of their work, is lost due to frequent lack in time for exposure and experimentation after the initial generation of method analysis tools. A modular automated synthesis generation approach allows for reduced time and effort in the design tool fabrication process, that if employed would allow for increased time in analysis application, design understanding, and overall improvement in an engineer's education.

The second note on time savings is that of actual tool generation which has been hinted at in the previous discussion. An automated synthesis architecture generation process permits for the development and deployment of synthesis architectures rapidly. Time savings occur through the automation of the tedious tasks of linking methods and data handling, in addition to identifying and presenting to the user methods and processes available, with potential specification of the best methods and tools for the hardware application and fidelity required in addition to other requirements, ensuring proper method application. In the event of adequately populated libraries (process, methods, and vehicles), input variable databases, and dependent on hardware decomposition level, architectures can be rapidly generated within mere minutes and architectures executed thereafter. Furthermore, they are additionally archivable and distributable with full access to the methods involved allowing for reuse and modification, as necessary.

6.3 Research and Development for Future Enhancement

The work presented here has been an iteration of a vehicle decomposition and modular synthesis assembly concept, with the goal of sequential development towards a cognitive design and research assistant. This work has led to the development of a modular framework that can be refined and inserted as a numerical analysis core into an AI framework with modification of course. As such, some foreseen efforts for continuation and improvement of the design kernel include the following.

Towards Architecture Planning:

» Vehicles as Trade Studies. The system current is set to treat each vehicle selected as a component of a greater vehicle; however, this same implementation could readily be converted to allow for trade study iterations to include the vehicles themselves. Vehicle trades could occur in two manners. First, rather than setting a vehicle group as a subset of a parent vehicle, they could be set to be individual independent concepts to be evaluated. Such a capacity would support pre-phase or program and architecture planning. A second case, a parent vehicle's component sub-vehicles could be traded in addition to the classical trades such as aspect ratio or fuel type. This would allow for a single project set up and run to evaluate many vehicle concepts simultaneously.

» Decomposed System Trades. In the same fashion as the point noted above, the individual components of a vehicle—whether it be the vehicle itself, vehicles within the vehicle, or any individual vehicle's concept, hardware, or operational conditions—could be a trade option. Traditionally trades are in specific input parameters; this system would allow a trade study to be much more global in consideration.

Towards Increased and Improved Automation:

» Natural Language Processing. The GUI itself is in place to benefit the user. However, for an autonomous system, the GUI is not necessary. A plausible improvement could be a natural language processor where in the instructions are given either verbally or via a standard keyboard input (text). In such as sense the system could have an integrated chatbot like interface. The GUI operations would be handled by the system without direct user interfacing other than through basic instruction however being dependent on a sound expert system or similar.

» Database and Knowledgebase Expansion and Integration. Improved and increased knowledgebase and database integration and population would directly benefit the system. Furthermore, it is necessary for a true research and design assistant. Currently, the system is limited to a selection of methods and references as the knowledge base and the project results exist in fragmented result databases. Additional knowledge and data handling capacity could be added through many means. For instance, the addition of a global projects results database with datamining and data reuse could be integrated allowing for a mechanism for improved convergence parameter initial guess values, input variables value assignment, or even new method derivation through datamining. Furthermore, system execution could be enhanced through an improved expert system or decision tree/process for automated method filtering based on knowledge of the available methods such as method applicability, accuracy, speed, or dependability. Given a concept statement, with the right knowledge and data, the system could make the correct choices the user classically makes during GUI operation allowing for time reduction in system operation and final product.

» Design and Analysis Recommendations. The purpose of a decision support system is to help the user arrive at the correct or best decision given the information available. The current system assists the user by providing a transparent synthesis assembly tool allowing the user to setup an analysis solution to the problem at hand that results in not only the analysis tool but also design figures. The user is left, however, to derive conclusions based on the results and figures presented. All though this may accelerate the problem solution process, it itself does not give design recommendations. A system improvement would be a direct result analysis process that arrives at and provides both intelligent design recommendations and intelligent analysis recommendations (recommendations to rerun the analysis to better evaluate the problem or new problems identified through the analysis, including changing methods).

» Improved Generative Coding. A primary deliverable of the DSS developed is a synthesis code tailor made for the user given the user’s selections in system operation. The system generates the synthesis code effectively through the use of code assembly rules and code block libraries. The system could be further advanced through improvements to the code assembly process such as through the utilization of agents or other auto-coding and generative techniques.

Chapter 7 BIBLIOGRAPHY

[1] Blair, J., Ryan, R., Schutzenhofer, L., and Humphries, W. "Launch Vehicle Design Process: Characterization, Technical Integration, and Lessons Learned," NASA/TP 2001-210992, NASA, 2001.

[2] Anon. "AIAA Technical Committee on Multidisciplinary Design Optimization (MDO) White Paper on Current State of the Art," 1991.

[3] Defense Acquistion University Press. "Systems Engineering Fundamentals," Department of Defense Systems Management College, 2001.

[4] Calkins, D. E., Gaevert, R. S., Michel, F. J., and Richter, K. J. "Aerospace System Unified Life Cycle Engineering: Producibility Measurment Issues," IDA Paper P-2151, Institute for Defense Analyses, 1989.

[5] Corning, G. *Aerospace Vehicle Design*, College Park, MD, 1964.

[6] Space & Missile Systems Center. "SMC Systems Engineering Primer & Handbook: Concepts, Processes, and Techniques," U.S. Air Force, 2005.

[7] Space & Missile Systems Center. "SMC Systems Engineering Primer & Handbook: Concepts, Processes, and Techniques," Vol. 1, U.S. Air Force, 2013.

[8] Krishnamoorthy, C., and Rajeev, S. *Artificial Intelligence and Expert Systems for Engineers*, Vol. 11, CRC press, 1996.

[9] Suri, R., and Shimizu, M. "Design for Analysis: A New Strategy to Improve the Design Process," *Research in Engineering Design*, Vol. 1, 1989, pp. 105–120. doi: 10.1007/BF01580204

[10] Heinze, W. "Ein Beitrag Zur Quantitativen Analyse Der Technischen Und Wirtschaftlichen Auslegungsgrenzen Verschiedener Flugzeugkonzepte Fur Den Transport Grosser Nutzlasten." TU Braunschweig, ZLR Forschungsbericht, 1994.

[11] Roskam, J. *Airplane Design Part VIII: Airplane Cost Estimation: Design, Development, Manufacturing and Operating*, DARcorporation, 1990.

[12] de Weck, O. "16.842 Fundamentals of Systems Engineering." Massachusetts Institute of Technology: MIT OpenCourseWare, https://ocw.mit.edu., Fall 2015.

[13] Torenbeek, E. "Synthesis of Subsonic Airplane Design," *Delft: Springer*, 1982.

[14] Nicolai, L. M., and Carichner, G. *Fundamentals of Aircraft and Airship Design*, *AIAA Educational Series*, Vol. 1, American Institute of Aeronautics and Astronautics, Reston, VA, 2010. doi: 10.2514/4.867538

[15] Striz, A., Kennedy, B., Siddique, Z., and Neeman, H. "A Roadmap for Moderate Fidelity Conceptual Design with Multilevel Analysis and MDO," *47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, AIAA 2006-1619, 2006. doi: 10.2514/6.2006-1619

[16] Chun-Lin, G., Liang-Xian, G., Heng-Jun, L., and Jian-Ke, S. "KE and MDO Based Intelligent Conceptual Design Method for Tactical Missile," *2010 IEEE International Conference on Intelligent Computing and Intelligent Systems*, 2010. doi: 10.1109/ICICISYS.2010.5658668

[17] Chudoba, B., and Heinze, W. "Evolution of Generic Flight Vehicle Design Synthesis," *The Aeronautical Journal*, Vol. 114, No. 1159, 2010, pp. 549–567.

[18] Chudoba, B. *Stability and Control of Conventional and Unconventional Aircraft Configurations: A Generic Approach*, BoD–Books on Demand, 2001.

[19] Chudoba, B. "Managerial Implications of Generic Flight Vehicle Design Synthesis," *44th AIAA Aerospace Sciences Meeting and Exhibit*, AIAA-2006-1178, 2006. doi: 10.2514/6.2006-1178

[20] Thurstone, L. L. *Primary Mental Abilities.*, Chicago, 1938.

[21] Gardner, H. *Frames of Mind: The Theory of Multiple Intelligences*, Basic books, 1993.

[22] Salovey, P., and Mayer, J. D. "Emotional Intelligence," *Imagination, Cognition and Personality*, Vol. 9, No. 3, 1990, pp. 185–211.

[23] Goleman, D. P. *Emotional Intelligence: Why It Can Matter More Than IQ*, New York: Bantam Books, 2006.

[24] Sternberg, R. J. *Beyond IQ: A Triarchic Theory of Human Intelligence*, Cambridge University Press, 1985.

[25] Krishnakumar, K. "Intelligent Systems for Aerospace Engineering-an Overview," 2003.

[26] Harrison, L., Saunders, P., and Janowitz, J. "Artificial Intelligence with Applications for Aircraft.," 1994.

[27] Munakata, T. *Fundamentals of the New Artificial Intelligence Neural, Evolutionary, Fuzzy and More*, 2 ed., Springer, 2008. doi: 10.1007/978-1-84628-839-5

[28] Russell, S. J., Norvig, P., Canny, J. F., Malik, J. M., and Edwards, D. D. *Artificial Intelligence: A Modern Approach*, 2 ed., Prentice Hall Upper Saddle River, 2003.

[29] Negnevitsky, M. *Artificial Intelligence: A Guide to Intelligent Systems*, Pearson Education, 2005.

[30] Hopgood, A. A. *Intelligent Systems for Engineers and Scientists*, CRC press, 2011.

[31] Noor, A. K. "Computational Intelligence and Its Impact on Future High-Performance Engineering Systems," NASA Conference Publication 3323, National Aeronautics and Space Administration, Langley Research Center, 1996.

[32] Blount, G. N., Kneebone, S., and Kingston, M. R. "Selection of Knowledge-Based Engineering Design Applications," *Journal of Engeering Design*, Vol. 6, No. 1, 1995, pp. 31-38. doi: 10.1080/09544829508907900

[33] Hopgood, A. A. "The State of Artificial Intelligence," *Advances in Computers*, Vol. 65, 2005, pp. 1 - 75.

[34] Marx, W. J., Schrage, D. P., and Mavris, D. N. "An Application of Artificial Intelligence for Computer-Aided Design and Manufacturing," *International Conference on Computational Engineering Science; Supercomputting in Multidisciplinary Analysis and Design*, 1995.

[35] Karppinen, N., Lucas, A., Ljungberg, M., and Repusseau, P. "Artificial Intelligence in Air Traffic Flow Management," *Proceedings of the International Aerospace Conference*, Technical Note 16, Melbourne, Australia, 1991.

[36] Weigang, L., Alves, C. J. P., and Omar, N. "An Expert System for Air Traffic Flow Management," *Journal of Advanced Transportation*, Vol. 31, No. 3, 1997, pp. 343–361.

[37] Ryan, J. C., Cummings, M., Roy, N., Banerjee, A., and Schulte, A. "Designing an Interactive Local and Global Decision Support System for Aircraft Carrier Deck Scheduling," *Infotech@ Aerospace 2011*, AIAA 2011-1516, 2011. doi: 10.2514/6.2011-1516

[38] Richards, R. "Application of Multiple Artificial Intelligence Techniques for an Aircraft Carrier Landing Decision Support Tool," *2002 IEEE World Congress on Computational Intelligence. 2002 IEEE International Conference on Fuzzy Systems. FUZZ-IEEE'02. Proceedings (Cat. No.02CH37291)*, Vol. 1, 2002, pp. 7–11. doi: 10.1109/FUZZ.2002.1004950

[39] Roy, A. G., and Peyada, N. "Aircraft Parameter Estimation Using Hybrid Neuro Fuzzy and Artificial Bee Colony Optimization (HNFABC) Algorithm," *Aerospace Science and Technology*, Vol. 71, 2017, pp. 772–782. doi: 10.1016/j.ast.2017.10.030

[40] Lala Jr, L. L., Wood, L. H., and Perrotta, C. D. "Intelligent Systems for Space Situational Awareness," *Infotech@Aerospace 2011*, AIAA 2011-1433, 2011. doi: 10.2514/6.2011-1433

[41] Calise, A. J. "Neural Networks in Nonlinear Aircraft Flight Control," *Aerospace and Electronic Systems Magazine, IEEE*, Vol. 11, No. 7, 1996, pp. 5–10.

[42] Butyrin, S. A., Makarov, V., Mukumov, R., Somov, Y., and Vassilyev, S. "An Expert System for Design of Spacecraft Attitude Control Systems," *Artificial Intelligence in Engineering*, Vol. 11, No. 1, 1997, pp. 49–59.

[43] La Rocca, G. "Knowledge Based Engineering: Between AI and CAD. Review of a Language Based Technology to Support Engineering Design," *Advanced Engineering Informatics*, Vol. 26, No. 2, 2012, pp. 159–179. doi: 10.1016/j.aei.2012.02.002

[44] Amadori, K. "Geometry Based Design Automation: Applied to Aircraft Modelling and Optimization," PhD Dissertation, Linköping University, 2012.

[45] La Rocca, G., and van Tooren, M. J. L. "A Knowledge Based Engineering Approach to Support Automatic Generation of FE Models in Aircraft Design," *45th AIAA Aerospace Sciences Meeting and Exhibit*, AIAA 2007-967, 2007. doi: 10.2514/6.2007-967

[46] Ledermann, C., Hanske, C., Wenzel, J., Ermanni, P., and Kelm, R. "Associative Parametric CAE Methods in the Aircraft Pre-Design," *Aerospace Science and Technology*, Vol. 9, No. 7, 2005, pp. 641–651.

[47] Rentema, D. "AIDA. Artificial Intelligence Supported Conceptual Design of Aircraft," PhD Dissertation, Delft University of Technology, 2004.

[48] Tsuchiya, T., Takenaka, Y., and Taguchi, H. "Multidisciplinary Design Optimization for Hypersonic Experimental Vehicle," *AIAA Journal*, Vol. 45, No. 7, 2007, pp. 1655–1662.

[49] Tianyuan, H., and Xiongqing, Y. "Aerodynamic/Stealthy/Structural Multidisciplinary Design Optimization of Unmanned Combat Air Vehicle," *Chinese Journal of Aeronautics*, Vol. 22, No. 4, 2009, pp. 380–386.

[50] Rao, C., Tsai, H., and Ray, T. "Aircraft Configuration Design Using a Multidisciplinary Optimization Approach," *42nd AIAA Aerospace Sciences Meeting and Exhibit*, AIAA 2004-536, 2004. doi: 10.2514/6.2004-536

[51] Shahrokhi, A., and Jahangirian, A. "Airfoil Shape Parameterization for Optimum Navier–Stokes Design with Genetic Algorithm," *Aerospace Science and Technology*, Vol. 11, No. 6, 2007, pp. 443–450.

[52] Oyama, A., Nonomura, T., and Fujii, K. "Data Mining of Pareto-Optimal Transonic Airfoil Shapes Using Proper Orthogonal Decomposition," *Journal of Aircraft*, Vol. 47, No. 5, 2010, pp. 1756–1762.

[53] Alemany, K., and Braun, R. D. "Survey of Global Optimization Methods for Low-Thrust, Multiple Asteroid Tour Missions," *AAS/AIAA Space Flight Mechanics Meeting*, AAS 07-211, 2007.

[54] Zotes, F. A. "Application of Intelligent Algorithms to Aerospace Problems," PhD Dissertation, Universidad Nacional de Educación a Distancia, 2011.

[55] Chae, H. G. "A Possibilistic Approach to Rotorcraft Design through a Multi-Objective Evolutionary Algorithm," PhD Dissertation, Georgia Institute of Technology, 2006.

[56] Neufeld, D., and Chung, J. "Unmanned Aerial Vehicle Conceptual Design Using a Genetic Algorithm and Data Mining," *Infotech@Aerospoace*, AIAA 2005-7051, 2005. doi: 10.2514/6.2005-7051

[57] Damp, L., Gonzalez, L. F., and Srinivas, K. "Multi-Objective and Multidisciplinary Design Optimisation (MDO) of UAV Systems Using Hierarchical Asynchronous Parallel Evolutionary Algorithms," University of Sydney, School of Aerosapce, Mechanical, and Mechatronic Engineering, 2007.

[58] Lee, D., Gonzalez, L. F., Srinivas, K., Auld, D., and Wong, K. C. "Aerodynamic/RCS Shape Optimisation of Unmanned Aerial Vehicles Using Hierarchical Asynchronous Parallel Evolutionary Algorithms," *24th AIAA Applied Aerodynamics Conference*, AIAA 2006-3331, 2006.

[59] Jones, B. R., Crossley, W. A., and Lyrintzis, A. S. "Aerodynamic and Aeroacoustic Optimization of Rotorcraft Airfoils Via a Parallel Genetic Algorithm," *Journal of Aircraft*, Vol. 37, No. 6, 2000, pp. 1088–1096.

[60] Carrese, R., Winarto, H., and Li, X. "Integrating User-Preference Swarm Algorithm and Surrogate Modeling for Airfoil Design," *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, AIAA 2011-1246, 2011.

[61] Khurana, M. S., Winarto, H., and Sinha, A. K. "Application of Swarm Approach and Artificial Neural Networks for Airfoil Shape Optimization," *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, AIAA 2008-5954, 2008.

[62] Rajagopal, S., and Ganguli, R. "Multidisciplinary Design Optimization of a UAV Wing Using Kriging Based Multi-Objective Genetic Algorithm," *50th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, AIAA 2009-2219, 2009.

[63] Viviani, A., Iuspa, L., and Aprovitola, A. "An Optimization-Based Procedure for Self-Generation of Re-Entry Vehicles Shape," *Aerospace Science and Technology*, Vol. 68, 2017, pp. 123–134.

[64] Bayley, D. J. "Design Optimization of Space Launch Vehivcles Using a Genetic Algorithm," PhD Dissertation, Auburn University, 2007.

[65] Mosher, T. "Conceptual Spacecraft Design Using a Genetic Algorithm Trade Selection Process," *Journal of Aircraft*, Vol. 36, No. 1, 1999, pp. 200–208.

[66] Huang, G., Lu, Y., and Nan, Y. "A Survey of Numerical Algorithms for Trajectory Optimization of Flight Vehicles," *Science China Technological Sciences*, Vol. 55, No. 9, 2012, pp. 2538–2560. doi: 10.1007/s11431-012-4946-y

[67] Oyama, A., Kawakatsu, Y., and Hagiwara, K. "Data Mining of Pareto-Optimal Solutions of a Solar Observatory Trajectory Design Problem," *Infotech@Aerospace*, AIAA 2012-2442, 2012.

[68] Chiba, K., Obayashi, S., and Morino, H. "Knowledge Discovery for Transonic Regional-Jet Wing through Multidisciplinary Design Exploration," *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, Vol. 2, No. 3, 2008, pp. 396–407.

[69] Chiba, K., Jeong, S., Obayashi, S., and Yamamoto, K. "Knowledge Discovery in Aerodynamic Design Space for Flyback–Booster Wing Using Data Mining," *14th AIAA/AHI Space Planes and Hypersonic System and Technologies Conference*, AIAA 2006-7992, 2006.

[70] Berke, L., Patnaik, S., and Murthy, P. "Optimum Design of Aerospace Structural Components Using Neural Networks," *Computers & Structures*, Vol. 48, No. 6, 1993, pp. 1001–1010.

[71] Khlopkov, Y. I., Dorofeev, E. A., Myint, Z. Y. M., Khlopkov, A. Y., Polyakov, M. S., and Agayeva, I. R. k. "Application of Artificial Neural Networks in Hypersonic Aerospace System," *Applied Mathematical Sciences*, Vol. 8, No. 95, 2014, pp. 4729 - 4735. doi: 10.12988/ams.2014.46494

[72] Rocca, G. L. "Knowledge Based Engineering Techniques to Support Aircraft Design and Optimization," PhD Dissertation, Delft University of Technology, 2011.

[73] Soulat, M. E. "Parametric Geometry Representation to Support Aircraft Design," *IEEE Aerospace Conference*, IEEEAC Paper 1745, Version 2, IEEE, 2012.

[74] van der Laan, A. H. "Knowledge Based Engineering Support for Aircraft Component Design," PhD Dissertation, Delft University of Technology, 2008.

[75] Hoogreef, M., and La Rocca, G. "An MDO Advisory System Supported by Knowledge-Based Technologies," *16th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, AIAA 2015-2945, 2015. doi: 10.2514/6.2015-2945

[76] Price, A. R., Keane, A. J., and E. Holden, C. M. "On the Coordination of Multidisciplinary Design Optimization Using Expert Systems," *AIAA Journal*, Vol. 49, No. 8, 2011, pp. 1778–1794.

[77] Antoni Virosi, i. M., and Selva, D. "Daphne: A Virtual Assistant for Designing Earth Observation Distributed Spacecraft Missions," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, Vol. 13, 2020, pp. 30-48.

[78] Velez, V. H. "Data Mining and Knowledge Discover-IBM Cognitive Alternatives for NASA KSC," 2016.

[79] Berquand, A., and Riccardi, A. "The Design Engineering Assistant Applying Ontology Learning to the Generation of a Space Mission Ontology," *Space Systems Ontology Brainstorming Workshop*, 2019.

[80] Martin, A. V., and Selva, D. "From Design Assistants to Design Peers: Turning Daphne into an AI Companion for Mission Designers," *AIAA Scitech 2019 Forum*, 2019. doi: 10.2514/6.2019-0402

[81] Morrison, J. H., Ambur, M. Y., and Bauer, S. X. "Comprehensive Digital Transformation NASA Langley Research Center," 2016.

[82] Ambur, M. Y., Yagle, J. J., Reith, W., and McLarney, E. "Big Data Analytics and Machine Intelligence Capability Development at NASA Langley Research Center: Strategy, Roadmap, and Progress," *NASA/TM-2016-219361*, 2016.

[83] Lintern, G. "What Is Cognitive System?," *2007 International Symposium on Aviation Psychology*, 2007.

[84] Huang, X. "A Prototype Computerized Synthesis Methodology for Generic Space Access Vehicle (SAV) Conceptual Designs," PhD Dissertation, University of Oklahoma, 2006.

[85] Coleman, G. "Aircraft Conceptual Design - an Adaptable Parametric Sizing Methodology," PhD Dissertation, The University of Texas At Arlington, 2010.

[86] Gonzalez, L. "Complex Multidisciplinary Systemm Composition for Aerospace Vehicle Conceptual Design," PhD Dissertation, The University of Texas at Arlington, 2016.

[87] Omoragbon, A. "Complex Multidisciplinary Systems Decomposition for Aerospace Vehicle Conceptual Design and Technology Acquisition," PhD Dissertation, The University of Texaas at Arlington, 2016.

[88] Coley, M. D. "On Space Program Planning Quantifying the Effects of Spacefaring Goals and Strategies on the Solution Space of Feasible Programs," PhD Dissertation, The University of Texas at Arlington, 2017.

[89] Oza, A. "A Portfolio-Based Approach to Evaluate Aerospace R&D Problem Formulation Towards Parametric Synthesis Tool Design," PhD Dissertation, The University of Texas at Arlington, 2016.

[90] Peng, X. "Formalization of the Engineering Science Discipline-Knowledge Engineering," PhD Dissertation, The University of Texas at Arlington, 2015.

[91] Haney, E. "Data Engineering in Aerospace Systems Design & Forecasting," PhD Dissertation, The Univeristy of Texas at Arlington, 2016.

[92] La Rocca, G., and van Tooren, M. J. L. "Knowledge-Based Engineering to Support Aircraft Multidisciplinary Design and Optimization," *26th International Congress of the Aeronautical Sciences*, 2008.

[93] Corning, G. *Supersonic and Subsonic, CTOL and VTOL, Airplane Design*, 4 ed., College Park, MD, 1976.

[94] Wood, K. D. *Aerospace Vehicle Design Vol. 1, Aircraft Design*, Johnson Publishing Company, Boulder, CO, 1963.

[95] Loftin Jr, L. K. "Subsonic Aircraft: Evolution and the Matching of Size to Performance," NASA RP-1060, Scientific and Technical Information Branch, NASA, Hampton, VA, 1980.

[96] Roskam, J. *Airplane Design*, Vol. 1-8, DARcorporation, Ottawa, KS, 1990.

[97] Raymer, D. P. *Aircraft Design: A Conceptual Approach*, *AIAA Education Series*, 3 ed., American Institute of Aeronautics and Astronautics, Reston, VA, 2018. doi: 10.2514/4.104909

[98] Stinton, D. *The Anatomy of the Airplane*, 2 ed., John Wiley and Sons Ltd, Oxford, U.K., 1998. doi: 10.2514/4.475146

[99] Anderson, J. D. *Aircraft Performance and Design*, 1999.

[100] Fielding, J. P. *Introduction to Aircraft Design*, 2 ed., Cambridge University Press, 2017.

[101] Jenkinson, L. R., Rhodes, D., and Simpkin, P. *Civil Jet Aircraft Design*, *AIAA Education Series*, Norwich, NY, 2006.

[102] Howe, D. *Aircraft Conceptual Design Synthesis*, *Aerospace Series*, Wiley, Hoboken, NJ, 2005.

[103] Schaufele, R. D. *The Elements of Aircraft Preliminary Design*, Aries Publications, Santa Ana, CA, 2007.

[104] Sadraey, M. H. *Aircraft Design: A Systems Engineering Approach*, John Wiley & Sons, 2012.

[105] Gudmundsson, S. *General Aviation Aircraft Design: Applied Methods and Procedures*, Butterworth-Heinemann, 2013.

[106] Sforza, P. M. *Commercial Airplane Design Principles*, Elsevier, 2014.

[107] Kundu, A. K., Price, M. A., and Riordan, D. *Conceptual Aircraft Design: An Industrial Perspective*, *Aerospace Series*, Wiley-Blackwell, Hoboken, NJ, 2018.

[108] White, J. F. *Flight Performance Handbook for Powered Flight Operations: Flight Mechanics and Space Vehicle Design, Empirical Formulae, Analytic Approximations, and Graphical Aids*, Wiley, New York, 1963.

[109] Wood, K. D. *Aerospace Vehicle Design: Spacecraft Design*, Vol. 2, Johnson Publishing Company, Boulder, CO, 1964.

[110] Harney, E. D. *Space Planners Guide,* Air Force Systems Command, United States Air Force, Andrews Air Force Base, Washington, D.C., 1965.

[111] Humble, R. W., Henry, G. N., and Larson, W. J. *Space Propulsion Analysis and Design*, McGraw-Hill, New York, 2007.

[112] Logsdon, T. *Orbital Mechanics: Theory and Applications,* John Wiley & Sons, New York, 1998.

[113] Hammond, W. E. *Design Methodologies for Space Transportation Systems, AIAA Education Series*, American Institute of Aeronautics and Astronautics, Reston, VA, 2001.

[114] Suresh, B. N., and Sivan, K. *Integrated Design for Space Transportation System*, 1 ed., Springer, New Delhi, India, 2015. doi: 10.1007/978-81-322-2532-4

[115] Sziroczák, D. "Conceptual Design Methodologies Appropriate to Hypersonic Space and Global Transportation Systems," PhD Dissertation, School of Aerospace, Transport and Manufacturing, Cranfield University, 2015.

[116] Rana, L., and Chudoba, B. "Demonstration of a Prototype Design Synthesis Capability for Space Access Vehicle Design," *The Aeronautical Journal*, Vol. 124, No. 1281, 2020, pp. 1761-1788. doi: 10.1017/aer.2020.55

[117] Sobieszczanski-Sobieski, J., and Haftka, R. T. "Multidisciplinary Aerospace Design Optimization: Survey of Recent Developments," *Structural Optimization*, Vol. 14, No. 1, 1997, pp. 1–23.

[118] Perez, R., Liu, H., and Behdinan, K. "Evaluation of Multidisciplinary Optimization Approaches for Aircraft Conceptual Design," *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, AIAA 2004-4537, 2004. doi: 10.2514/6.2004-4537

[119] Yao, W., Chen, X., Luo, W., van Tooren, M., and Guo, J. "Review of Uncertainty-Based Multidisciplinary Design Optimization Methods for Aerospace Vehicles," *Progress in Aerospace Sciences*, Vol. 47, No. 6, 2011, pp. 450–479.

[120] Simpson, T. W., Toropov, V., Balabanov, V., and Viana, F. A. C. "Design and Analysis of Computer Experiments in Multidisciplinary Design Optimization: A Review of How Far We Have Come or Not," *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, AIAA 2008-5802, 2008. doi: 10.2514/6.2008-5802

[121] Martins, J. R. R. A., and Lambe, A. B. "Multidisciplinary Design Optimization: A Survey of Architectures," *AIAA Journal*, Vol. 51, No. 9, 2013, pp. 2049-2075.

[122] Vandenbrande, J. H., Grandine, T. A., and Hogan, T. "The Search for the Perfect Body: Shape Control for Mulidisciplinary Design Optimization," *44th AIAA Sciences Meeting and Exhibit*, AIAA 2006-928, 2006.

[123] Rafique, A., LinShu, H., Zeeshan, Q., Nisar, K., and Xiaowei, W. "Hybrid Optimization Method for Multidisciplinary Design of Air Launched Satellite Launch Vehicle," *45th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, AIAA 2009-5535, 2009. doi: 10.2514/6.2009-5535

[124] Hajela, P. "Soft Computing in Multidisciplinary Aerospace Design - New Directions for Research," *Aerodynamic Design and Optimisation of Flight Vehicles in a Concurrent Multi-Disciplinary Environmen*, 1999.

[125] Sobieszczanski-Sobieski, J. "Multidisciplinary Design Optimization: An Emerging New Engineering Discipline," NASA Technical Memorandum 107761, 1995.

[126] Giesing, J., and Barthelemy, J.-F. "A Summary of Industry MDO Applications and Needs," *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 1998. doi: 10.2514/6.1998-4737

[127] Riccardi, A. "Multidisciplinary Design Optimization for Space Applications," PhD Dissertation, Universität Bremen, 2012.

[128] Frenzel, M., Heiserer, D., Keller, D., Schemat, M., Balabanov, V., Dreisbach, R., Georgiadis, S., and Trop, D. "Multidisciplinary Optimization and Integration Requirements for Large-Scale Automotive and Aerospace Design Work," *11th World Congress on Structural and Multidisciplinary Optimisation*, 2015.

[129] van Gent, I. "Agile MDAO Systems a Graph-Based Methodology to Enhance Collaborative Multidisciplinary Design," PhD Dissertation, Delft University of Technology, 2019.

[130] Berkeley AI Research. "Caffe," URL: https://caffe.berkeleyvision.org/ [retrieved December 21, 2020].

[131] Phoenix Integration. "Modelcenter," URL: https://www.phoenix-int.com/ [retrieved December 21, 2020].

[132] Sandia National Laboratories. "Dakota," URL: https://dakota.sandia.gov/ [retrieved December 21, 2020].

[133] Noesis Solutions. "Optimus," URL: https://www.noesissolutions.com/our-products/optimus [retrieved December 21, 2020].

[134] Esteco. "Modefrontier," URL: https://www.esteco.com/modefrontier [retrieved December 21, 2020].

[135] German Aerospace Center (DLR). "RCE," URL: https://rcenvironment.de/ [retrieved December 21, 2020].

[136] Bowcutt, K. G. "A Perspective on the Future of Aerospace Vehicle Design," *12th AIAA International Space Planes and Hypersonic Systems and Technologies*, AIAA 2003-6957, 2003. doi: 10.2514/6.2003-6957

[137] Schut, E. J. "Conceptual Design Automation Abstraction Complexity Reduction by Feasilisation and Knowledge Engineering," PhD Dissertation, Delft University of Technology, 2010.

[138] La Rocca, G., and van Tooren, M. J. "Knowledge-Based Engineering Approach to Support Aircraft Multidisciplinary Design and Optimization," *Journal of Aircraft*, Vol. 46, No. 6, 2009, pp. 1875–1885. doi: 10.2514/1.39028

[139] de Weck, O., Agte, J., Sobieszczanski-Sobieski, J., Arendsen, P., Morris, A., and Spieck, M. "State-of-the-Art and Future Trends in Multidisciplinary Design Optimization," *48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, AIAA 2007-1905, AIAA, 2007.

[140] Duffy, M., Chung, S. J., and Bergman, L. "An Evolutionary Architecture for the Automated Conceptual Design of Aerospace Systems," *Infotech@Aerospace 2011*, AIAA 2011-1632, 2011. doi: 10.2514/6.2011-1632

[141] Wang, C. "Insights from Developing a Multidisciplinary Design and Analysis Environment," *Computers in Industry*, Vol. 65, No. 4, 2014, pp. 786–795.

[142] Antoine, N. E., Kroo, I. M., Willox, K., and Barter, G. "A Framework for Aircraft Conceptual Design and Environmental Performance Studies," *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, AIAA 2004-4314, 2004. doi: 10.2514/6.2004-4314

[143] Lundström, D. "Aircraft Design Automation and Subscale Testing: With Special Reference to Micro Air Vehicles," PhD Dissertation, Linköping University, 2012.

[144] Anon. "CJCSI 3170.01E Joint Capabilities Integration and Development System," *Chairman of the Joint Cheifs of Staff Instructions*, 2005.

[145] Anon. "Adoption of ISO/IEC 15288:2002, Systems Engineering-System Life Cycle Processes," *IEEE Std 15288-2004 (Adoption of ISO/IEC Std 15288:2002)*, 2005. doi: 10.1109/IEEESTD.2005.96287

[146] Combs, H. G., Campbell, D. H., Cassidy, M. D., Sumpter, C. D., Seitz, E., Kachel, B. J., James, R. P., Walters, J., Love, J., and Passon, R. T. "Configuration Development Study

of the X-24C Hypersonic Research Airplane Executive Summary," NASA-CR-145274, NASA Langley Research Center, 1977.

[147] Ruttle, B., Stork, J., and Liston, G. "Generic Hypersonic Vehicles for Conceptual Design Analyses," AFRL/RQHT, Wright-Patterson AFB OH, 2012.

[148] Mutzman, R., and Murphy, S. "X-51 Development: A Chief Engineer's Perspective," *17th AIAA International Space Planes and Hypersonic Systems and Technologies Conference*, 2011.

[149] Czysz, P., and Vandenkerckhove, J. "Transatmospheric Launcher Sizing," *Scramjet Propulsion*, *Progress in Astronautics and Aeronautics*, edited by Murthy, S. N. B., and Curran, E. T., 2000, pp. 979-1103.

[150] Chudoba, B. "MAE 4351 Aerospace Vehicle Design II Design Project Next Generation Military Spacecraft," *MAE 4351 Course Notes*. The University of Texas at Arlington, 2018.

[151] McDonnell Aircraft Company. "Hypersonic Research Facilities Study. Volume II Part 2 Phase I Preliminary Studies Flight Vehicle Synthesis," NASA, 1970.

[152] Sutton, G. P., and Biblarz, O. *Rocket Propulsion Elements*, 8 ed., John Wiley & Sons, 2010.

[153] Heiser, W., Pratt, D., Daley, D., and Mehta, U. *Hypersonic Airbreathing Propulsion*, American Institute of Aeronautics and Astronautics, Inc., 1994. doi: 10.2514/4.470356

[154] Miele, A. *Flight Mechanics: Volume 1: Theory of Flight Paths*, Addison-Wesley, 1962.

[155] Harloff, G. J., and Berkowitz, B. M. "HASA, Hypersonic Aerospace Sizing Analysis, for the Preliminary Design of Aerospace Vehicles," NASA-Contractor Report 182226, NASA, 1988.

[156] Hank, J., Murphy, J., and Mutzman, R. "The X-51A Scramjet Engine Flight Demonstration Program," *15th AIAA International Space Planes and Hypersonic Systems and Technologies Conference*, *HYTSAP-8: Program Overview I*, AIAA 2008-2540, 2008. doi: 10.2514/6.2008-2540

[157] Anon. "X-51A Waverider," U.S. Air Force, 2011, URL: https://www.af.mil/About-Us/Fact-Sheets/Display/Article/104467/x-51a-waverider/ [retrieved 4/20/2020].

[158] White, M., and Price, W. "Affordable Hypersonic Missiles for Long-Range Precision Strike," *Johns Hopkins APL Technical Digest*, Vol. 20, No. 3, 1999, pp. 415-423.

[159] Dolvin, D. "High Speed Flight Research Insight Briefing," *USAF AFOSR Industry Program Review*, Basic Research Innovation and Collaboration Center (BRICC), Arlington VA, 2016.

[160] Bowcutt, K. G., Dolvin, D., Paull, A., and Smart, M. "HIFiRE: An International Collaboration to Advance the Science and Technology of Hypersonic Flight," *ICAS 2012 Congress*, Brisbane, Australia, 2012.

Appendix A   Case Studies Expanded

The following section provides additional documentation of the case studies ran. The order of presentation follows the case studies—verification then trade study.

A.1 All Body: X-51A

A.1.1 Vehicle Description

The X-51A is a hypersonic scramjet powered demonstrator vehicle. The Boeing Company and Pratt & Whitney Rocketdyne developed the vehicle for the US Air Force and DARPA. The X-51A is a waverider concept designed for Mach 6+ flight and a powered flight time of 240 seconds. The vehicle has no onboard subsonic propulsion and, as such, is accelerated by a booster rocket to the engine start condition. The X-51A is launched from a carrier vehicle, the B-52H. The X-51A vehicle and stack is illustrated in Figure A-1. The vehicle is a spatula nosed concept derived from the Affordable Rapid Response Missile Demonstrator (ARRMD) program[156]. On observation, the configuration shares many similarities to the legacy lifting bodies FDL-7 and McDonnell Douglas MRS. The vehicle is non-recoverable and is destroyed on mission completion. Four flight vehicles were built and flown; the X-51A conducted its first scramjet powered flight on May 26, 2010 and its last test flight on May 1, 2013 with a peak speed of Mach 5.1.[157]. For additional discussion on the X-51A see [148, 156, 157]. For a discussion on the ARRMD program see [158].

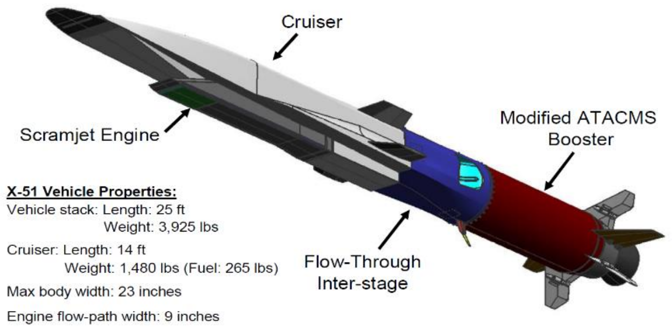


Figure A-1 X-51 configuration and full stack [156]

A.1.2 Vehicle Weights

The vehicles' primary geometric parameters and weight breakdown are given in Table A-1 and Table A-2 respectively.

Table A-1 X-51A primary dimensions, adapted from [148]

	in	**m**
AVD Stack Length	301	7.645
Cruiser Length	168	4.267
Max Body Width	23	0.584
Engine Flow-Path Width	9	0.229

Table A-2 X-51A Primary Mass Properties, adapted from [148]

	lb_m	**kg**	**N**
Cruiser Operating Weight	1225	556	5449
Cruiser Launch Weight	1504	682	6690
JP-7 Fuel (Useable)	265	120	1179
Booster	2277	1033	10129
Interstage	160	73	712
Stack Gross Weight (Captive Carry)	3942	1788	17535

A.1.3 Notional Mission

The X-51A mission is an airdropped non-reusable mission. The notional mission is illustrated in Figure A-2. The mission is described by references [156, 157]. The mission is very similar to the GHV mission. The vehicle is released from a B-52H at approximately 49,500 ft. and Mach 0.8. Shortly after release, the rocket booster ignites and burns for about 35 seconds, performing an accelerating climb. The rocket accelerates the X-51A to approximately Mach 4.5+ and 60,000 ft. at which point the cruise vehicle separates. The X-51A coasts until scramjet ignition. After ignition, the vehicle accelerates under the power of the scramjet from approximately Mach 4.5 to the cruise condition of Mach 6 and 70,000+ ft. The scramjet continues operating until the usable fuel is consumed after which the engine shuts off and the vehicle descends, performing unpowered experimental flight. The vehicle is ditched in the ocean and is not recovered.

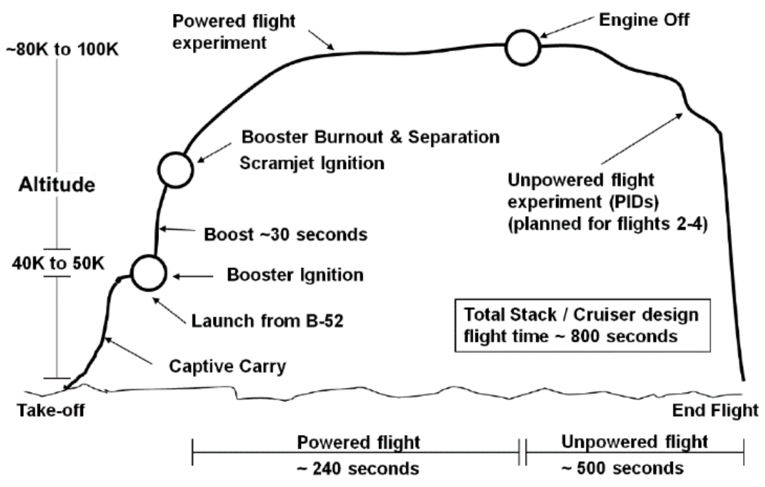


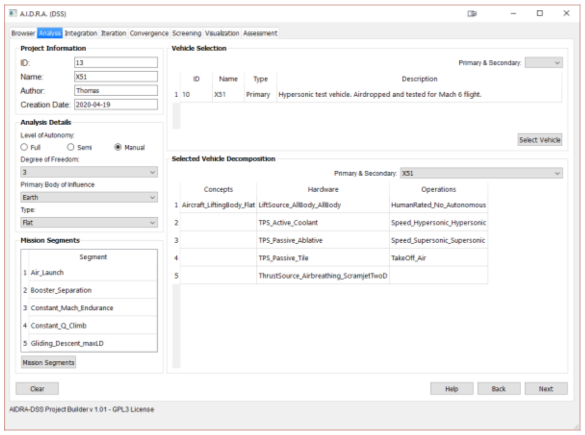
Figure A-2 X-51 notional mission[148]

Table A-3 Summary of X-51A notional mission profile

Mission Segment	Mach	Altitude		Dynamic Pressure	
		ft.	m	psf	kPa
Carrier Separation (booster ignition)	0.8	49,500	15,088	111	5
Initiation (booster separation)	4.5	60,000	18,288	2123	102
Acceleration and Climb	6	70,000	21,336	2336	112
Lifting Cruise	6	+70,000	21,336	2336	112

A.1.4 Project Builder Selections

This section documents the *Project Builder* selections. Depicted are each page's selections. The selection is organized according to the *Project Builder*'s tabs.


A.I.D.R.A. (DSS)
Browser Analysis Integration Iteration Convergence Screening Visualization Assessment
Select Vehicle
Mission Segments
Clear
Help
Back
Next

Project Information

ID:	13
Name:	X51
Author:	Thomas
Creation Date:	2020-04-19

Analysis Details

Level of Autonomy: Full Semi [✓ Manual]

Degree of Freedom: 3

Primary Body of Influence: Earth

Type: Flat

Mission Segments

	Segment
1	Air_Launch
2	Booster_Separation
3	Constant_Mach_Endurance
4	Constant_Q_Climb
5	Gliding_Descent_maxLD

Vehicle Selection

Primary & Secondary:

	ID	Name	Type	Description
1	10	X51	Primary	Hypersonic test vehicle. Airdropped and tested for Mach 6 flight.

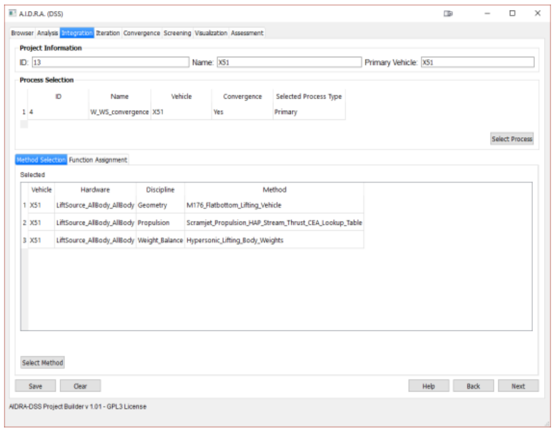
Selected Vehicle Decomposition

Primary & Secondary: X51

	Concepts	Hardware	Operations
1	Aircraft_LiftingBody_Flat	LiftSource_AllBody_AllBody	HumanRated_No_Autonomous
2		TPS_Active_Coolant	Speed_Hypersonic_Hypersonic
3		TPS_Passive_Ablative	Speed_Supersonic_Supersonic
4		TPS_Passive_Tile	TakeOff_Air
5		ThrustSource_Airbreathing_ScramjetTwoD	

AIDRA-DSS Project Builder v 1.01 - GPL3 License

Figure A-3 X-51A Analysis page


A.I.D.R.A. (DSS)
Browser Analysis Integration Iteration Convergence Screening Visualization Assessment
Select Process
Method Selection Function Assignment
Select Method
Save
Clear
Help
Back
Next

Project Information

ID: 13 Name: X51 Primary Vehicle: X51

Process Selection

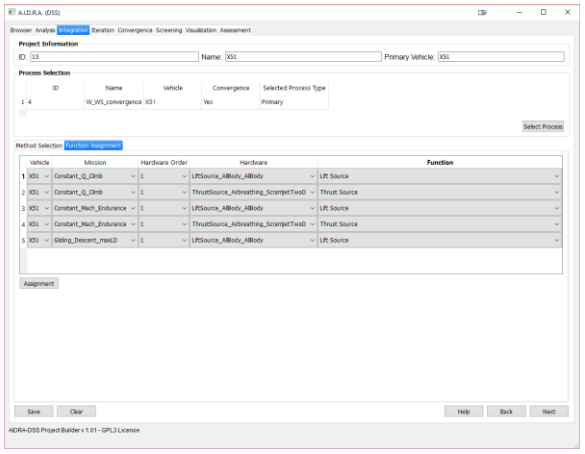
	ID	Name	Vehicle	Convergence	Selected Process Type
1	4	W_WS_convergence	X51	Yes	Primary

Selected

	Vehicle	Hardware	Discipline	Method
1	X51	LiftSource_AllBody_AllBody	Geometry	M176_Flatbottom_Lifting_Vehicle
2	X51	LiftSource_AllBody_AllBody	Propulsion	Scramjet_Propulsion_HAP_Stream_Thrust_CEA_Lookup_Table
3	X51	LiftSource_AllBody_AllBody	Weight_Balance	Hypersonic_Lifting_Body_Weights

AIDRA-DSS Project Builder v 1.01 - GPL3 License

Figure A-4 X-51A Integration page, (a) Method Selection


A.I.D.R.A. (DSS)
Browser Analysis Integration Iteration Convergence Screening Visualization Assessment
Select Process
Method Selection Function Assignment
Assignment
Save
Clear
Help
Back
Next

Project Information

ID: 13 Name: X51 Primary Vehicle: X51

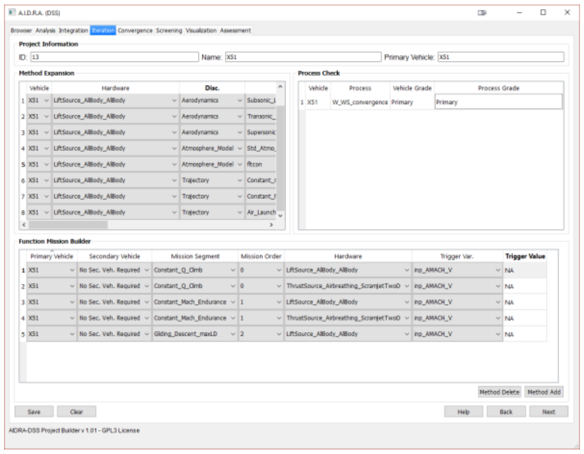
Process Selection

	ID	Name	Vehicle	Convergence	Selected Process Type
1	4	W_WS_convergence	X51	Yes	Primary

	Vehicle	Mission	Hardware Order	Hardware	Function
1	X51	Constant_Q_Climb	1	LiftSource_AllBody_AllBody	Lift Source
2	X51	Constant_Q_Climb	1	ThrustSource_Airbreathing_ScramjetTwoD	Thrust Source
3	X51	Constant_Mach_Endurance	1	LiftSource_AllBody_AllBody	Lift Source
4	X51	Constant_Mach_Endurance	1	ThrustSource_Airbreathing_ScramjetTwoD	Thrust Source
5	X51	Gliding_Descent_maxLD	1	LiftSource_AllBody_AllBody	Lift Source

AIDRA-DSS Project Builder v 1.01 - GPL3 License

Figure A-5 X-51A Integration page, (b) Function Assignment


A.I.D.R.A. (DSS)
Browser Analysis Integration Iteration Convergence Screening Visualization Assessment
Method Delete
Method Add
Save
Clear
Help
Back
Next

Project Information

ID: 13 Name: X51 Primary Vehicle: X51

Method Expansion

	Vehicle	Hardware	Disc.	
1	X51	LiftSource_AllBody_AllBody	Aerodynamics	Subsonic_L
2	X51	LiftSource_AllBody_AllBody	Aerodynamics	Transonic_
3	X51	LiftSource_AllBody_AllBody	Aerodynamics	Supersonic
4	X51	LiftSource_AllBody_AllBody	Atmosphere_Model	Std_Atmo_
5	X51	LiftSource_AllBody_AllBody	Atmosphere_Model	fltcon
6	X51	LiftSource_AllBody_AllBody	Trajectory	Constant_(
7	X51	LiftSource_AllBody_AllBody	Trajectory	Constant_I
8	X51	LiftSource_AllBody_AllBody	Trajectory	Air_Launch

Process Check

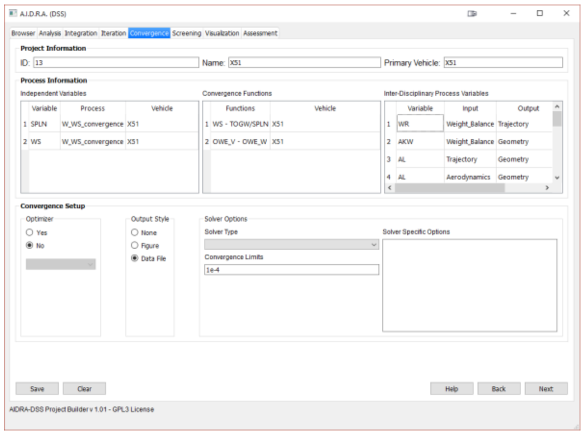
	Vehicle	Process	Vehicle Grade	Process Grade
1	X51	W_WS_convergence	Primary	Primary

Function Mission Builder

	Primary Vehicle	Secondary Vehicle	Mission Segment	Mission Order	Hardware	Trigger Var.	Trigger Value
1	X51	No Sec. Veh. Required	Constant_Q_Climb	0	LiftSource_AllBody_AllBody	inp_AMACH_V	NA
2	X51	No Sec. Veh. Required	Constant_Q_Climb	0	ThrustSource_Airbreathing_ScramjetTwoD	inp_AMACH_V	NA
3	X51	No Sec. Veh. Required	Constant_Mach_Endurance	1	LiftSource_AllBody_AllBody	inp_AMACH_V	NA
4	X51	No Sec. Veh. Required	Constant_Mach_Endurance	1	ThrustSource_Airbreathing_ScramjetTwoD	inp_AMACH_V	NA
5	X51	No Sec. Veh. Required	Gliding_Descent_maxLD	2	LiftSource_AllBody_AllBody	inp_AMACH_V	NA

AIDRA-DSS Project Builder v 1.01 - GPL3 License

Figure A-6 X-51A Iteration page


A.I.D.R.A. (DSS)
Browser Analysis Integration Iteration Convergence Screening Visualization Assessment
Save
Clear
Help
Back
Next

Project Information

ID: 13 Name: X51 Primary Vehicle: X51

Process Information

Independent Variables

	Variable	Process	Vehicle
1	SPLN	W_WS_convergence	X51
2	WS	W_WS_convergence	X51

Convergence Functions

	Functions	Vehicle
1	WS - TOGW/SPLN	X51
2	OWE_V - OWE_W	X51

Inter-Disciplinary Process Variables

	Variable	Input	Output
1	WR	Weight_Balance	Trajectory
2	AKW	Weight_Balance	Geometry
3	AL	Trajectory	Geometry
4	AL	Aerodynamics	Geometry

Convergence Setup

Optimizer: Yes [✓ No]

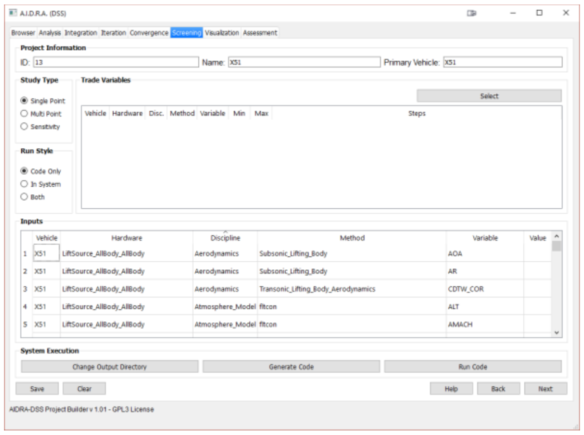
Output Style: None Figure [✓ Data File]

Solver Options — Solver Type: Solver Specific Options:

Convergence Limits: 1e-4

AIDRA-DSS Project Builder v 1.01 - GPL3 License

Figure A-7 X-51A Convergence page



Project Information

ID: 13 Name: X51 Primary Vehicle: X51

Study Type: [✓ Single Point] Multi Point Sensitivity

Run Style: [✓ Code Only] In System Both

Trade Variables

Vehicle	Hardware	Disc.	Method	Variable	Min	Max	Steps

Inputs

	Vehicle	Hardware	Discipline	Method	Variable	Value
1	X51	LiftSource_AllBody_AllBody	Aerodynamics	Subsonic_Lifting_Body	AOA	
2	X51	LiftSource_AllBody_AllBody	Aerodynamics	Subsonic_Lifting_Body	AR	
3	X51	LiftSource_AllBody_AllBody	Aerodynamics	Transonic_Lifting_Body_Aerodynamics	CDTW_COR	
4	X51	LiftSource_AllBody_AllBody	Atmosphere_Model	fltcon	ALT	
5	X51	LiftSource_AllBody_AllBody	Atmosphere_Model	fltcon	AMACH	

System Execution

AIDRA-DSS Project Builder v 1.01 - GPL3 License

Figure A-8 X-51A Screening page

A.1.5 *Trade Study*

The X-51A all-body concept, in addition to inspiration from the Model-176 and FDL-7, were used as baseline concepts for a trade study of all-body hypersonic cruiser vehicles. This section presents the trade study trade matrix, the convergence behavior and results, and the final consideration of the results pictorially in context to potential carrier vehicles.

A.1.5.1 *Trade Matrix*

The all-body concept evaluation evolved around two concepts—airbreathing and non-airbreathing. For each concept volume coefficient (τ), cruise time (t_{cruise}), and payload weight (W_{pay}) were traded. For the non-airbreathing cases, the fuel type was traded as well. The fuels traded were RP-1 and H_2. The trade ranges are shown in Table A-4.

Table A-4 All-Body trade matrix

Vehicle Tag	Baseline Vehicle	Propulsion System	Boost Type	Fuel Type	Tau Range	Payload (N)	Cruise Time (s)
AB2DS	X-51 / MODEL 176 Scram	2D Scramjet	External	JP-7	0.14 – 0.2143	0 – 4000	0 – 750
ABRKT	MODEL 176	Liquid Rocket	External	H_2 / RP-1	0.14 – 0.2143	0 – 5000	0 – 500
ABRKT	MODEL 176	Liquid Rocket	Internal	H_2 / RP-1	0.14 – 0.2143	0 – 5000	0 – 500

A.1.5.2 *Trade Convergence Summary: Iteration Errors*

The trades were executed as laid out. For reference, the final convergence error and convergence error per independent variable iteration for each trade are presented pictorially below. The order presented follows that of the row order in the trade matrix table. As can be seen, the convergence for the airbreathing cases occurred much more readily and with no noticeable difficulty. However, the problem solving became more laborious as the problem progressed through the non-airbreathing cases and specifically the integrated (internal boost) case. Note that for all trades, the all-body solutions converged, unlike the blended-body case.

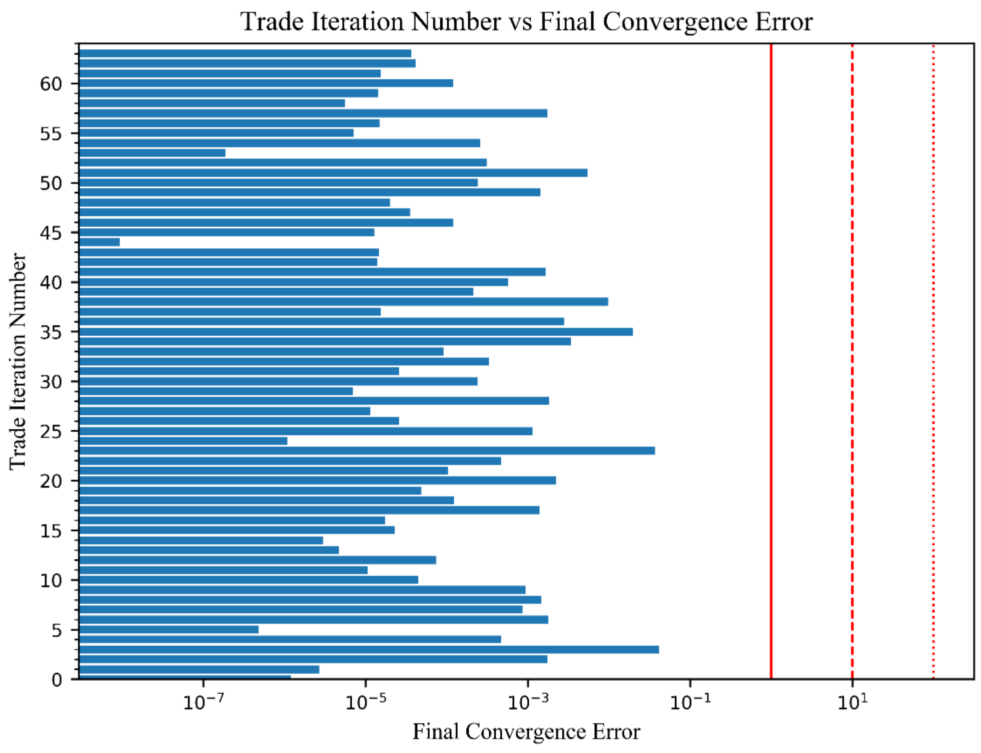


Figure A-9 AB scramjet iteration final convergence

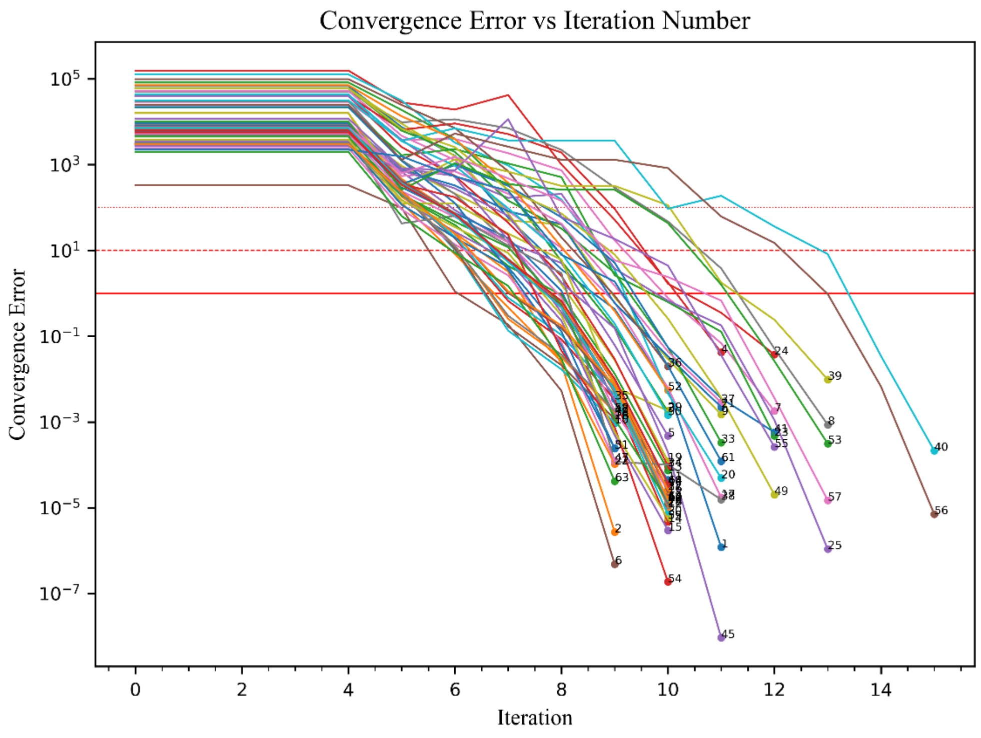


Figure A-10 AB scramjet iteration convergence error by step

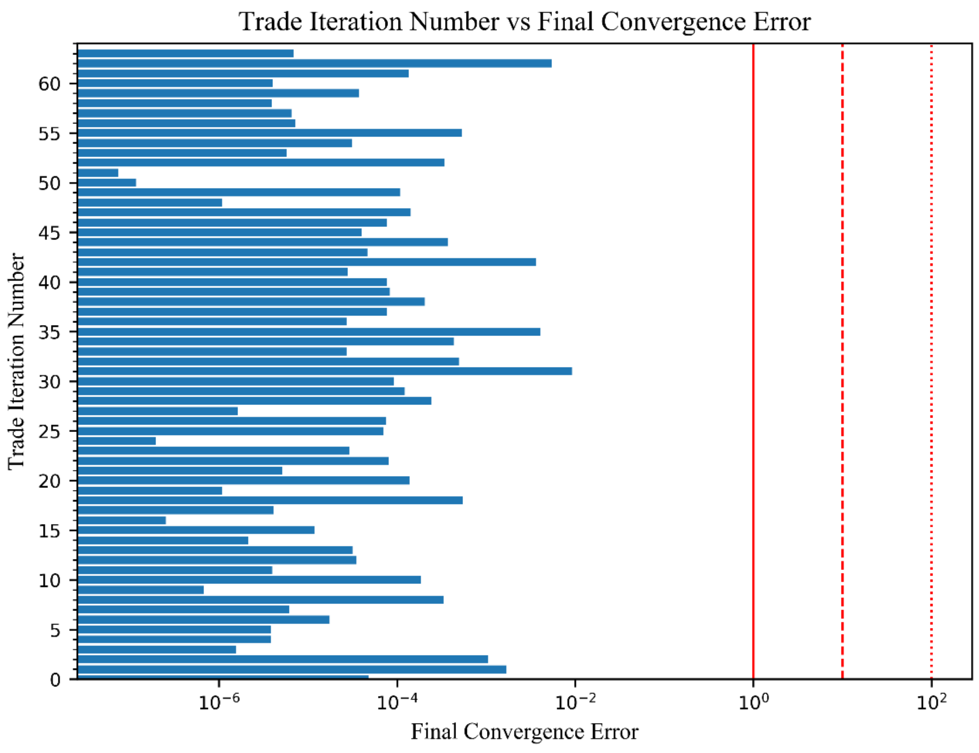


Figure A-11 AB boosted rocket (RP1) iteration final convergence

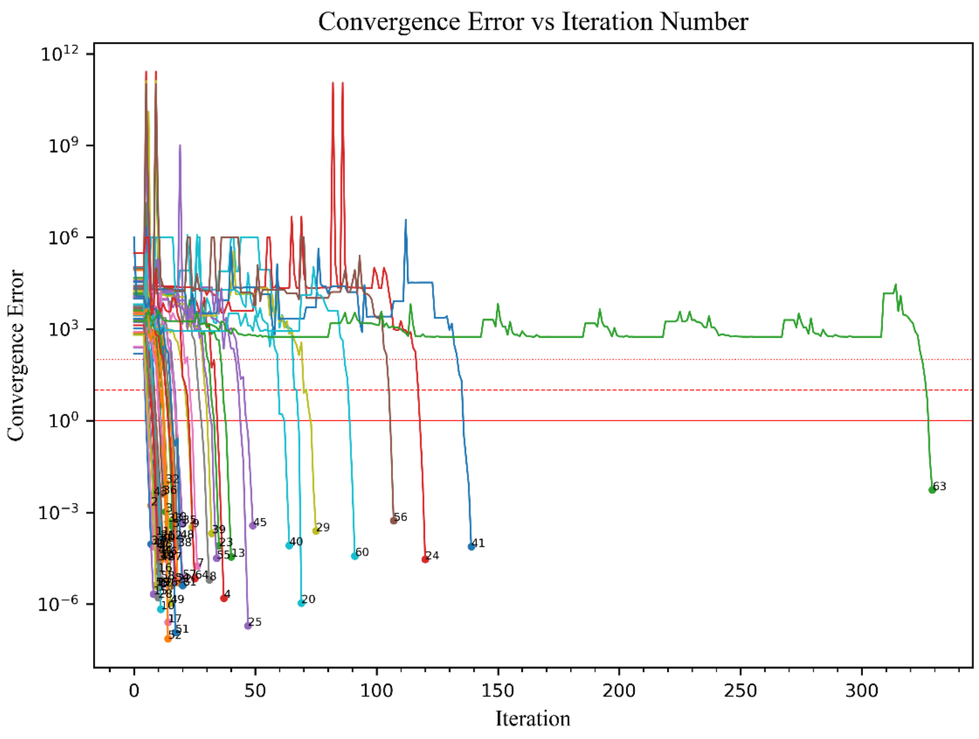


Figure A-12 AB boosted rocket (RP1) iteration convergence error by step

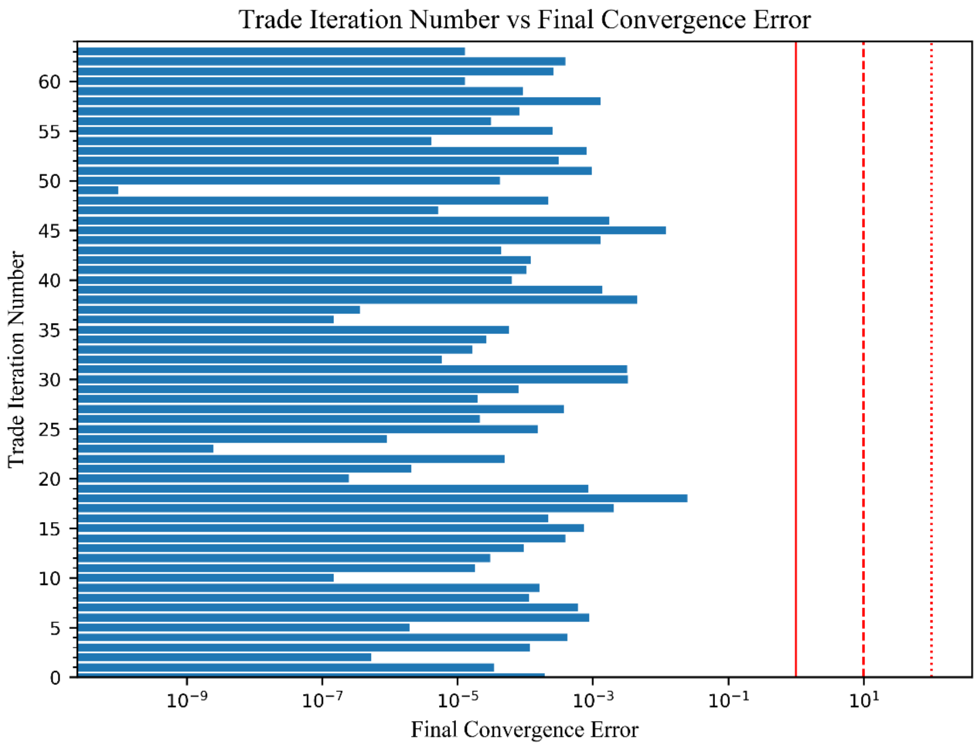


Figure A-13 AB boosted rocket (H_2) iteration final convergence

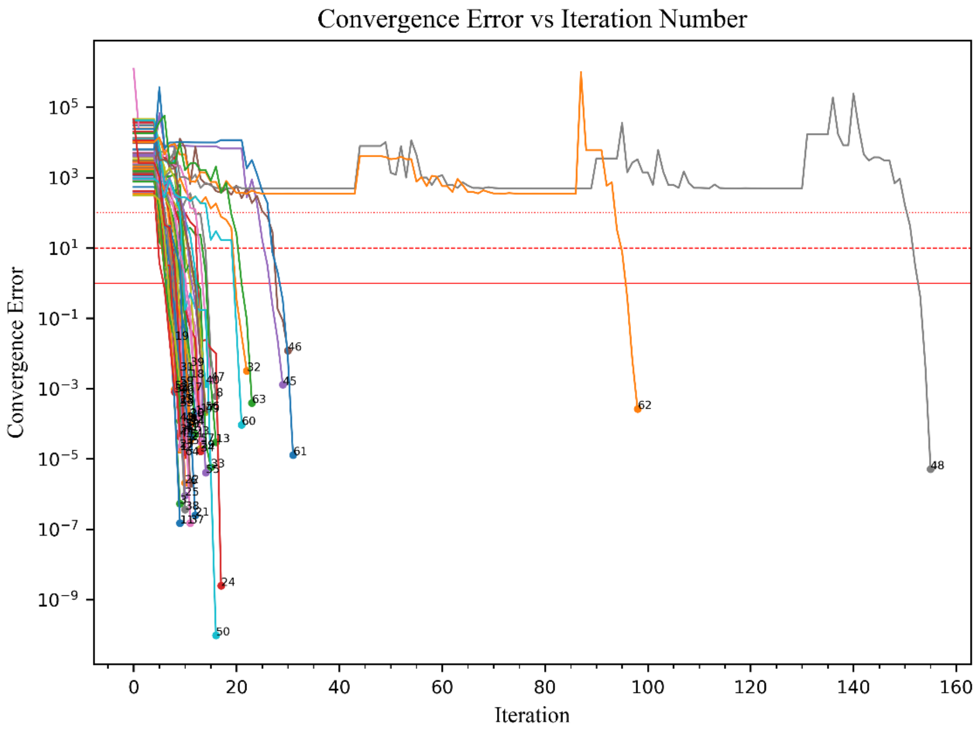


Figure A-14 AB boosted rocket (H_2) iteration convergence error by step

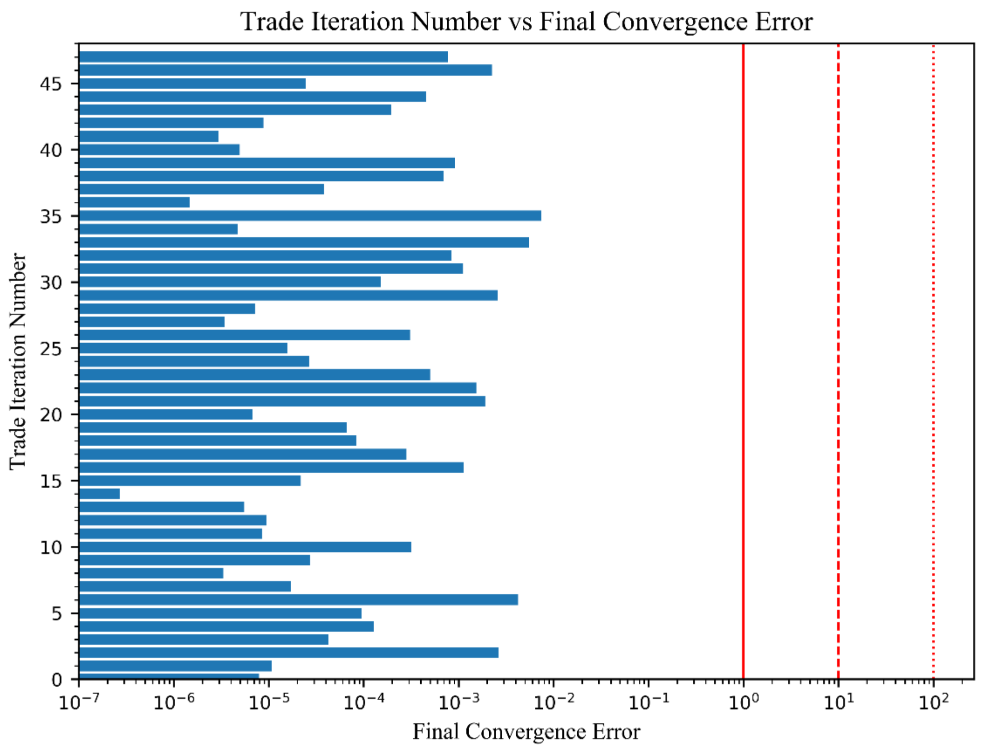


Figure A-15 AB integrated rocket (RP1) iteration final convergence

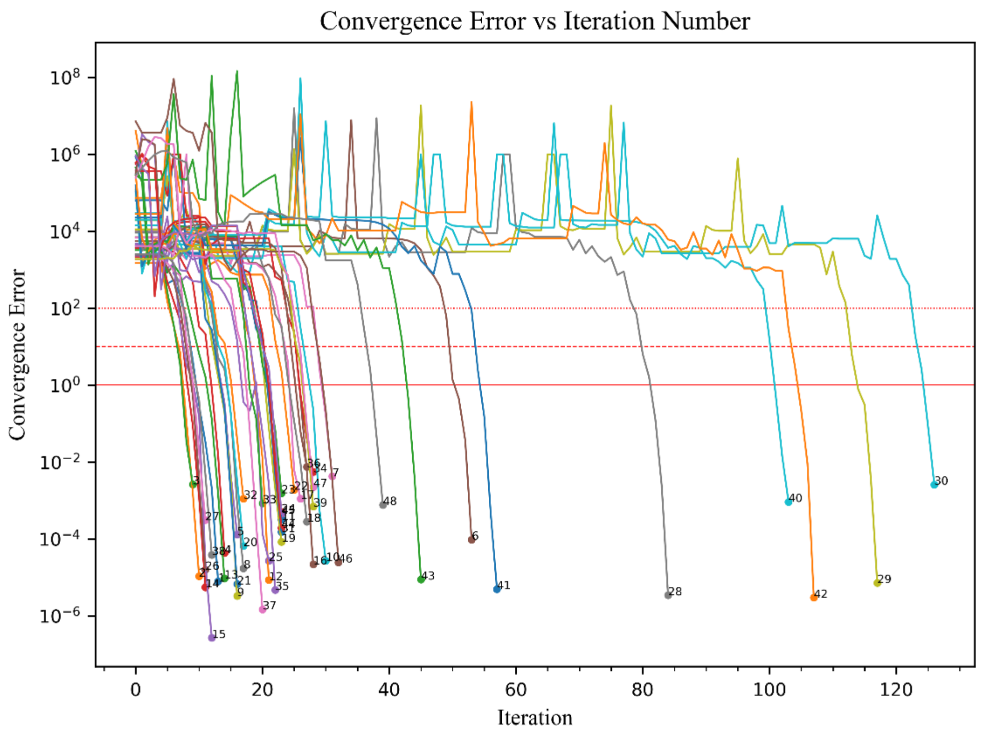


Figure A-16 AB integrated rocket (RP1) iteration convergence error by step

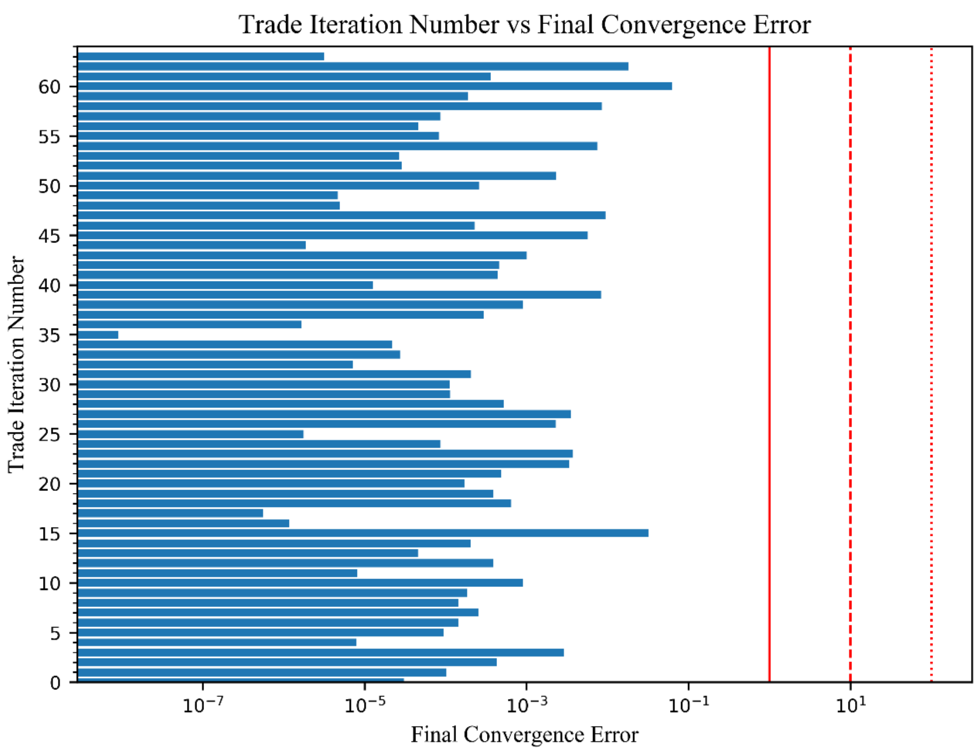


Figure A-17 AB integrated rocket (H_2) iteration final convergence

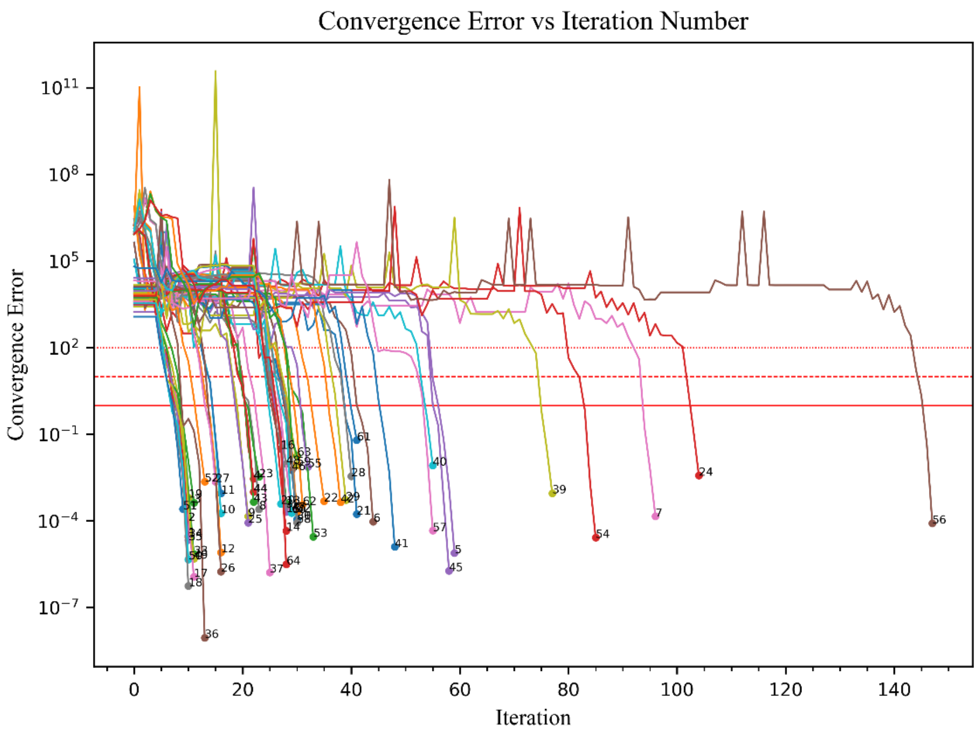


Figure A-18 AB integrated rocket (H_2) iteration convergence error by step

A.1.5.3 Trade Study Results: Solution Spaces

The solution spaces are considered by first, the reason for solution reduction in the main text, second, the full solution set for cruise vehicle, and lastly the full solution set for the total launch stack. While considering the solutions presented here in, the reader is encouraged to recall that these solutions are not optimal in the generally recognized sense. Understand that these solutions identify an area of mathematically plausible solutions but do not necessitate that they are the best nor only solutions for the given case. Additionally, when reviewing the results, note the significance that the τ parameter has on the solutions. For a given payload and cruise time, the solutions $TOGW$ can be twice the value while comparing the minimum versus maximum τ solutions, underlining the significance of volumetric efficiency for the hypersonic case.

A.1.5.3.1Solution Space: Area Reduction

The hydrogen rocket AB class solution space grows rapidly in both weight and planform area with increasing cruise time and in doing, it dominates the solution area. Consider the minimal case of zero payload. Figure A-19 illustrates this case. As one can clearly see, the hydrogen-based rocket dominates in terms of growth and total weight. The vehicle solutions rapidly exceed the B-52's capacities at 250s cruise time and even exceed the Cosmic Girl's weight limit at the 750s case. The vehicle grows rapidly due to the compounding nature of weight, volume, and aerodynamic forces. In effect, the hydrogen all-body cases above the 250s mark represent solutions more on par with a second or third vehicle iteration in a growth vehicle program. Many of these solutions are not practical solutions for most air-launched scenarios. Rather, they would likely be better suited operating as standalone vehicles or be accelerated atop a vertically launched rocket such as a Minotaur or Falcon 9. Furthermore, these design points more readily represent upper stage orbital class vehicles and should be viewed for that application. In fact, the Model-176 was intended for this purpose. Lastly, due to the hydrogen solutions far exceeding the other trade solutions in both weight and planform area, the solutions within the main text body are limited to the solutions within the comparable range of the other trades and within practical carry vehicle application (B-52 pylon limit). The full solution sets are shown here.

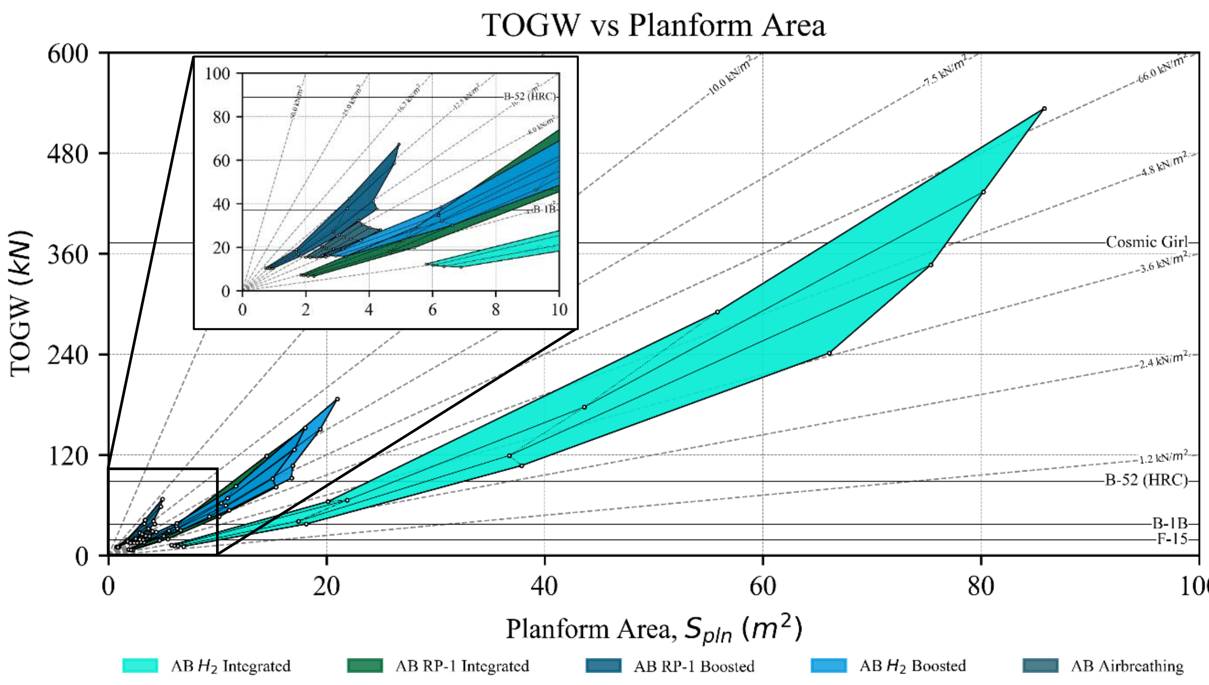


Figure A-19 Selected All-Body solutions illustrating H_2 solution dominance: S_{pln} vs $TOGW$

A.1.5.3.2 Solution Space: Cruiser

The solutions for the cruise vehicle are presented below. The cruise vehicle does not include the expendable booster for the airbreathing and boosted rocket cases. All solutions are presented in a singular figure first and are individually plotted thereafter in the order of airbreathing, inserted vehicle (externally boosted), and lastly the integrated vehicle.

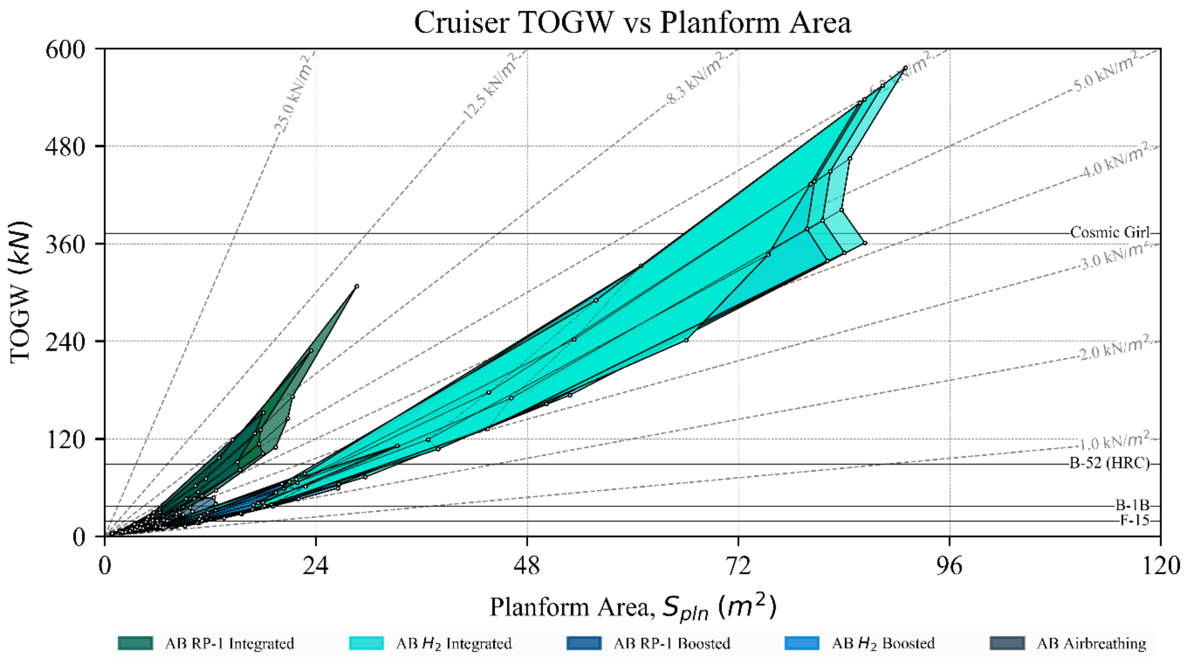


Figure A-20 All-Body cruiser vehicle solutions: S_{pln} vs $TOGW$

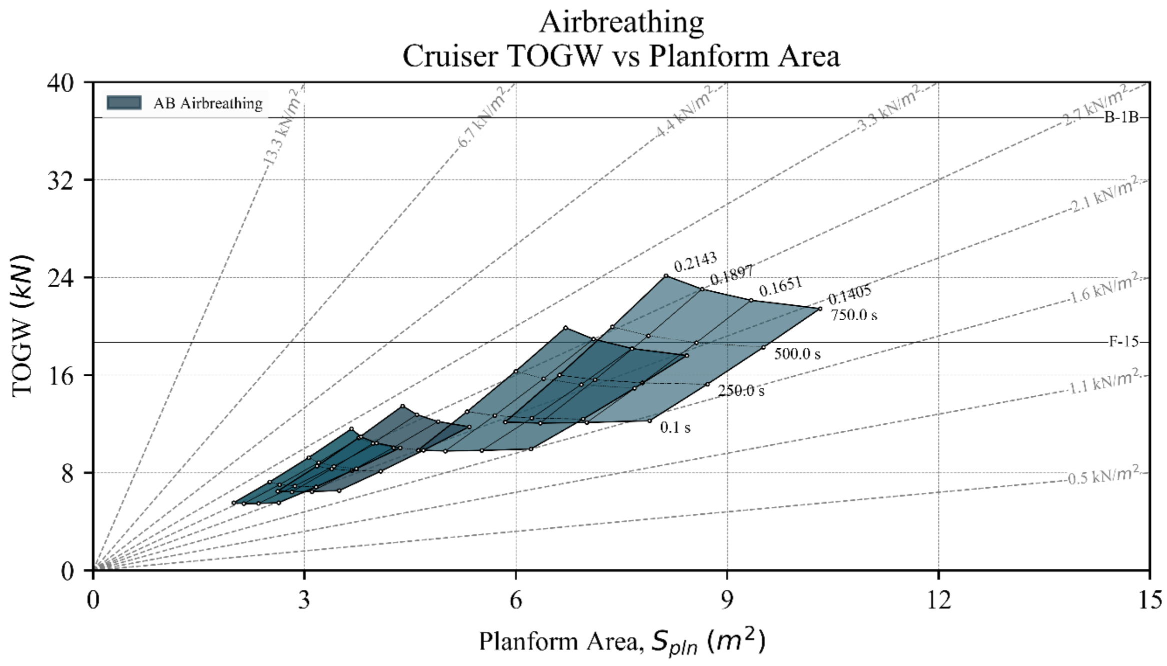


Figure A-21 Boosted All-Body airbreathing cruiser solutions: S_{pln} vs $TOGW$

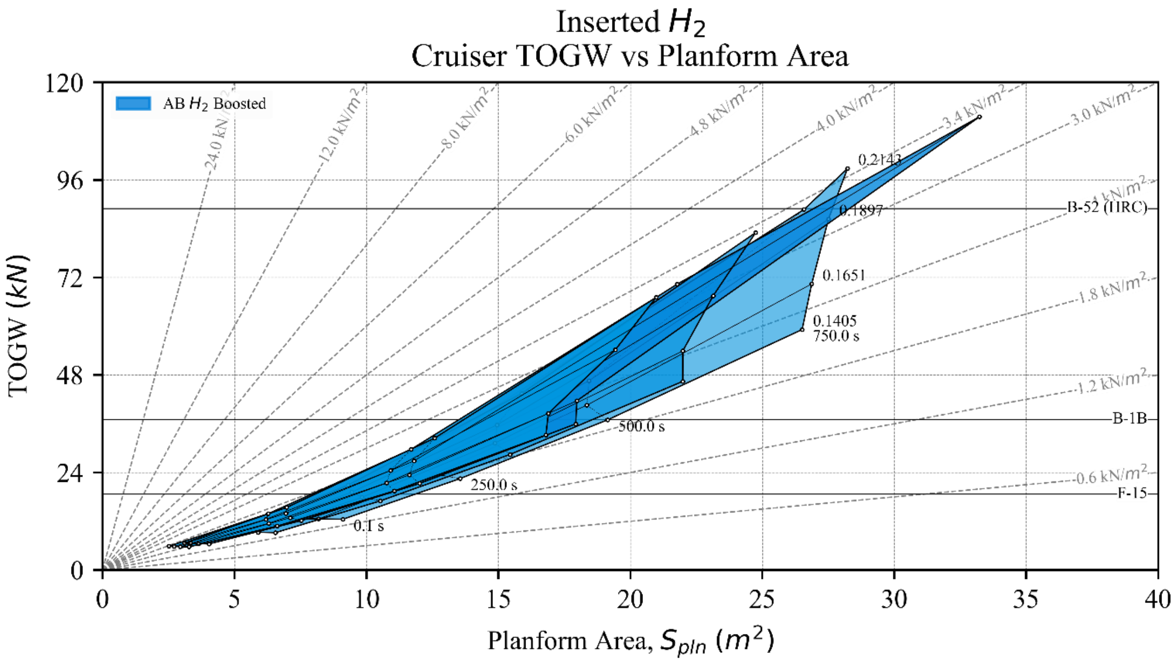


Figure A-22 Boosted All-Body H_2 rocket cruiser solutions: S_{pln} vs $TOGW$

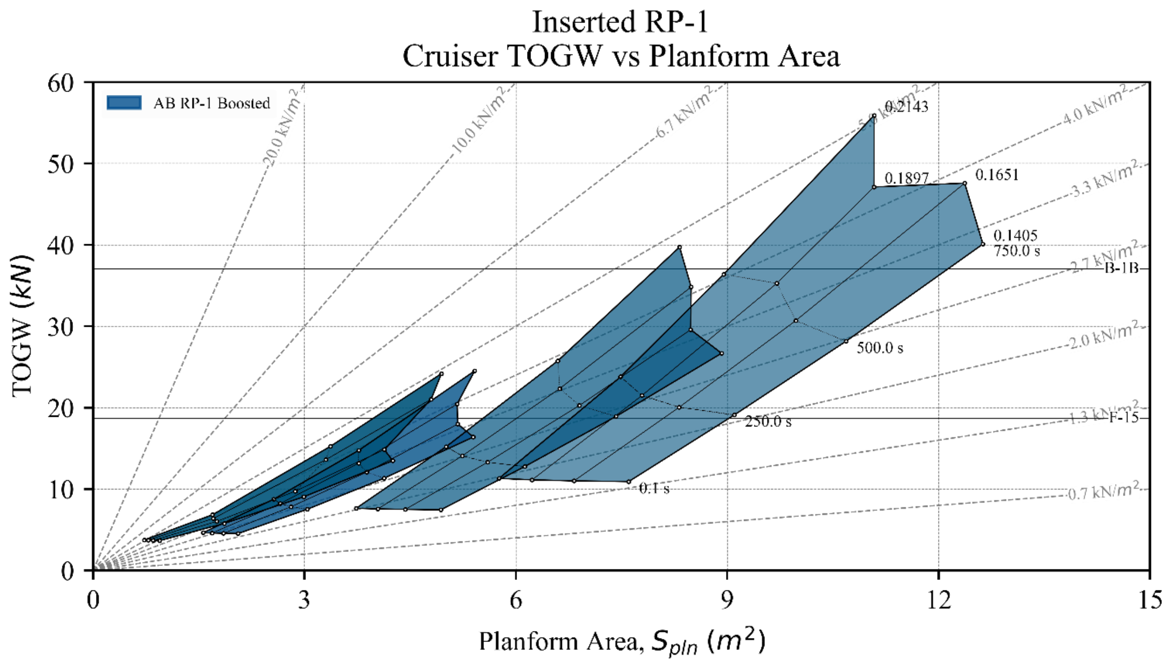


Figure A-23 Boosted All-Body RP-1 rocket cruiser vehicle solutions: S_{pln} vs $TOGW$

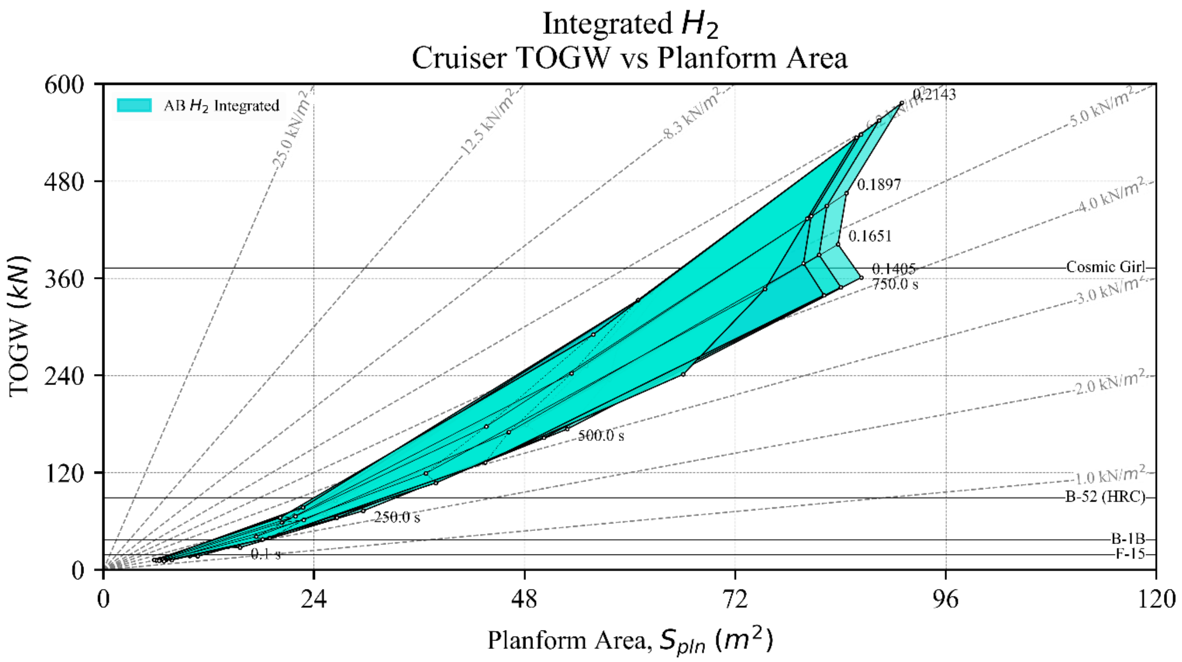


Figure A-24 Integrated All-Body H_2 rocket cruiser vehicle solutions: S_{pln} vs $TOGW$

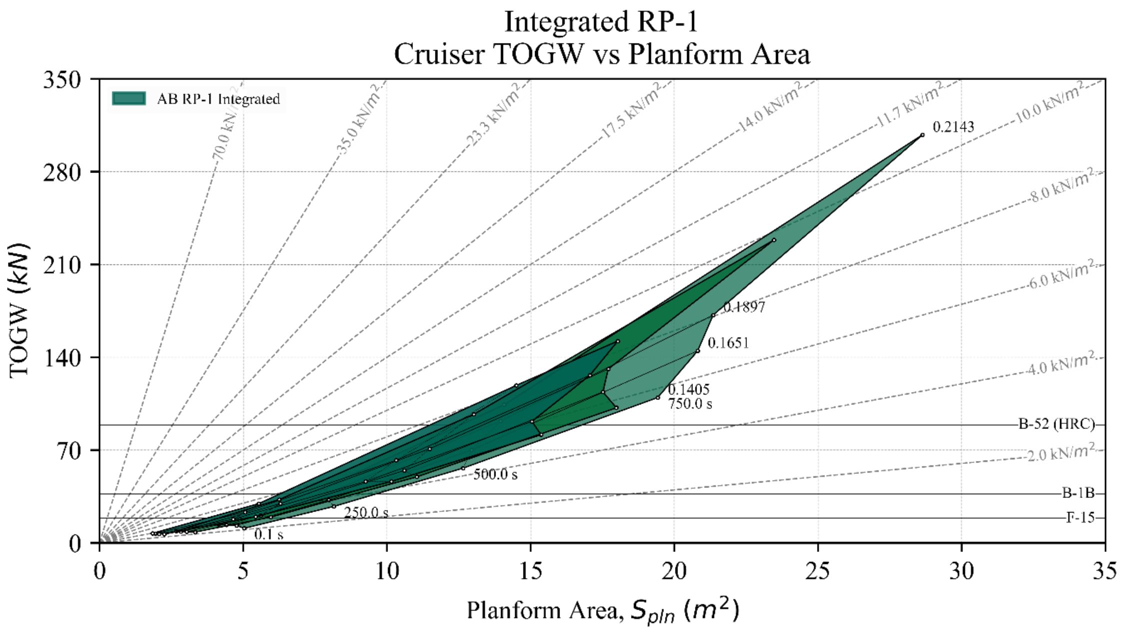


Figure A-25 Integrated All-Body RP-1 rocket cruiser vehicle solutions: S_{pln} vs $TOGW$

A.1.5.3.3 Solution Space: Full Stack

The solutions for the full launch stack (cruise with external booster as necessary) are presented below. The stack launch weight ($TOGW$) versus cruiser planform area (S_{pln}) solutions are presented in a singular figure first and are individually plotted thereafter in the order of airbreathing, inserted vehicle (externally boosted), and lastly the integrated vehicle. These are followed by the illustration of cruiser span (b) and stack length (l) versus cruiser planform area.

$TOGW$ versus S_{pln}:

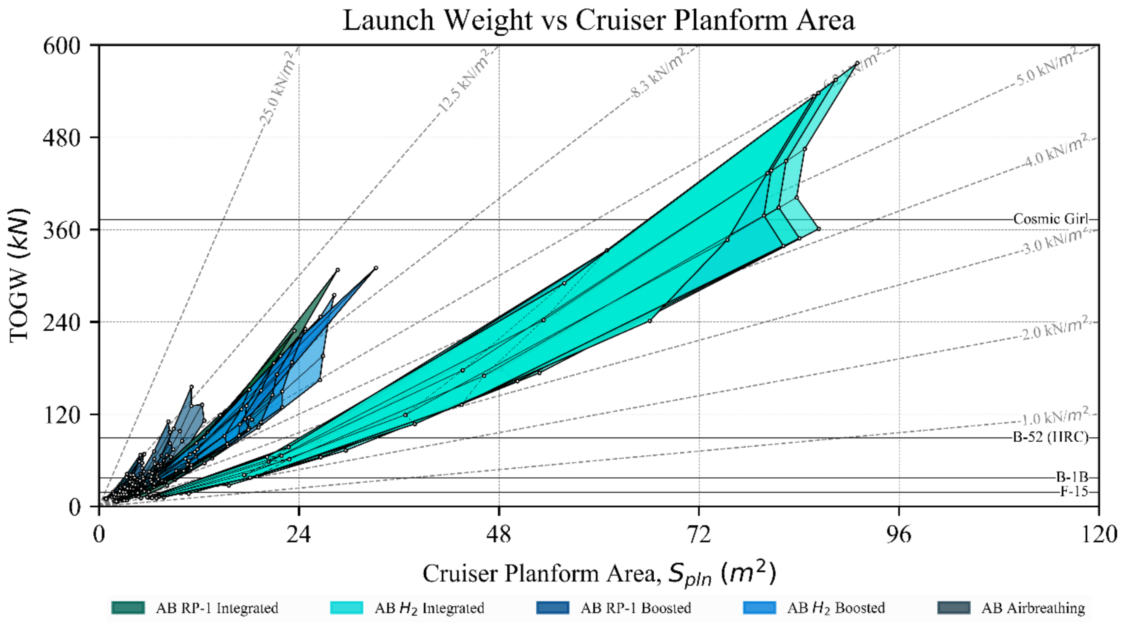


Figure A-26 All-Body full stack solutions: S_{pln} vs $TOGW$

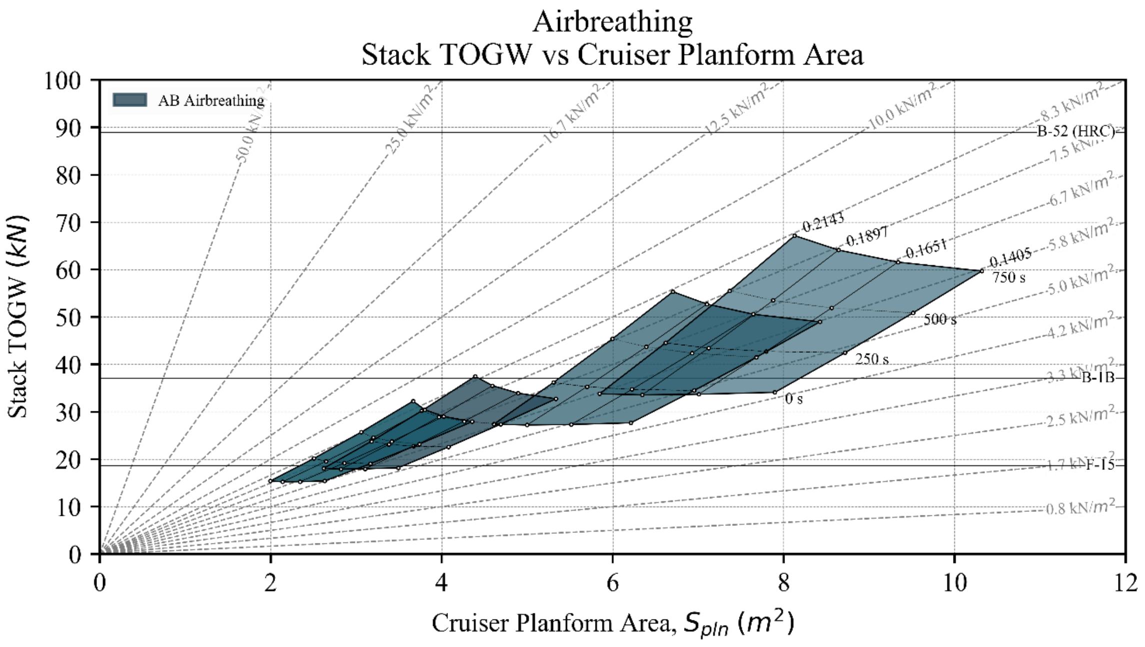


Figure A-27 Boosted All-Body airbreathing full stack solutions: S_{pln} vs $TOGW$

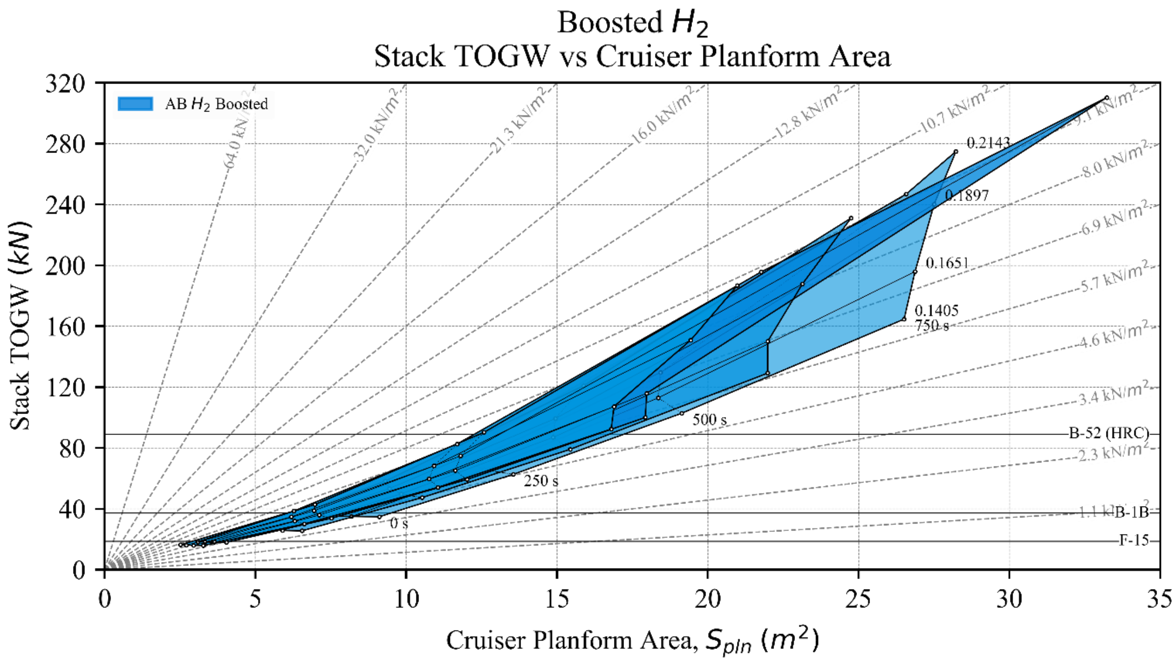


Figure A-28 Boosted All-Body H_2 rocket full stack solutions: S_{pln} vs $TOGW$

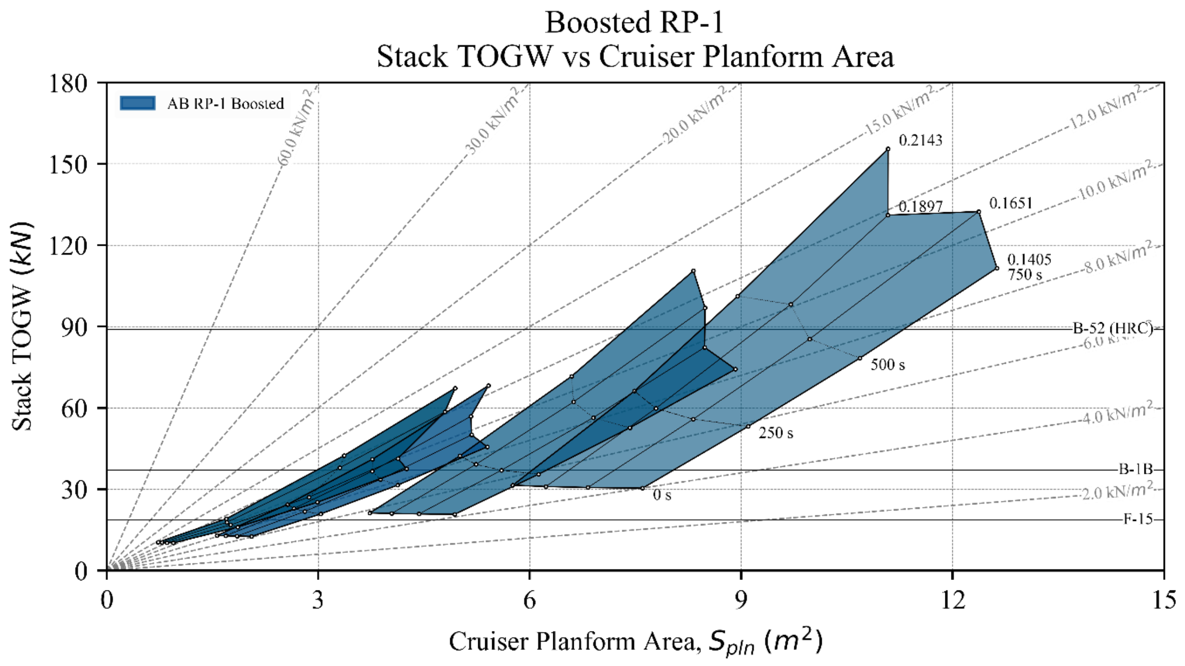


Figure A-29 Boosted All-Body RP-1 rocket full stack solutions: S_{pln} vs $TOGW$

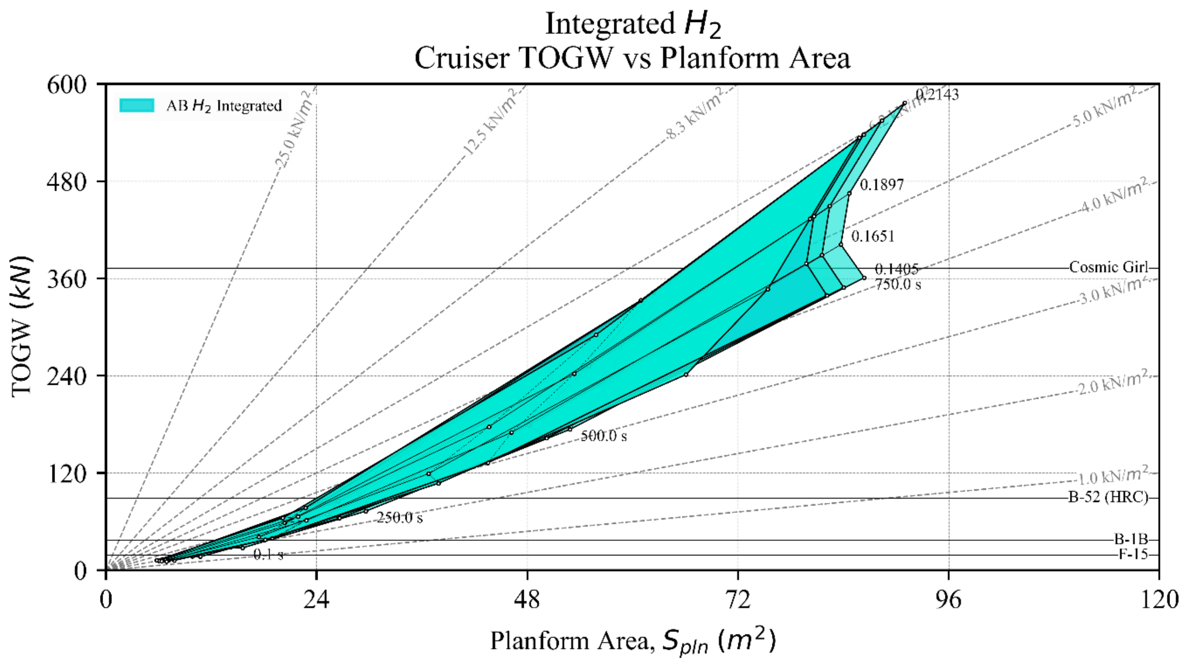


Figure A-30 Integrated All-Body H_2 rocket full stack solutions: S_{pln} vs $TOGW$

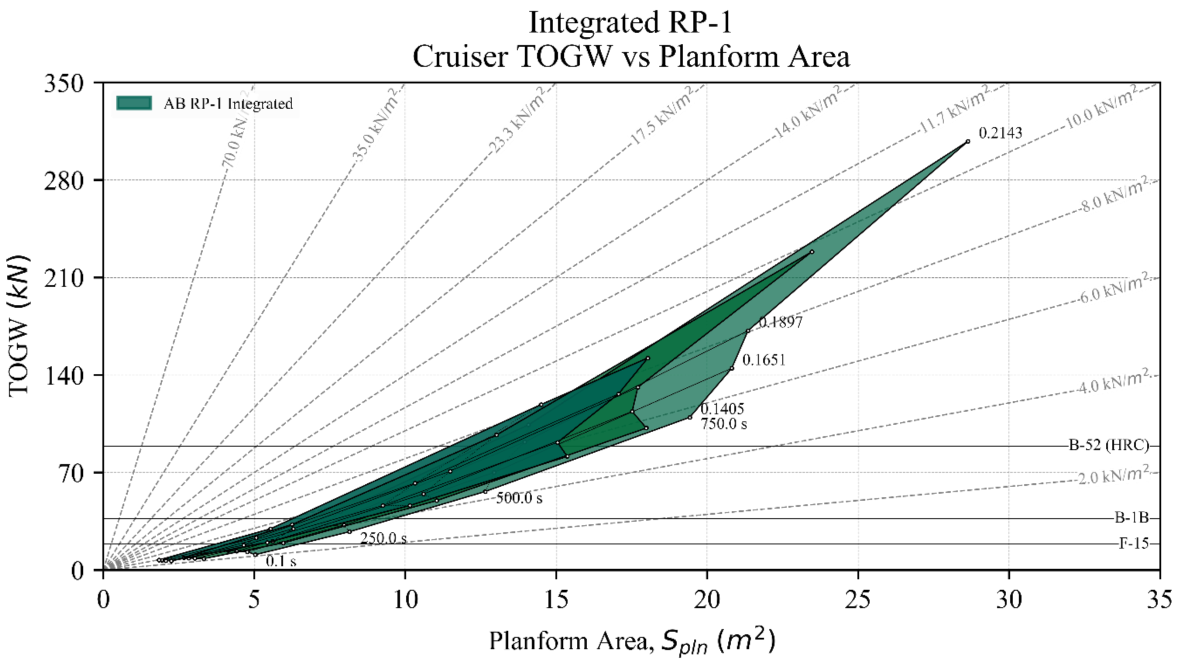


Figure A-31 Integrated All-Body RP-1 rocket full stack solutions: S_{pln} vs $TOGW$

Span and Length:

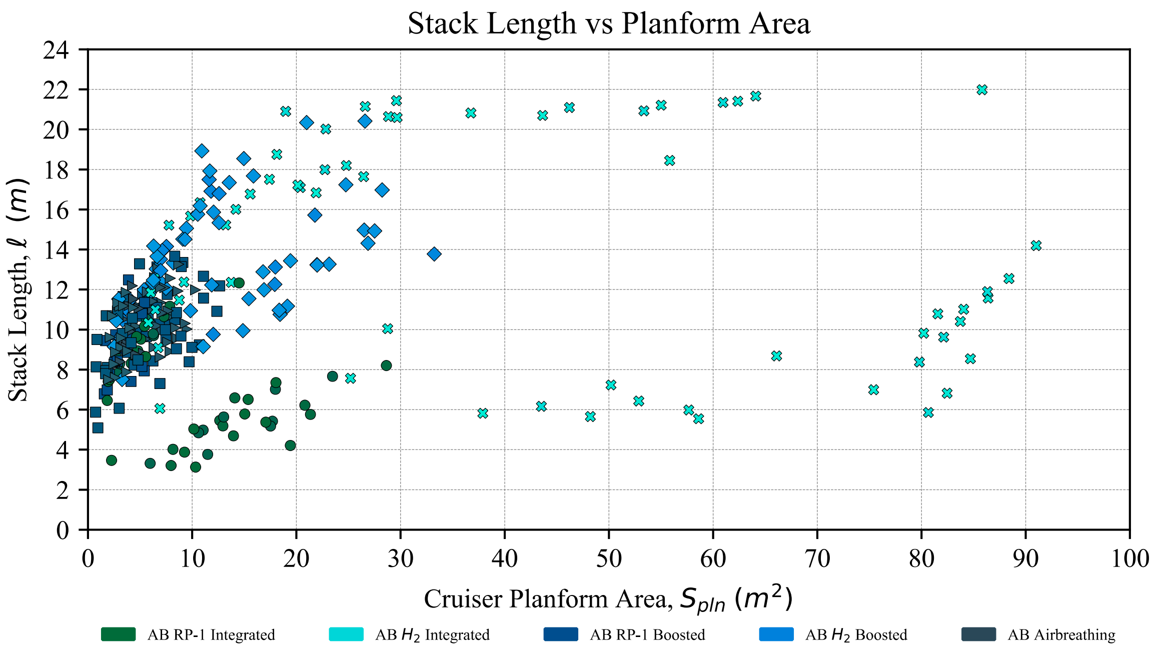


Figure A-32 All-Body full stack solutions: S_{pln} vs l

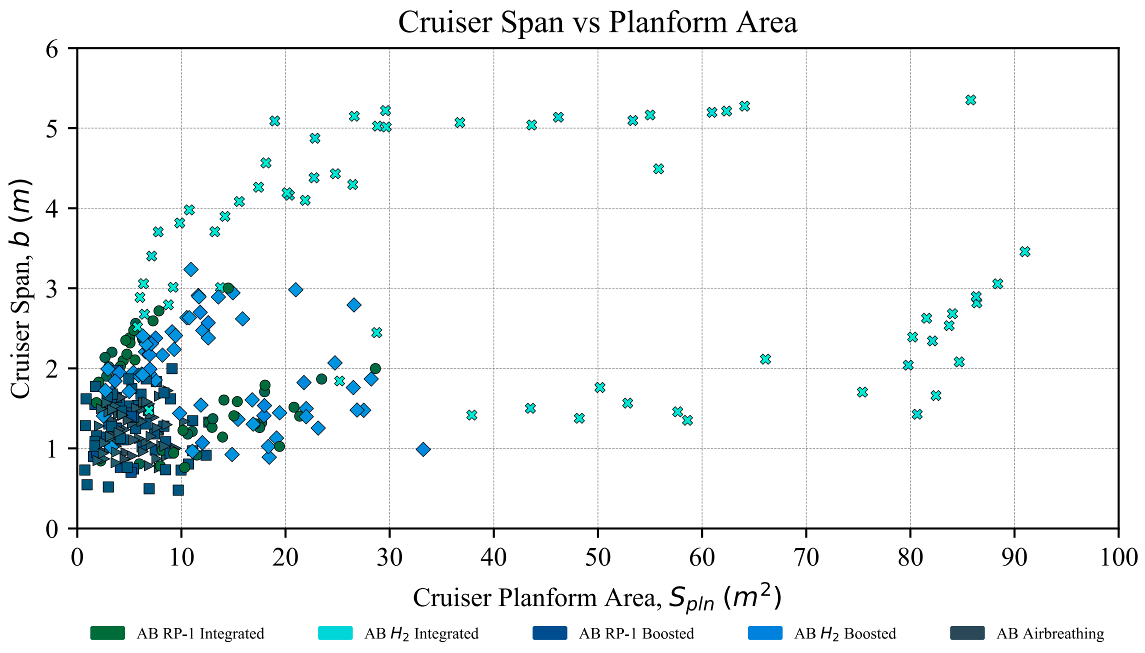


Figure A-33 All-Body full stack solutions: S_{pln} vs b

A.2 Blended Body: Road Runner Generic Hypersonic Vehicle

A.2.1 Vehicle Description

The Road Runner Generic Hypersonic Vehicle (GHV) is a family of hypersonic vehicles. The vehicles share the same concept and configuration. The top view, bottom view, and internal layout are illustrated in Figure A-34, Figure A-35, and Figure A-36 respectively. Significant features are indicated. The vehicle has a blended-body underside with a distinct fuselage on the topside. A central through flow scramjet system characterizes the vehicle. The propulsion system is ethylene based. The inlet and nozzle are both three-dimensional. The combustor is axisymmetric. The vehicle concept is the baseline for the family of five vehicle sizes. The mass flow rate scales the vehicle. The vehicle operates up to Mach 6 and a dynamic pressure range of 1000 to 2000 psf.

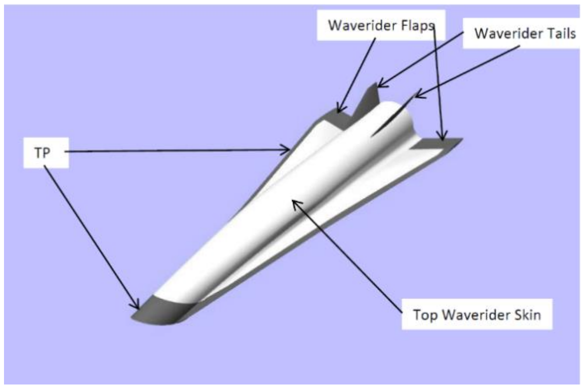


Figure A-34 GHV top view with features indicated [147]

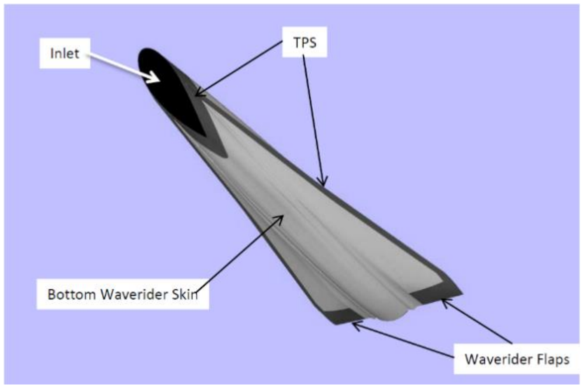


Figure A-35 GHV bottom view with features indicated [147]

The vehicle family was created with the intent to have a publicly distributable and creditable hypersonic vehicle design case for research and development. As the study, see reference [147], states:

> *[i]t was decided that a family of in-house designs should be created which would be publicly releasable and relevant to current hypersonic projects. AFRL would then be able to share these designs and any data derived from them with other government, academic or industry partners and thereby foster greater collaboration within the area.*

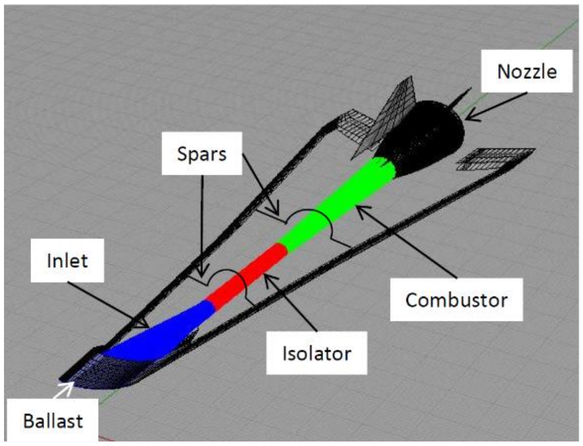


Figure A-36 Propulsive system internal layout [147]

The concepts were generated for improved research, development, and collaboration. To ensure a credible baseline hypersonic design point, the concept has been based on credible vehicles. For instance, the GHV shares many configuration similarities with the HIFiRE-6 and HIFiRE-8 [159, 160].

A.2.2 Vehicle Weights

The vehicles' primary geometric parameters and weight breakdown are given in Table A-5 and Table A-6 respectively. Mass flow rate scales the vehicle. The family set comprises of five scaled designs; each is represented.

Table A-5 GHV family primary dimensions, adapted from [147]

		Flow-Path Scale (X)				
Element	**Units**	**1**	**2**	**3**	**4**	**5**
Length Overall	m	4.4681	6.3189	7.7390	8.9362	9.9910
Fuselage Length	m	4.3134	6.1001	7.4711	8.6269	9.6451
Effective Fuselage Diameter - Nose	m	0.2793	0.3951	0.4838	0.5587	0.6246
Effective Fuselage Diameter - Tail	m	0.4786	0.6769	0.8290	0.9573	1.0703
Wing Span	m	1.4877	2.1039	2.5767	2.9754	3.3265
Nose-to-root Offset	m	0.2568	0.3632	0.4448	0.5136	0.5742
Root Chord	m	4.1059	5.8066	7.1116	8.2118	9.1811
Tip Chord	m	0.4884	0.6908	0.8460	0.9769	1.0922
Effective Leading Edge Sweep	deg	80.6	80.6	80.6	80.6	80.6
Effective Trailing Edge Sweep	deg	13.6	13.6	13.6	13.6	13.6

Table A-6 GHV family primary mass breakdown, adapted from [147]

		Flow-Path Scale (X)				
	Mass (Kg)	**1**	**2**	**3**	**4**	**5**
Structure	Total Skin	96.3	228.9	405.3	613.7	674.1
	Flaps	21.5	58.7	110.0	166.1	233.6
	Tails	7.9	21.4	39.5	59.6	83.4
	TPS	15.3	40.1	73.4	110.0	153.3
	Spars and Kneels	7.2	14.1	21.1	27.9	35.0
	Inlet	12.8	24.7	36.4	48.0	59.6
	Isolator	17.1	33.6	50.0	66.3	82.6
	Combustor	42.3	83.6	124.6	165.7	206.6
	Nozzle	54.2	101.2	156.0	200.4	251.5
Fluids	Usable Fuel	102.9	327.4	624.0	968.2	1425.0
	Residual Fuel	9.0	28.7	54.7	84.9	125.0
	Nitrogen	1.9	6.0	11.5	17.9	26.3
Other	Ballast	31.8	79.4	136.1	181.4	226.8
	GN&C	133.4	133.4	133.4	133.4	133.4
	Payload	0.0	0.0	0.0	0.0	0.0
Summary	Gross	553.5	1181.0	1976.1	2843.4	3716.3
	Empty	450.6	853.7	1352.1	1875.2	2291.2
	Dry	439.6	818.9	1285.9	1772.4	2139.9
	Structure Mass Fraction	0.496	0.513	0.514	0.513	0.479
	Fluids Mass Fraction	0.206	0.307	0.349	0.377	0.424
	Other Mass Fraction	0.298	0.180	0.136	0.111	0.097

A.2.3 Notional Mission

The GHV's notional mission is illustrated in Figure A-37. The general mission segment conditions are given in Table A-7. The vehicle is air-launched, and it is assumed that a rocket booster accelerates the vehicle to the engine start condition—between 1500-2500 psf—at which point the booster separates. On engine start, the vehicle accelerates from Mach 4 to Mach 6, climbing in altitude as it does. At Mach 6, the vehicle levels out and performs a lifting cruise segment. Powered cruise occurs at a dynamic pressure between 1000 – 2000 psf. The cruise segment is optionally split by a maneuver execution. Maneuver options considered include an 180^{o} turn or a 45^{o}-90^{o}-45^{o}switch back maneuver. In the event that no maneuver is executed, a straight fly-out mission is executed. In this mission situation, no maneuvers are performed, only acceleration-climb and cruise conditions are considered. After the cruise and optional maneuver, the engine shuts off; this ends the powered segments of the mission. After engine shutoff, the vehicle descends, during which it is able to perform unpowered tests. The vehicle is not reusable.

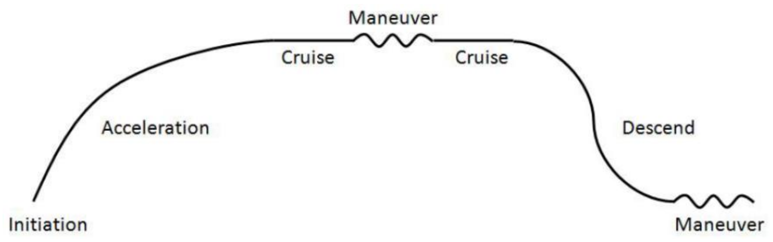


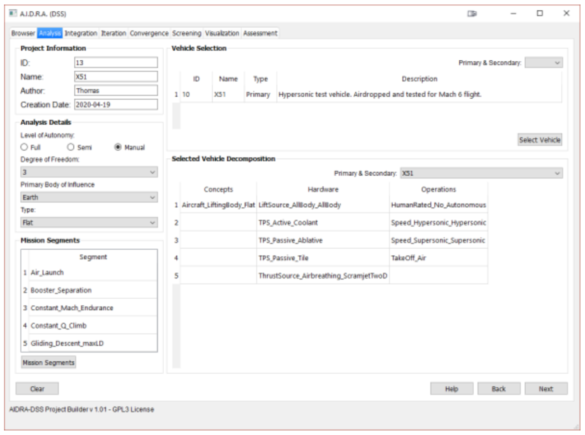
Figure A-37 GHV notional mission profile [147]

Table A-7 GHV notional mission conditions, adapted from [147]

	Mach	Lift/Weight	Dynamic Pressure	
			psf	kPa
Initiation (booster separation)	4 – 5	1	2500 - 1500	120 - 70
Acceleration and Climb	4 – 6	>1	2500 - 1500	120 - 70
Cruise	6	1	2000 - 1000	96 - 48
Maneuver (#1 or #2)	~6	~2	2000 - 1000	96 - 48
Cruise	6	1	2000 - 1000	96 - 48
Descend (powered)	6 – 4	<1	2000 - 3000	96 - 140
Descend (unpowered)	4 – 3	<1	2500 - 5000	120 - 240
Maneuver (unpowered)	~3	>1	5000	240

A.2.4 Project Builder Selections

This section documents the *Project Builder* selections. Depicted are each page's selections. The selection is organized according to the *Project Builder*'s tabs. The selections shown are for the reverse sizing case however, the same general selections were made for the trade study as well. The only significant variance being the mission profile and selected method (as laid out in Chapter 5 Section 5.4) in addition to the trade variable selections.


A.I.D.R.A. (DSS)
Browser Analysis Integration Iteration Convergence Screening Visualization Assessment
Select Vehicle
Mission Segments
Clear Help Back Next

Project Information

ID: 13
Name: X51
Author: Thomas
Creation Date: 2020-04-19

Analysis Details

Level of Autonomy: Full Semi [✓ Manual]
Degree of Freedom: 3
Primary Body of Influence: Earth
Type: Flat

Mission Segments

	Segment
1	Air_Launch
2	Booster_Separation
3	Constant_Mach_Endurance
4	Constant_Q_Climb
5	Gliding_Descent_maxLD

Vehicle Selection

Primary & Secondary:

	ID	Name	Type	Description
1	10	X51	Primary	Hypersonic test vehicle. Airdropped and tested for Mach 6 flight.

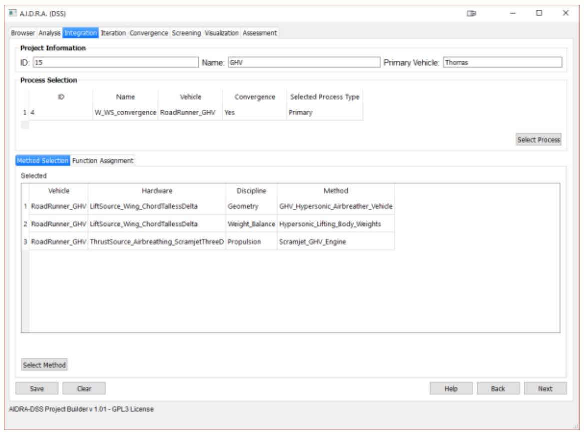
Selected Vehicle Decomposition

Primary & Secondary: X51

	Concepts	Hardware	Operations
1	Aircraft_LiftingBody_Flat	LiftSource_AllBody_AllBody	HumanRated_No_Autonomous
2		TPS_Active_Coolant	Speed_Hypersonic_Hypersonic
3		TPS_Passive_Ablative	Speed_Supersonic_Supersonic
4		TPS_Passive_Tile	TakeOff_Air
5		ThrustSource_Airbreathing_ScramjetTwoD	

AIDRA-DSS Project Builder v 1.01 - GPL3 License

Figure A-38 GHV Analysis page


A.I.D.R.A. (DSS)
Browser Analysis Integration Iteration Convergence Screening Visualization Assessment
Select Process
Method Selection Function Assignment
Select Method
Save Clear Help Back Next

Project Information

ID: 15 Name: GHV Primary Vehicle: Thomas

Process Selection

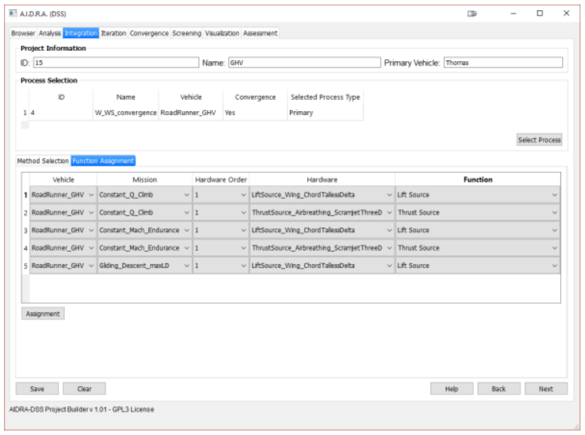
	ID	Name	Vehicle	Convergence	Selected Process Type
1	4	W_WS_convergence	RoadRunner_GHV	Yes	Primary

Selected

	Vehicle	Hardware	Discipline	Method
1	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Geometry	GHV_Hypersonic_Airbreather_Vehicle
2	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Weight_Balance	Hypersonic_Lifting_Body_Weights
3	RoadRunner_GHV	ThrustSource_Airbreathing_ScramjetThreeD	Propulsion	Scramjet_GHV_Engine

AIDRA-DSS Project Builder v 1.01 - GPL3 License

Figure A-39 GHV Integration page, (a) Method Selection


A.I.D.R.A. (DSS)
Browser Analysis Integration Iteration Convergence Screening Visualization Assessment
Select Process
Method Selection Function Assignment
Assignment
Save Clear Help Back Next

Project Information

ID: 15 Name: GHV Primary Vehicle: Thomas

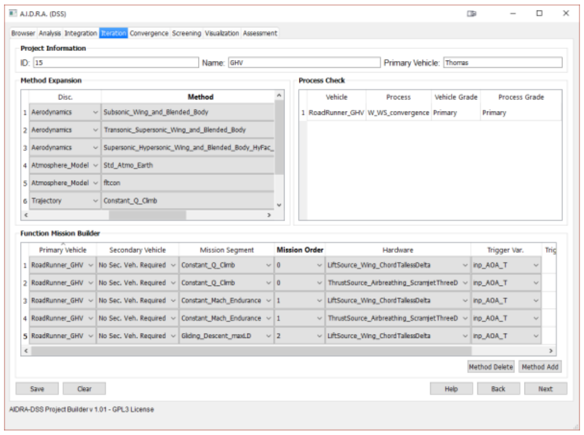
Process Selection

	ID	Name	Vehicle	Convergence	Selected Process Type
1	4	W_WS_convergence	RoadRunner_GHV	Yes	Primary

	Vehicle	Mission	Hardware Order	Hardware	Function
1	RoadRunner_GHV	Constant_Q_Climb	1	LiftSource_Wing_ChordTallessDelta	Lift Source
2	RoadRunner_GHV	Constant_Q_Climb	1	ThrustSource_Airbreathing_ScramjetThreeD	Thrust Source
3	RoadRunner_GHV	Constant_Mach_Endurance	1	LiftSource_Wing_ChordTallessDelta	Lift Source
4	RoadRunner_GHV	Constant_Mach_Endurance	1	ThrustSource_Airbreathing_ScramjetThreeD	Thrust Source
5	RoadRunner_GHV	Gliding_Descent_maxLD	1	LiftSource_Wing_ChordTallessDelta	Lift Source

AIDRA-DSS Project Builder v 1.01 - GPL3 License

Figure A-40 GHV Integration page, (b) Function Assignment


A.I.D.R.A. (DSS)
Browser Analysis Integration Iteration Convergence Screening Visualization Assessment
Method Delete Method Add
Save Clear Help Back Next

Project Information

ID: 15 Name: GHV Primary Vehicle: Thomas

Method Expansion

	Disc.	Method
1	Aerodynamics	Subsonic_Wing_and_Blended_Body
2	Aerodynamics	Transonic_Supersonic_Wing_and_Blended_Body
3	Aerodynamics	Supersonic_Hypersonic_Wing_and_Blended_Body_HyFac_
4	Atmosphere_Model	Std_Atmo_Earth
5	Atmosphere_Model	fltcon
6	Trajectory	Constant_Q_Climb

Process Check

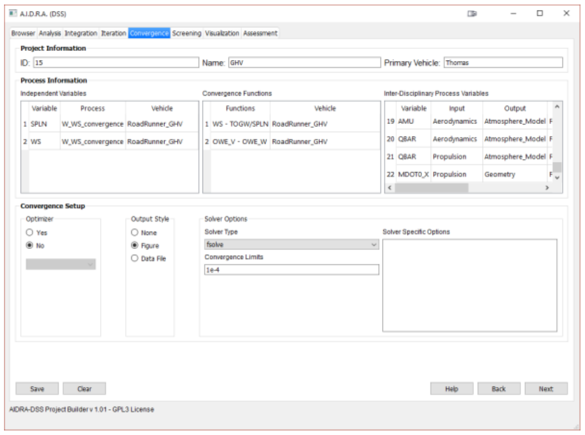
	Vehicle	Process	Vehicle Grade	Process Grade
1	RoadRunner_GHV	W_WS_convergence	Primary	Primary

Function Mission Builder

	Primary Vehicle	Secondary Vehicle	Mission Segment	Mission Order	Hardware	Trigger Var.	Trig
1	RoadRunner_GHV	No Sec. Veh. Required	Constant_Q_Climb	0	LiftSource_Wing_ChordTallessDelta	inp_AOA_T	
2	RoadRunner_GHV	No Sec. Veh. Required	Constant_Q_Climb	0	ThrustSource_Airbreathing_ScramjetThreeD	inp_AOA_T	
3	RoadRunner_GHV	No Sec. Veh. Required	Constant_Mach_Endurance	1	LiftSource_Wing_ChordTallessDelta	inp_AOA_T	
4	RoadRunner_GHV	No Sec. Veh. Required	Constant_Mach_Endurance	1	ThrustSource_Airbreathing_ScramjetThreeD	inp_AOA_T	
5	RoadRunner_GHV	No Sec. Veh. Required	Gliding_Descent_maxLD	2	LiftSource_Wing_ChordTallessDelta	inp_AOA_T	

AIDRA-DSS Project Builder v 1.01 - GPL3 License

Figure A-41 GHV Iteration page


A.I.D.R.A. (DSS)
Browser Analysis Integration Iteration Convergence Screening Visualization Assessment
Save Clear Help Back Next

Project Information

ID: 15 Name: GHV Primary Vehicle: Thomas

Process Information

Independent Variables

	Variable	Process	Vehicle
1	SPLN	W_WS_convergence	RoadRunner_GHV
2	WS	W_WS_convergence	RoadRunner_GHV

Convergence Functions

	Functions	Vehicle
1	WS - TOGW/SPLN	RoadRunner_GHV
2	OWE_V - OWE_W	RoadRunner_GHV

Inter-Disciplinary Process Variables

	Variable	Input	Output	
19	AMU	Aerodynamics	Atmosphere_Model	F
20	QBAR	Aerodynamics	Atmosphere_Model	F
21	QBAR	Propulsion	Atmosphere_Model	F
22	MDOT0_X	Propulsion	Geometry	F

Convergence Setup

Optimizer: Yes [✓ No]

Output Style: None [✓ Figure] Data File

Solver Options

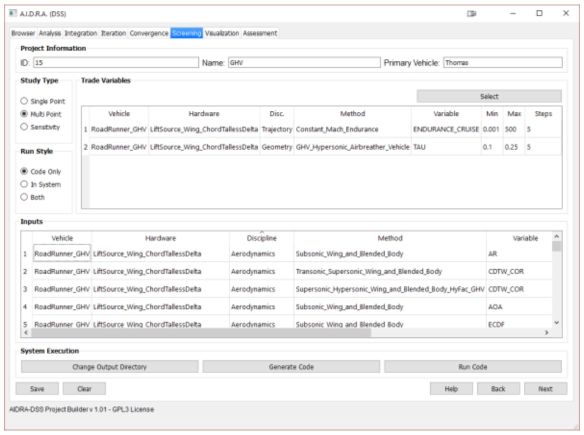
Solver Type: fsolve

Convergence Limits: 1e-4

Solver Specific Options

AIDRA-DSS Project Builder v 1.01 - GPL3 License

Figure A-42 GHV Convergence page


A.I.D.R.A. (DSS)
Browser Analysis Integration Iteration Convergence Screening Visualization Assessment
Select
Change Output Directory Generate Code Run Code
Save Clear Help Back Next

Project Information

ID: 15 Name: GHV Primary Vehicle: Thomas

Study Type

Single Point [✓ Multi Point] Sensitivity

Run Style

[✓ Code Only] In System Both

Trade Variables

	Vehicle	Hardware	Disc.	Method	Variable	Min	Max	Steps
1	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Trajectory	Constant_Mach_Endurance	ENDURANCE_CRUISE	0.001	500	5
2	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Geometry	GHV_Hypersonic_Airbreather_Vehicle	TAU	0.1	0.25	5

Inputs

	Vehicle	Hardware	Discipline	Method	Variable
1	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Aerodynamics	Subsonic_Wing_and_Blended_Body	AR
2	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Aerodynamics	Transonic_Supersonic_Wing_and_Blended_Body	CDTW_COR
3	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Aerodynamics	Supersonic_Hypersonic_Wing_and_Blended_Body_HyFac_GHV	CDTW_COR
4	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Aerodynamics	Subsonic_Wing_and_Blended_Body	AOA
5	RoadRunner_GHV	LiftSource_Wing_ChordTallessDelta	Aerodynamics	Subsonic_Wing_and_Blended_Body	ECDF

System Execution

AIDRA-DSS Project Builder v 1.01 - GPL3 License

Figure A-43 GHV Screening page

A.2.5 Trade Study

The GHV Blended-Body (BB) concept was used as a baseline concept for a trade study of blended-body hypersonic cruiser vehicles. This section presents the trade study's trade matrix, the convergence behavior and results, and a consideration of the results pictorially in context to potential carrier vehicles and general solution space behavior.

A.2.5.1 Trade Matrix

The blended-body concept evaluation evolved around two concepts—airbreathing and non-airbreathing. For each concept volume coefficient (τ), cruise time (t_{cruise}), and payload weight (W_{pay}) were traded. For the non-airbreathing cases, the fuel type was traded as well. The fuels traded were RP-1 and H_2. The trade ranges are shown in Table A-8.

Table A-8 Blended-body trade matrix

Vehicle Tag	**Baseline Vehicle**	**Propulsion System**	**Boost Type**	**Fuel Type**	**Tau Range**	**Payload (N)**	**Cruise Time (s)**
BB3DS	GHV	3D Scramjet	External	Ethylene	0.0657 – 0.0735	0	0 – 750
BBRKT	GHV	Liquid Rocket	External	H_2 / RP-1	0.09 - 0.12	0 – 4000	0 – 300
BBRKT	GHV	Liquid Rocket	Internal	H_2 / RP-1	0.09 – 0.12	0 – 4000	0 – 300

A.2.5.2 Trade Convergence Summary: Iteration Errors

The trades were executed as laid out. For reference, the final convergence error and convergence error per independent variable iteration for each trade are presented pictorially below. The order presented follows that of the row order in the trade matrix table. As can be seen, the convergence for the airbreathing cases occurred much more readily and with less noticeable difficulty (as measured by total independent variable iterations to solution convergence). Furthermore, the problem solving became exceedingly more laborious as the problem progressed through the non-airbreathing cases and specifically the hydrogen-fueled cases, as the vehicle increased in size due to mission parameters. Additionally, the solution finding could be exceptionally sensitive to slight changes in independent variable value. The numerical solver would frequently fall into local valleys of no solution. On repeated evaluation of the same point, different solutions would occur as well, highlighting that the solutions are non-unique. Future studies should rely on robust global solvers. Lastly, note that not all points converged, that is—for the given process—the solutions would not close mathematically.

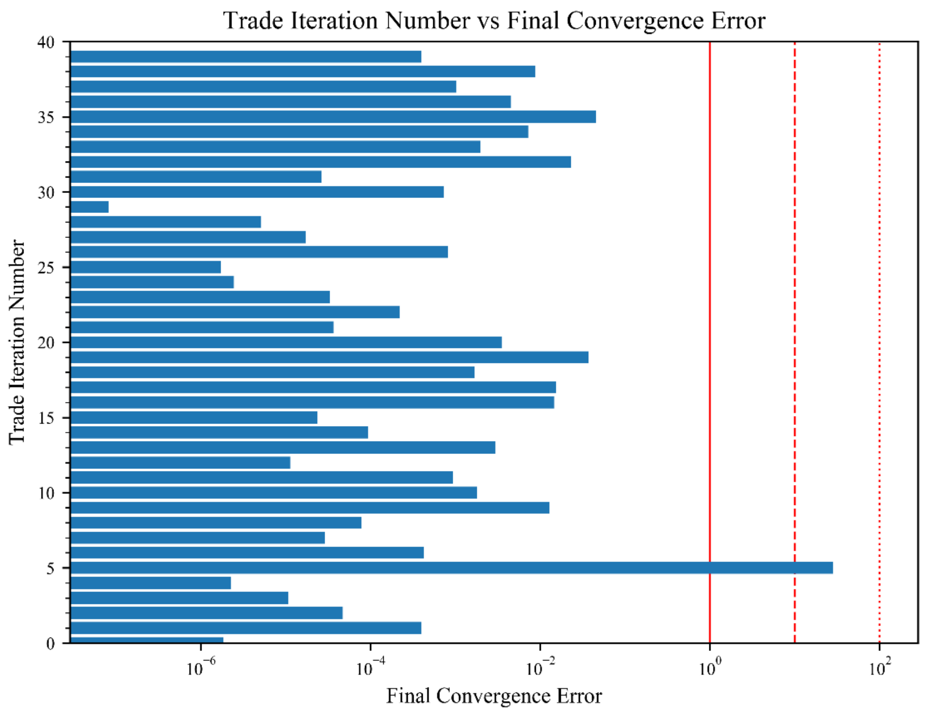


Figure A-44 BB scramjet iteration final convergence

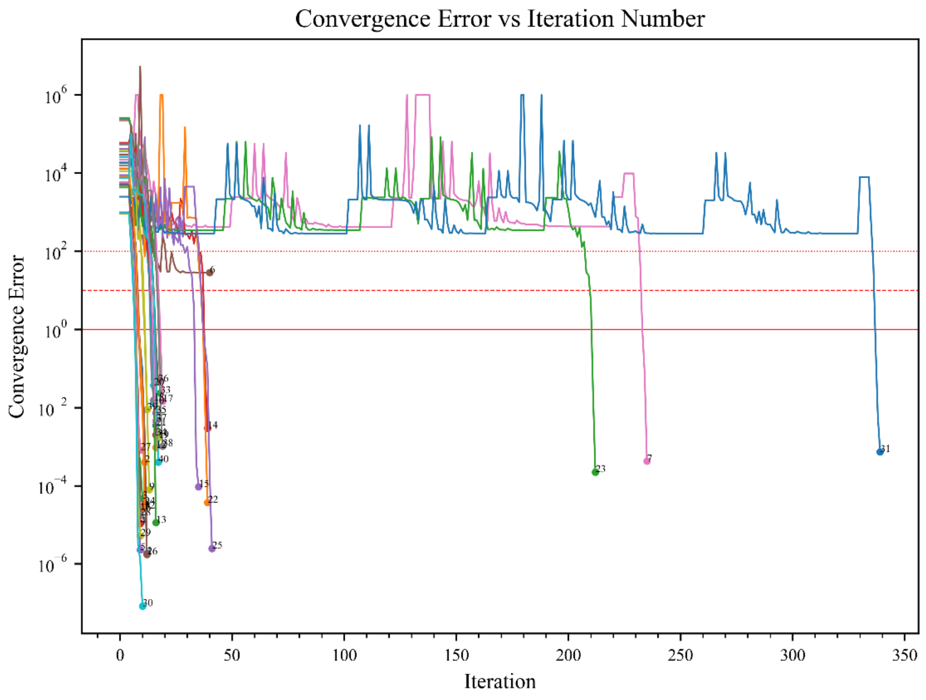


Figure A-45 BB scramjet iteration convergence error by step

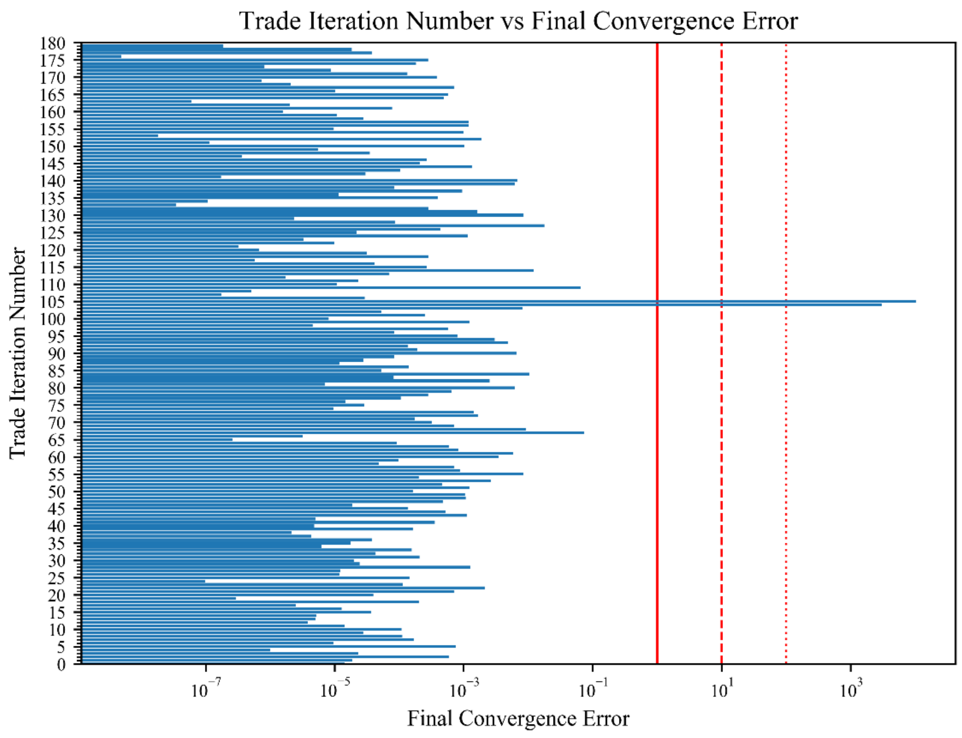


Figure A-46 BB boosted rocket (RP1) iteration final convergence

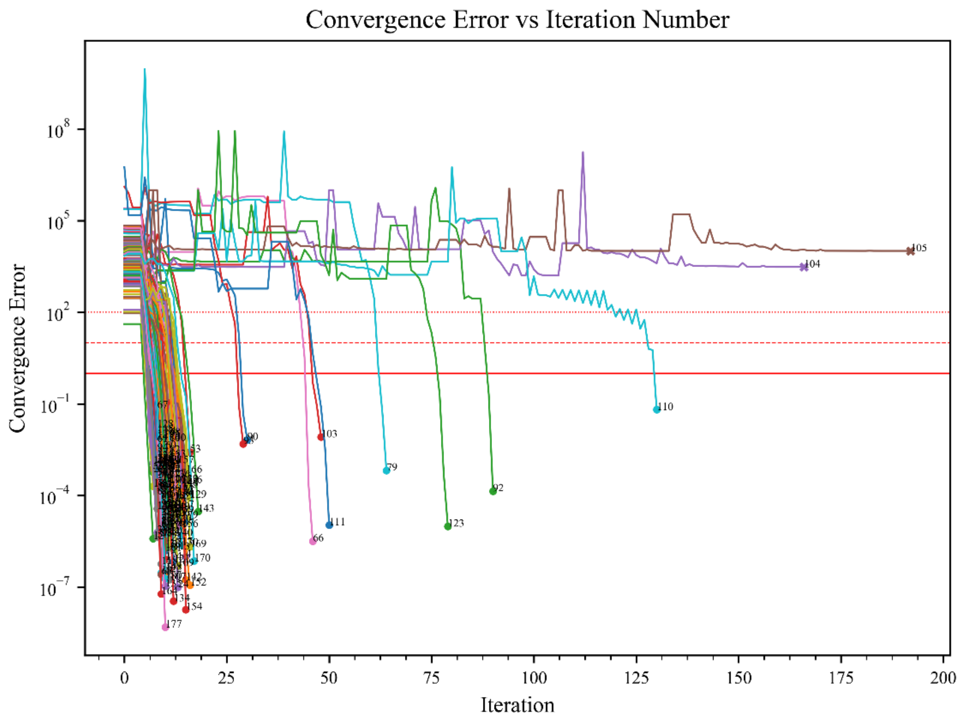


Figure A-47 BB boosted rocket (RP1) iteration convergence error by step

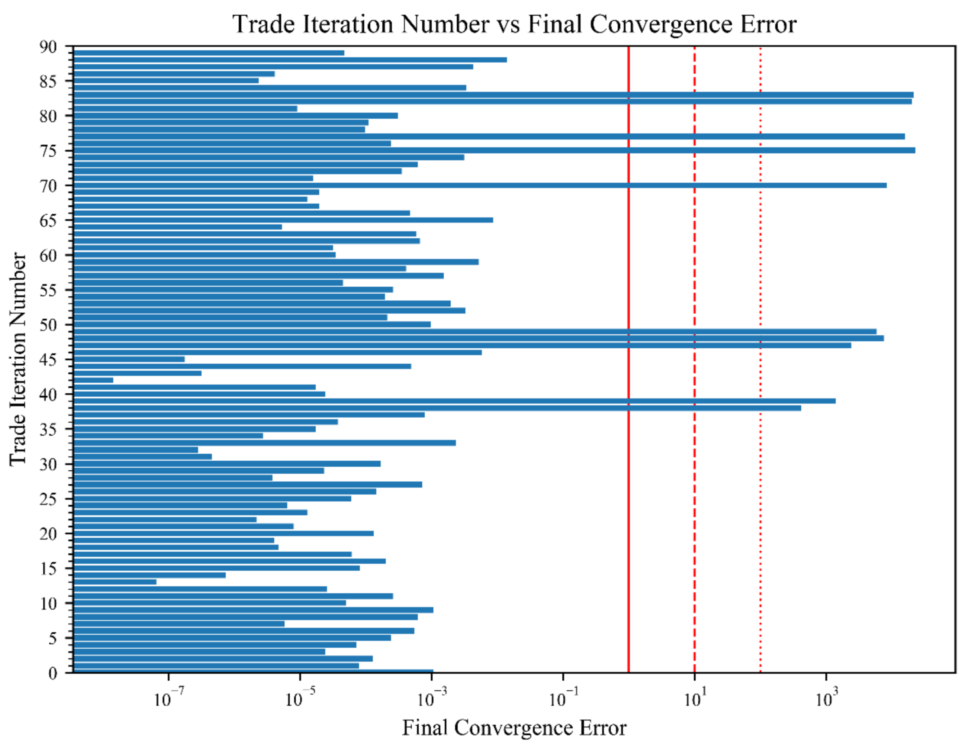


Figure A-48 BB boosted rocket (H_2) iteration final convergence

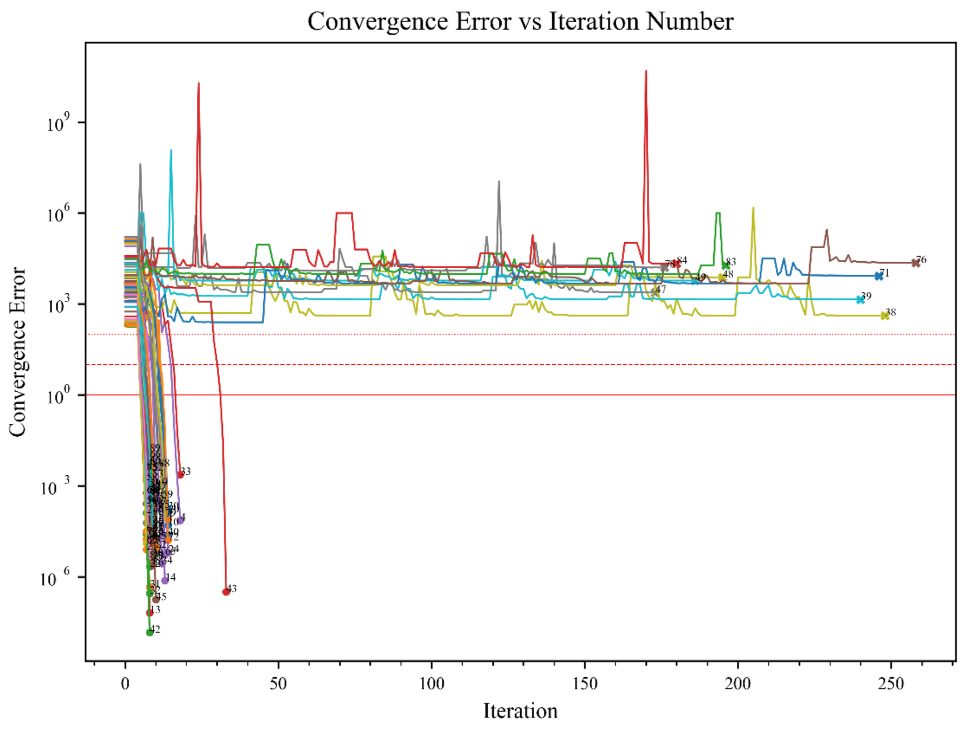


Figure A-49 BB boosted rocket (H_2) iteration convergence error by step

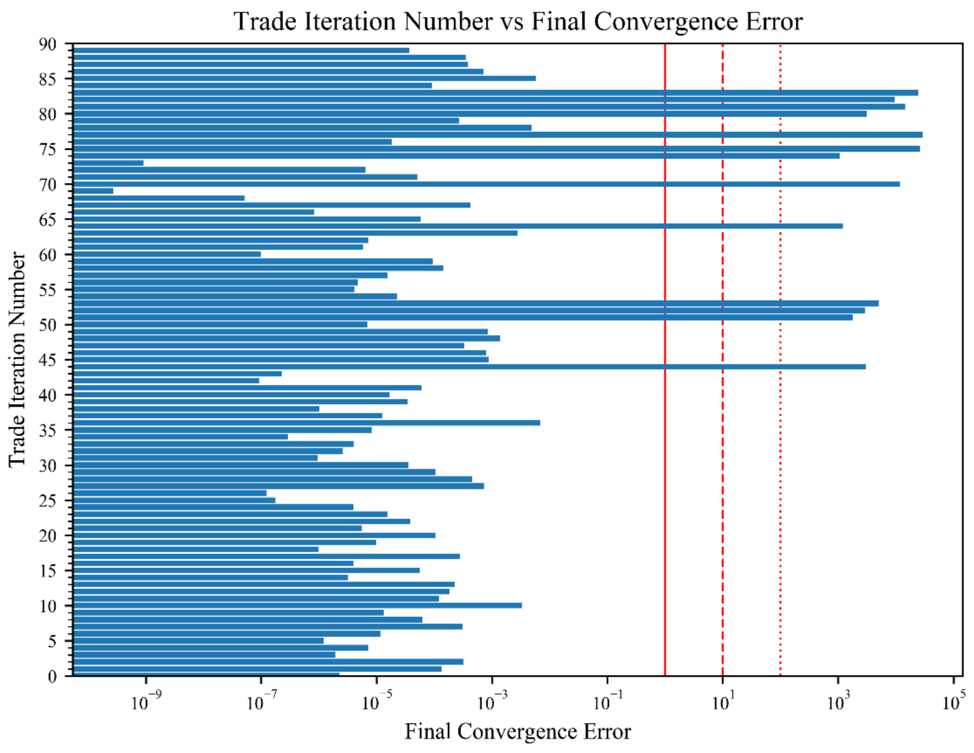


Figure A-50 BB integrated rocket (RP1) iteration final convergence

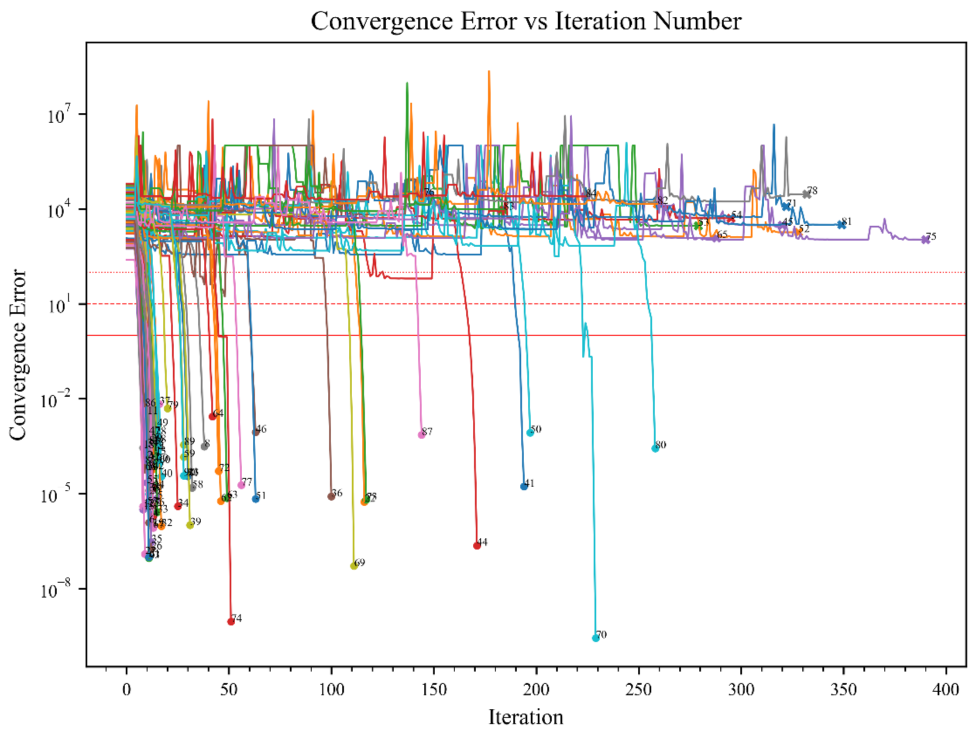


Figure A-51 BB integrated rocket (RP1) iteration convergence error by step