COMPLEX MULTIDISCIPLINARY SYSTEM COMPOSITION FOR AEROSPACE VEHICLE CONCEPTUAL DESIGN

by

LEX GONZALEZ

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY OF AEROSPACE ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2016

Acknowledgements

I would first like to thank my advisor, Dr. Chudoba, for his support throughout all these long years. He stuck with me when all I had was ‘idle potential’, trusted me when I opted to research fields of study not traditionally focused in aerospace, and aided me forming those concepts into a structured solution.

To the late Prof. Paul Czysz, the few times I was privileged to meet and work with him have shaped me into the engineer I am today. He has truly been the epitome of what I believe it means to be an engineer, and will be the high watermark I strive to match for the rest of my days.

Thanks to Dr. Gary Coleman for literally teaching me how to build aircraft synthesis systems, without his patience and guidance I would not have had the knowledge or skill required to complete my research.

A special thank you to Amen Omoragbon and Amit Oza for allowing me to join them on this ridiculous research endeavor, and working tirelessly over the past 2 years to turn an idea on a whiteboard into a fully functioning system. Also, I would like to not thank Amit Oza for suggesting that I join them on this ridiculous research endeavor; you know what you did... I would like to thank Eric Haney, Brandon Watters, Reza Mansouri, Doug Coley, Thomas McCall, James Haley and all of the past and current members of the AVD Lab with whom I have had of the privilege of working, for their support and friendship throughout the course of my research.

I would also like to thank my parents Ruben, Letty, Mike and Mona, my sister Melissa, and all of my family and friends for their love and support through my lengthy (and seemingly never-ending) undergraduate and graduate career. I would like to express my appreciation to my brothers Ian, Jordan, and Tim, you’ve kept me sane in the chaos of the last few years. Additionally and especially, to my brother Alex, even in the most strenuous of circumstances you have always been the steady hand holding the lighter.

Of course, I would like to reserve the most special of thank yous and words of appreciation for my wife, Taryn. Thank you for the never-ending encouragement and belief, and for bringing me happiness in the darkest of times by simply reminding me to turn on the light. I would like to thank my beautiful daughter, Lena, for teaching me just how much potential a person really has, and how little sleep someone really needs to reach that potential.

August 11, 2016

In Memoriam

Lastly, I would like to dedicate this research to the most genuine and unique person I have ever known, the late Geoff Gibson. Calling him my friend has been one of the true honors in my life. The following is an excerpt from a story by Alex Lyle, describing the conclusion of a tradition and a good bye to our dear friend.

"I found the tent, and it still served coffee, only now it had expanded its menu to many other drinks and even food. The adjoining stage was now a place for up and coming artists lucky enough to show their stuff to impromptu passers-by. Determined to make this moment poignant, and give it the weight it deserved, I ventured in to the crowded tent and stood in line and eventually got two large piping hot coffees. I found a table with two empty seats, put the coffees on the table in front of them, and sat down in silence. After taking in the moment for a bit, I toasted Geoff's cup, and sipped mine away for the next half hour. An earthy girl in a dirty sundress began playing a folky ballad on guitar from the small stage, a scene I immediately knew Geoff would have obsessed over. As the young chanteuse's voice floated across the warm breeze, I situated Geoff's cup of cooling coffee down in the tall grass next to the table at the base of a sturdy tent pole. Accepting sadly that it was inevitably time to leave, I stood up to go just as a young girl walked up and motioned to the empty seat next to me and asked if anyone was sitting there.

'A friend of mine was. But he left....'

As I offered her the seat, which she quickly took and nodded thanks, I took one last look down at Geoff's cup nestled away from danger of being kicked or spotted. A large bug had curiously wandered up the side, and was peering over the rim down into the dark brown liquid, antennas waving in speculation. As I walked away, I knew a tradition had come full circle, and was now complete. "

Abstract

COMPLEX MULTIDISCIPLINARY SYSTEM COMPOSITION

FOR AEROSPACE VEHICLE CONCEPTUAL DESIGN

Lex Gonzalez, PhD

The University of Texas at Arlington, 2016

Supervising Professor: Bernd Chudoba

Although, there exists a vast amount of work concerning the analysis, design, integration of aerospace vehicle systems, there is no standard for how this data and knowledge should be combined in order to create a synthesis system. Each institution creating a synthesis system has in house vehicle and hardware components they are attempting to model and proprietary methods with which to model them. This leads to the fact that synthesis systems begin as one-off creations meant to answer a specific problem. As the scope of the synthesis system grows to encompass more and more problems, so does its size and complexity; in order for a single synthesis system to answer multiple questions the number of methods and method interface must increase.

As a means to curtail the requirement that the increase of an aircraft synthesis systems capability leads to an increase in its size and complexity, this research effort focuses on the idea that each problem in aerospace requires its own analysis framework. By focusing on the creation of a methodology which centers on the matching of an analysis framework towards the problem being solved, the complexity of the analysis framework is decoupled from the complexity of the system that creates it.

The derived methodology allows for the composition of complex multi-disciplinary systems (CMDS) through the automatic creation and implementation of system and disciplinary method interfaces. The CMDS Composition process follows a four step methodology meant to take a problem definition and progress towards the creation of an analysis framework meant to answer said problem. The unique implementation of the CMDS Composition process take user selected disciplinary analysis methods and automatically integrates them, together in order to create a syntactically composable analysis framework.

As a means of assessing the validity of the CMDS Composition process a prototype system (AVD^{DBMS}) has been developed. AVD^{DBMS} has been used to model the Generic Hypersonic Vehicle (GHV), an open source family of hypersonic vehicles originating from the Air Force Research Laboratory. AVD^{DBMS} has been applied in three different ways in order to assess its validity: Verification using GHV disciplinary data, Validation using selected disciplinary analysis methods, and Application of the CMDS Composition Process to assess the design solution space for the GHV hardware. The research demonstrates the holistic effect that selection of individual disciplinary analysis methods has on the structure and integration of the analysis framework.

Table of Contents

List of Illustrations

List of Tables

Chapter 1

Introduction and Objectives

The objectives of this study are best summarized in the words of Brockway McMillan, Under Secretary of the Air Force (McMillan 1964):

> *"The gap I refer to is the planning gap our failure to answer adequately the question I just asked … we don't spend enough time, energy, or talent in deciding how to deploy our technological resources in other words, in deciding what to develop out of the products of our research. Just as our research and development program must match the risks that we face in the international arena, so also must our planning of that program be commensurate with the commitments we are making. …How much effort should we expend to be sure we are committing these resources toward a product that we really need and one that we can really use?"*

The question of whether or not the aerospace problem being solved is of sufficient value to the stakeholders to warrant further investment should be the first question answered in any technology forecasting setting. The "gap" discussed by McMillan is directed at the disparate level of attention/resources given towards 'how' to solve a given aerospace problem as opposed to 'should' we solve a given aerospace problem. As a result, the impetus of this research will be to answer the question of **'how' to assess if a problem 'should' be solved.**

1.1 Introduction

Jackson (Jackson 1997) defines aircraft synthesis as "*the act of designing the aircraft or a segment of it. … Hence, synthesis is a collection of steps which occur throughout the systems engineering process*". Torenbeek (Torenbeek 2013) further defines aircraft design synthesis as an activity that includes:

a) An assessment of the enabling technologies required to comply with the design and certification requirements

b) Comparative studies to evaluate the implications of choosing different conceptual general arrangements of the design

c) Identification of the selection variables to be optimized in order to obtain an economically superior aircraft

In simpler terms, aircraft design synthesis is the evaluation of the level of vehicle performance needed in order to solve a given problem, and/or satisfy a problem-specific objective function. Aircraft synthesis tries to answer the question of 'how well, if at all' can you do the things you are required to do.

The product development life cycle is comprised of chronological phases detailing the evolution a vehicle from initial design to operation, see Figure 1-1. Although the design process takes place through the entirety of the product development life cycle, it has the greatest impact during the requirements definition and conceptual design phases.

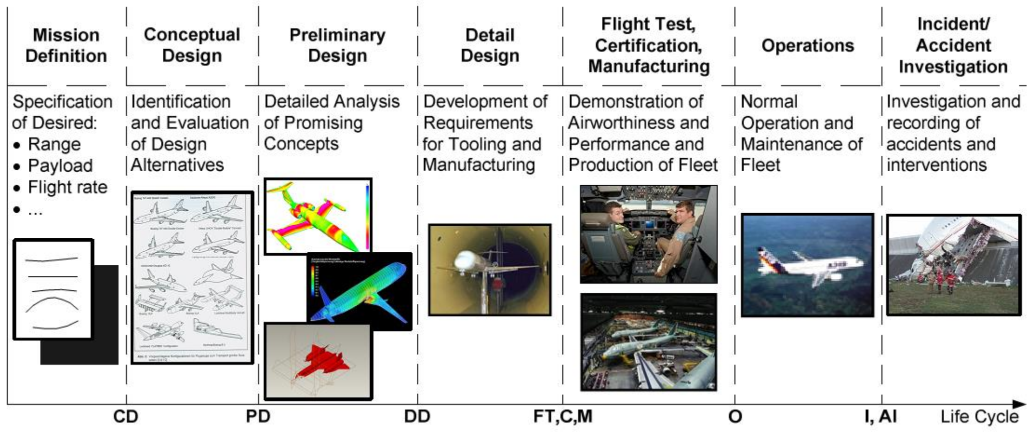

Mission Definition
Conceptual Design
Preliminary Design
Detail Design
Flight Test, Certification, Manufacturing
Operations
Incident/ Accident Investigation
Specification of Desired:
• Range
• Payload
• Flight rate
• ...
Identification and Evaluation of Design Alternatives
Detailed Analysis of Promising Concepts
Development of Requirements for Tooling and Manufacturing
Demonstration of Airworthiness and Performance and Production of Fleet
Normal Operation and Maintenance of Fleet
Investigation and recording of accidents and interventions
CD
PD
DD
FT,C,M
O
I, AI
Life Cycle

Figure 1-1 Aircraft Product Development Lifecycle (Omoragbon 2010)

This is due to the fact that the freedom to make design changes is greatest during these initial phases; however, there is a minimum of design data/knowledge available, see Figure 1-2. This leaves the aircraft designer in the position of having the most control over the direction of a vehicle design, whilst the least understanding of the problem he/she is trying to solve. The focus of the current research is the advancement of the capability of the synthesis specialist to analyze, and assess aerospace problems in early conceptual design.

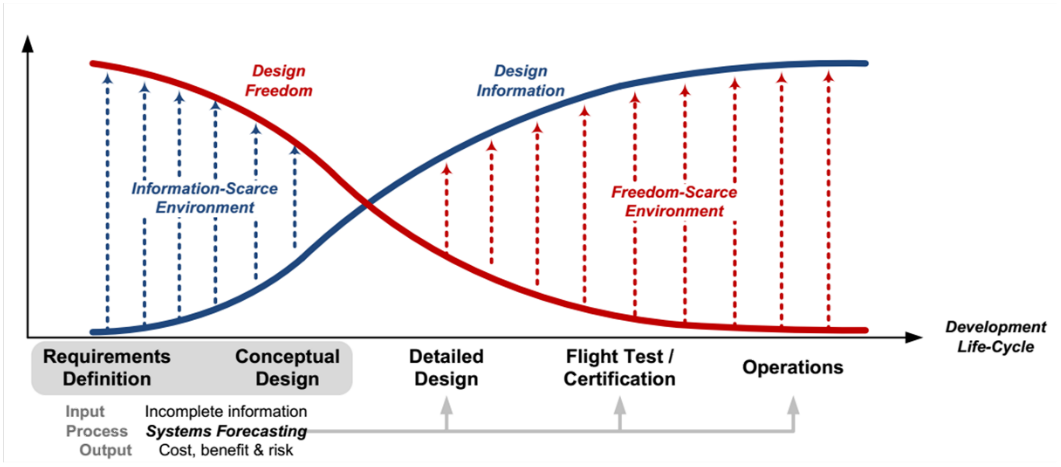


Figure 1-2 Aerospace Product Development Life Cycle (Haney 2016)

The goal of conceptual design is the assessment of the relationship between the problem being solved (design mission, operational constraints), and combinations of vehicle hardware technology (including technology performance assumptions) that solves it. The result of conceptual design should be an assessment of these combinations with respect to their ability to address the given problem requirements. This does not result in the choice of a specific vehicle concept, but rather highlights that one or more hardware concepts (combinations of hardware technologies) warrants further study, see Figure 1-3. To this end, it becomes imperative to be able to compare multiple combinations of hardware concepts to solve a given problem. The result is an analytical assessment of whether or not a solution to a given problem is feasible, and if so what combinations of design input parameters yield feasible solutions. The continuum of those feasible solutions makes up the design solutions space.

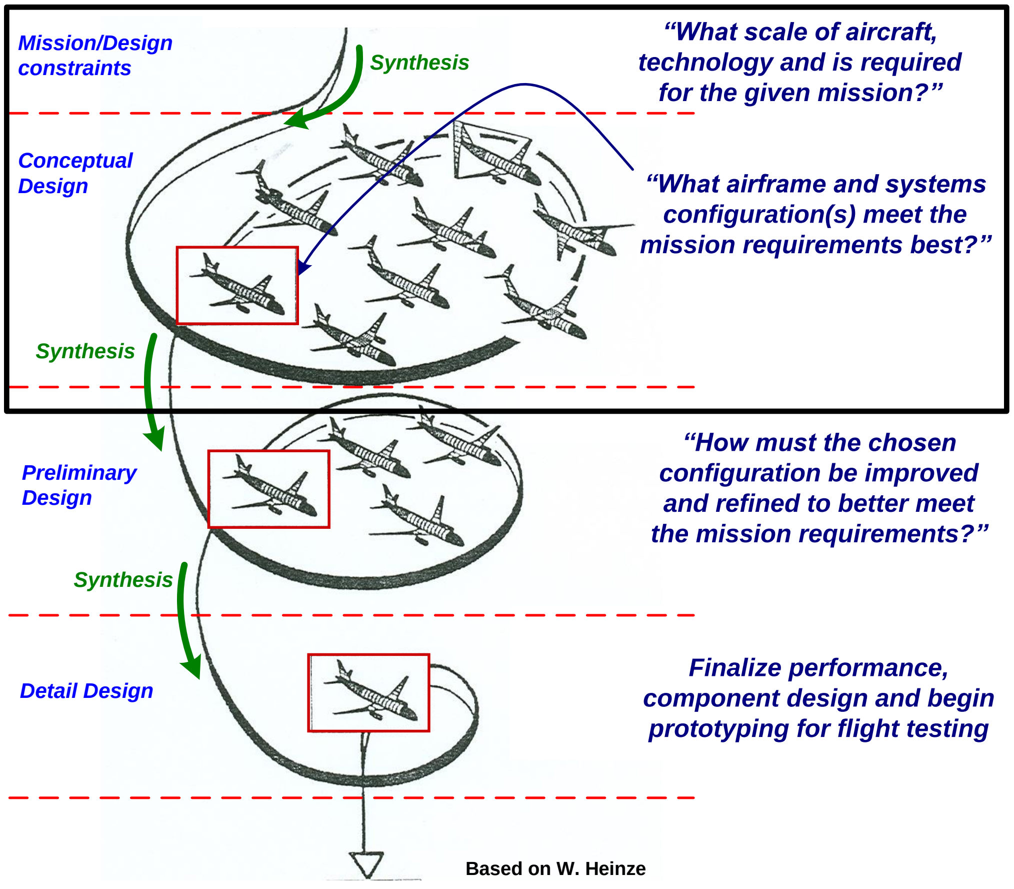

Mission/Design constraints
Synthesis
"What scale of aircraft, technology and is required for the given mission?"
Conceptual Design
"What airframe and systems configuration(s) meet the mission requirements best?"
Synthesis
Preliminary Design
"How must the chosen configuration be improved and refined to better meet the mission requirements?"
Synthesis
Detail Design
Finalize performance, component design and begin prototyping for flight testing
Based on W. Heinze

Figure 1-3 Example Aerospace Vehicle Design Process (Heinze 1994)

This type of analysis has historically been achieved through the use of aerospace synthesis systems, see Appendix A. These aircraft design tools have been created by institutions in both industry and academia, and attempt to provide insight into the effects specific design drivers have in a total vehicle context.

Although, there exists a vast amount of work concerning the analysis, design, integration of aerospace vehicle systems, there is no standard for how this data and knowledge should be combined in order to create a synthesis system. Each institution creating a synthesis system has in house vehicle and hardware components they are attempting to model and proprietary methods with which to model them. This leads to the fact that synthesis systems begin as one-off creations meant to answer a specific problem. As the scope of the synthesis system grows to encompass more and more problems, so does its size and complexity; in order for a single synthesis system to answer multiple questions the number of methods and method interface must increase.

Synthesis systems are comprised of disciplinary analysis modules that are run sequentially, where the outputs of a discipline may serve as inputs to one or more subsequent disciplines. Figure 1-4 shows an example of a Design Structure Matrix (DSM); a visualization showing the synthesis system in terms of its disciplinary analysis modules, as well as the multi-disciplinary connections between those modules.

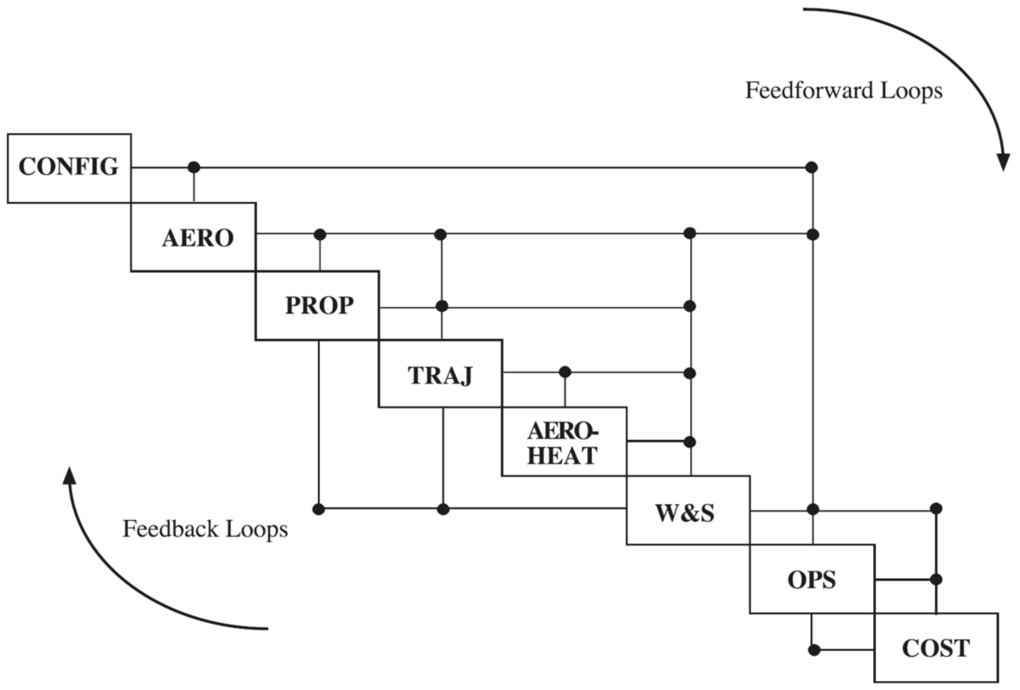


Figure 1-4 Design Structure Matrix for Hypersonic Launch Vehicle (Bradford 2001)

The example in Figure 1-4 is for a system designed to model a hypersonic launch vehicle. One important thing to note about the DSM is that the sequence of disciplinary modules, and interdisciplinary connections have been set in order to match a given problem. This means that a designer, in this case Bradford (Bradford 2001), has set up his system for hypersonic launch vehicle design. If another designer had attempted to create a system for this problem, or if the problem requirements had been adjusted, the resultant system would change. The ability for a system to be adaptable to classical and new/novel problems in aerospace means that it must be able to adjust the type and sequence of disciplinary modules, as well as the interdisciplinary relationships connecting them.

There are two ways for a synthesis system to obtain this adaptability: (1) Integrate all methods and method interfaces into a single system, (2) Create method and method interfaces for specific problems. The first option has been classically applied to aircraft synthesis systems. All method interfaces are defined apriori; every path through the synthesis system is pre-defined by the synthesis system programmer. This leads to the requirement that all data needed to define multidisciplinary integration be known by the programmer. Additionally, in this setting as methods and method interfaces are added the synthesis system will grow in size and complexity. The second option has been implemented more recently in aerospace synthesis systems. The method interfaces are created at run-time. This means that the synthesis system has a framework where methods can be chosen, and interfaces created based on those choices. In this setting the synthesis environment is providing methods for the user to choose, and once chosen is directing their integration into a single system; this is analogous to an orchestra composer directing not only which instruments should be playing at a given time, but also their tempo. An advantage to this setting is the fact that it is not required to add method interfaces when adding new methods to the system. Although this leads the overall system growing at a slower rate than the previous case, it does create a burden on the system to create method interfaces at run-time. This means that in order for this approach to be successfully implemented into a synthesis environment, there must be a methodology with the explicit purpose of defining and creating these interfaces.

1.2 Objectives

Every aerospace design problem is unique, containing specific design requirements and constraints. In order to account for this, an environment aimed at the composition of problem specific analysis frameworks is needed. The thesis objective then becomes the advancement of the state-of-the-art in aerospace conceptual design through the creation of a methodology for the composition of complex multi-disciplinary systems meant to solve specific problems in aerospace.

1.3 Research Strategy

First an historical review of aircraft synthesis system will be presented in order to familiarize the reader with past and present implementation characteristics. This review will focus on the ability of the synthesis system to generate new method interfaces at run-time and the level at which this capability exists in the system. Next a review on non-aerospace techniques related to the decomposition and composition of complex multidisciplinary systems will be presented. These techniques will then be applied to the aircraft synthesis problem and a methodology will be derived for the composition of complex multidisciplinary systems. This will lead to the creation of AVD^{DBMS}, a software tool for the composition of complex multidisciplinary systems for use in aerospace conceptual design. Finally, AVD^{DBMS} will be used to perform three case studies showing the adaptability of the tool and emphasizing the effect of Disciplinary Method selection on the overall analysis framework capability.

Chapter 2

Literature Survey and Objectives Refinement

The following sections review the current literature regarding aerospace synthesis. This has been done in an effort to define the current state of the art, leading to specifications for a system able to address the research objective presented at the end of Chapter 1. The initial review focuses on the research done on the characterization of aerospace synthesis systems done at the Aerospace Vehicle Design Laboratory. This review provides an accumulated listing of attributes required in a next generation aerospace synthesis system. This is followed by a survey of aerospace synthesis systems, both in academia and industry, with the goal being the comparison of these systems in the context of the previously defined next generation attributes. Finally, the outcome has been the specification of attributes for an aerospace synthesis setting with the capability to compose complex multidisciplinary systems for aerospace conceptual design.

2.1 Aerospace Synthesis System Characterization

Chudoba (Chudoba 2001), provides an assessment of aircraft synthesis systems, detailing specifically the change in modeling complexity as a function of time. He explains, “The classification scheme selected distinguishes the multitude of vehicle analysis and synthesis approaches according to their modeling complexity, thereby expressing their limitations and potential.” Five different classes (see Table 2-1) of flight vehicle design sophistications emerge, clearly distinguishing advances in knowledge and technology. The classes measure the chronological implementation and integration of design knowledge with computer automation in aerospace design.

Table 2-1 Classification of aerospace design synthesis approaches (Chudoba 2001)

Class	Design Definition	Develop Time	Characteristics
Class I	Early Dawn	Until 1905	Trial and error approach, experiment, no systematic methodology
Class II	Manual Design Sequence	1905 – 1955	Physical design transparency, parameter studies, standard aircraft design handbooks
Class III	Computer Automation	1955 – Today	Reduced design cycles, detailed exploration of the design space, discipline-specific software programs
Class IV	Multidisciplinary Integration	1960 – Today	Computerized design system, MDO, data sharing, centralizing design
Class V	Generic Design	Future Generation	Configuration independent, sophisticated design synthesis framework, detailed engineering analysis, synthesis of a user-defined aircraft, true inverse design capability, KBS

Classes I represents the early days in aerospace engineering; these systems are manual in implementation and rely heavily on trial and error experimentation. The empirical data resulting from Class I analysis are for the first time combined into manual design methodologies in Class II. It is at this point that the so-called “handbook design methodologies” are created. These design sequences are nominally guided through the use of integrated and sequential nomograms. A nomogram is a “a diagram representing the relations between three or more variable quantities by means of a number of scales, so arranged that the value of one variable can be found by a simple geometric construction, for example, by drawing a straight line intersecting the other scales at the appropriate values”. Example design sequences from this era include USAF Stability and Control DATCOM (Finck, Hoak, and Douglas Aircraft Company 1978), USAF Space Planners Guide (United States., Air Force.,Systems Command., 1965).

Class III begins the era of computer automation for disciplinary analysis. This era has been spurred by the advent of the multiprocessor and its availability to research scientists and engineers. The coding of specific Class II disciplinary methods (Lifting Line Theory, Panel Methods) are the first examples of computer automation in aircraft design. Lovell (Lovell 1980) comments, “Initial computer applications were confined to aspects of structural analysis and wing design. There was some resistance to the use of computers in initial project design because of the complex decision-making process involved. However, they enabled more detailed analyses to be made and hence allowed a greater range of carpet plots with additional overlays to be prepared to show the effects of configuration variables on performance." Although specific disciplinary analysis is automated, the synthesis of vehicle performance is still done manually. In this setting the automated disciplinary produces results in the form of carpet plot or lookup tables, the synthesis engineer then takes this data and manually integrates it together to assess total vehicle performance attributes.

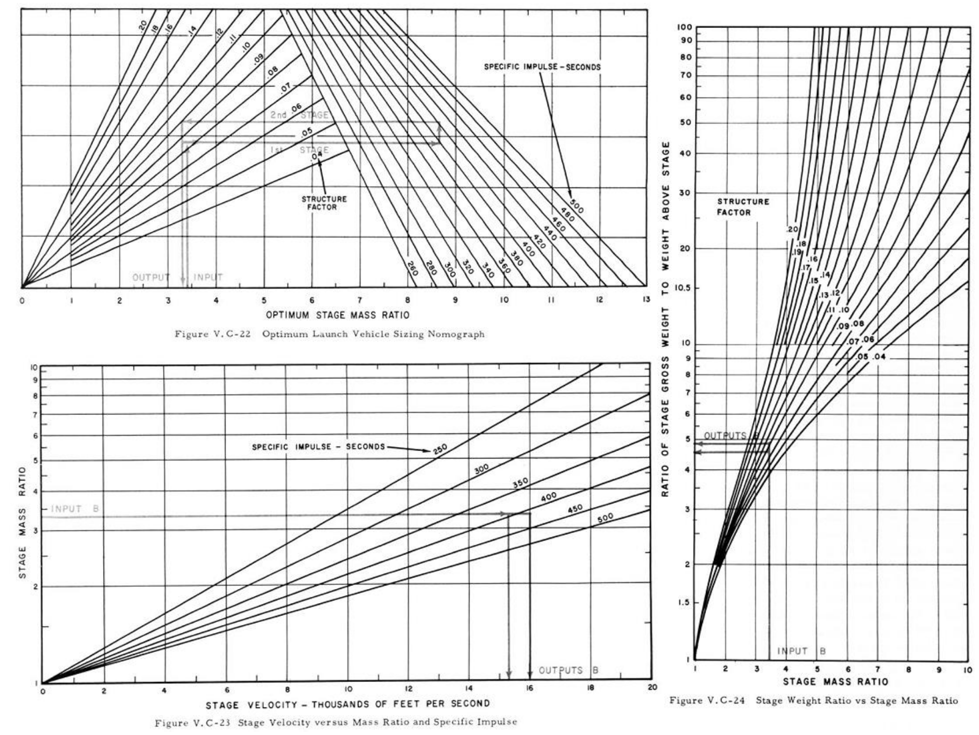
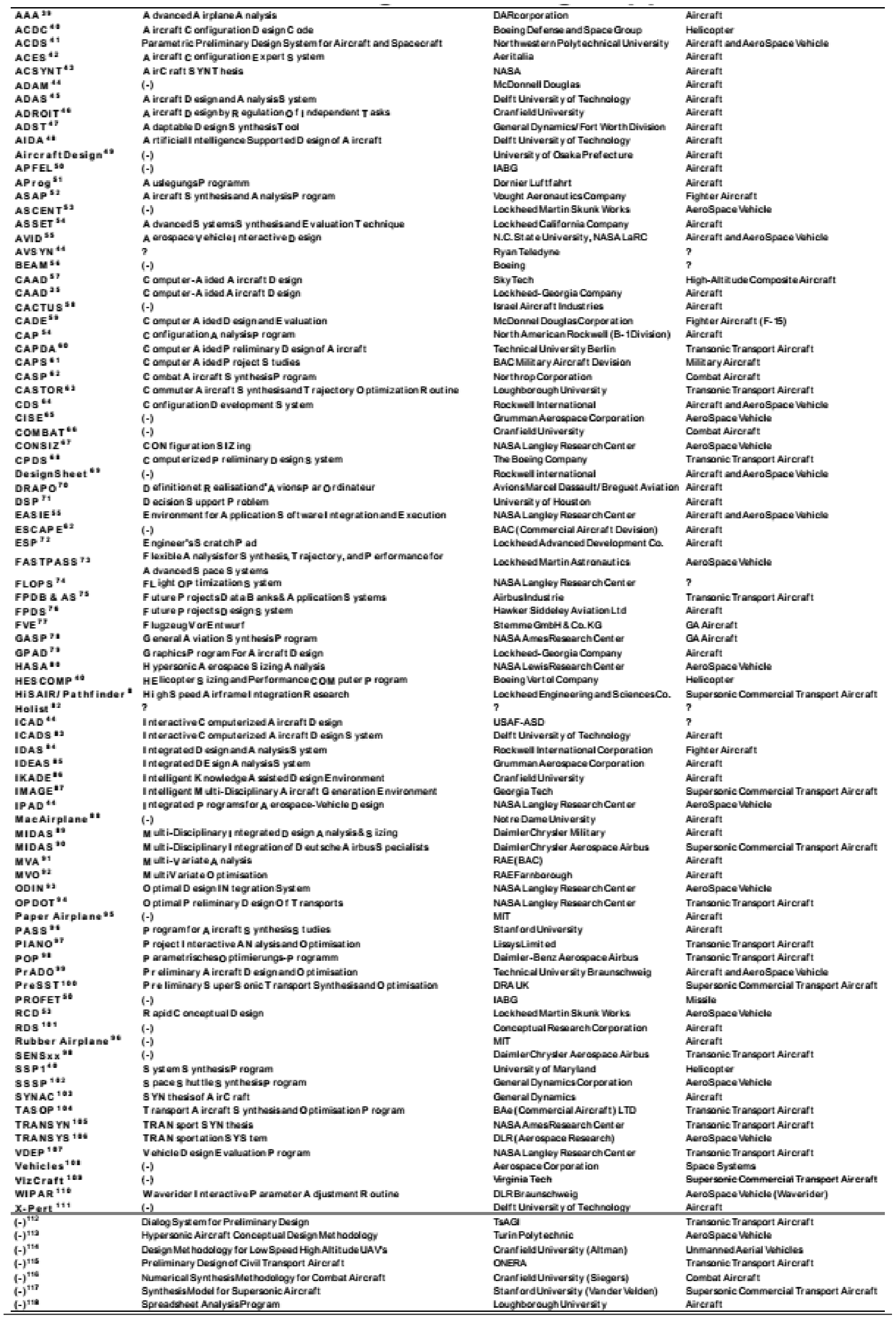


Figure 2-1 Example Nomogram (United States., Air Force.,Systems Command., 1965)

Classes IV systems are those that provide multidisciplinary integrations capability in a computer setting. Whereas, in Class III the results of disciplinary analysis are manually integrated by the designer, Class IV systems utilize computer generated interfaces in order to integrate disciplinary analysis into standalone multidisciplinary settings. Chudoba (Chudoba 2001) notes that the "*Development of more robust optimisation algorithms resulted in more complex design synthesis systems for conceptual design application*." Although this allows the aircraft designer to solve more and more complex problems, this capability comes at the expense of the design clarity found in Class II synthesis systems. A listing of "past and contemporary" Class IV system is presented in Table 2-2. Chudoba also notes that "… *advanced generations of computer systems have enabled the first steps towards true multi-dimensional (multi-point) optimisation capability, still with little physical insight into the multidisciplinary coupling effects*. …"

The result of this review and subsequent classification scheme has been the specification of the "*Class V – Generic Synthesis Capability*". This breakdown places emphasis on the integration of multi-disciplinary effects, and the use of dedicated methods libraries. It is important to note that Chudoba defines Class V Synthesis as a design process NOT a design tool; concluding that more emphasis should be placed on developing the capability of a synthesis system as opposed to the implementation of the tool itself. Chudoba specifies the attributes of a Class V sys as follows: Generic & Physical Methods, Life-Cycle Synthesis, Knowledgebase System, Multidisciplinary Optimization, Multi-Fidelity, Design Skill, Methods Library, Integrated People Management Process.

Table 2-2 Class IV Synthesis Systems (Chudoba 2001)



AAA[39]	Advanced Airplane Analysis	DARcorporation	Aircraft
ACDC[40]	Aircraft Configuration Design Code	Boeing Defense and Space Group	Helicopter
ACDS[41]	Parametric Preliminary Design System for Aircraft and Spacecraft	Northwestern Polytechnical University	Aircraft and AeroSpace Vehicle
ACES[42]	Aircraft Configuration Expert System	Aeritalia	Aircraft
ACSYNT[43]	AirCraft SYNThesis	NASA	Aircraft
ADAM[44]	(-)	McDonnell Douglas	Aircraft
ADAS[45]	Aircraft Design and Analysis System	Delft University of Technology	Aircraft
ADROIT[46]	Aircraft Design by Regulation Of Independent Tasks	Cranfield University	Aircraft
ADST[47]	Adaptable Design Synthesis Tool	General Dynamics/ Fort Worth Division	Aircraft
AIDA[48]	Artificial Intelligence Supported Design of Aircraft	Delft University of Technology	Aircraft
AircraftDesign[49]	(-)	University of Osaka Prefecture	Aircraft
APFEL[50]	(-)	IABG	Aircraft
AProg[51]	AuslegungsProgramm	Dornier Luftfahrt	Aircraft
ASAP[52]	Aircraft Synthesis and Analysis Program	Vought Aeronautics Company	Fighter Aircraft
ASCENT[53]	(-)	Lockheed Martin Skunk Works	AeroSpace Vehicle
ASSET[54]	Advanced Systems Synthesis and Evaluation Technique	Lockheed California Company	Aircraft
AVID[55]	Aerospace Vehicle Interactive Design	N.C. State University, NASA LaRC	Aircraft and AeroSpace Vehicle
AVSYN[44]	?	Ryan Teledyne	?
BEAM[56]	(-)	Boeing	?
CAAD[57]	Computer-Aided Aircraft Design	SkyTech	High-Altitude Composite Aircraft
CAAD[35]	Computer-Aided Aircraft Design	Lockheed-Georgia Company	Aircraft
CACTUS[58]	(-)	Israel Aircraft Industries	Aircraft
CADE[59]	Computer Aided Design and Evaluation	McDonnel Douglas Corporation	Fighter Aircraft (F-15)
CAP[54]	Configuration Analysis Program	North American Rockwell (B-1 Division)	Aircraft
CAPDA[60]	Computer Aided Preliminary Design of Aircraft	Technical University Berlin	Transonic Transport Aircraft
CAPS[61]	Computer Aided Project Studies	BAC Military Aircraft Devision	Military Aircraft
CASP[62]	Combat Aircraft Synthesis Program	Northrop Corporation	Combat Aircraft
CASTOR[63]	Commuter Aircraft Synthesis and Trajectory Optimization Routine	Loughborough University	Transonic Transport Aircraft
CDS[64]	Configuration Development System	Rockwell International	Aircraft and AeroSpace Vehicle
CISE[65]	(-)	Grumman Aerospace Corporation	AeroSpace Vehicle
COMBAT[66]	(-)	Cranfield University	Combat Aircraft
CONSIZ[67]	CONfiguration SIZing	NASA Langley Research Center	AeroSpace Vehicle
CPDS[68]	Computerized Preliminary Design System	The Boeing Company	Transonic Transport Aircraft
DesignSheet[69]	(-)	Rockwell international	Aircraft and AeroSpace Vehicle
DRAPO[70]	Définition et Réalisation d'Avions Par Ordinateur	Avions Marcel Dassault/ Breguet Aviation	Aircraft
DSP[71]	Decision Support Problem	University of Houston	Aircraft
EASIE[55]	Environment for Application Software Integration and Execution	NASA Langley Research Center	Aircraft and AeroSpace Vehicle
ESCAPE[62]	(-)	BAC (Commercial Aircraft Devision)	Aircraft
ESP[72]	Engineer's Scratch Pad	Lockheed Advanced Development Co.	Aircraft
FASTPASS[73]	Flexible Analysis for Synthesis, Trajectory, and Performance for Advanced Space Systems	Lockheed Martin Astronautics	AeroSpace Vehicle
FLOPS[74]	FLight OPtimization System	NASA Langley Research Center	?
FPDB & AS[75]	Future Projects Data Banks & Application Systems	Airbus Industrie	Transonic Transport Aircraft
FPDS[76]	Future Projects Design System	Hawker Siddeley Aviation Ltd	Aircraft
FVE[77]	Flugzeug Vor Entwurf	Stemme GmbH & Co. KG	GA Aircraft
GASP[78]	General Aviation Synthesis Program	NASA Ames Research Center	GA Aircraft
GPAD[79]	Graphics Program For Aircraft Design	Lockheed-Georgia Company	Aircraft
HASA[80]	Hypersonic Aerospace Sizing Analysis	NASA Lewis Research Center	AeroSpace Vehicle
HESCOMP[40]	HElicopter Sizing and Performance COMputer Program	Boeing Vertol Company	Helicopter
HiSAIR/ Pathfinder[8]	High Speed Airframe Integration Research	Lockheed Engineering and Sciences Co.	Supersonic Commercial Transport Aircraft
Holist[82]	?	?	?
ICAD[44]	Interactive Computerized Aircraft Design	USAF-ASD	?
ICADS[83]	Interactive Computerized Aircraft Design System	Delft University of Technology	Aircraft
IDAS[84]	Integrated Design and Analysis System	Rockwell International Corporation	Fighter Aircraft
IDEAS[85]	Integrated DEsign Analysis System	Grumman Aerospace Corporation	Aircraft
IKADE[86]	Intelligent Knowledge Assisted Design Environment	Cranfield University	Aircraft
IMAGE[87]	Intelligent Multi-Disciplinary Aircraft Generation Environment	Georgia Tech	Supersonic Commercial Transport Aircraft
IPAD[44]	Integrated Programs for Aerospace-Vehicle Design	NASA Langley Research Center	AeroSpace Vehicle
MacAirplane[88]	(-)	Notre Dame University	Aircraft
MIDAS[89]	Multi-Disciplinary Integrated Design Analysis & Sizing	DaimlerChrysler Military	Aircraft
MIDAS[90]	Multi-Disciplinary Integration of Deutsche Airbus Specialists	DaimlerChrysler Aerospace Airbus	Supersonic Commercial Transport Aircraft
MVA[91]	Multi-Variate Analysis	RAE (BAC)	Aircraft
MVO[92]	MultiVariate Optimisation	RAE Farnborough	Aircraft
ODIN[93]	Optimal Design INtegration System	NASA Langley Research Center	AeroSpace Vehicle
OPDOT[94]	Optimal Preliminary Design Of Transports	NASA Langley Research Center	Transonic Transport Aircraft
Paper Airplane[95]	(-)	MIT	Aircraft
PASS[96]	Program for Aircraft Synthesis Studies	Stanford University	Aircraft
PIANO[97]	Project Interactive ANalysis and Optimisation	Lissys Limited	Transonic Transport Aircraft
POP[98]	Parametrisches Optimierungs-Programm	Daimler-Benz Aerospace Airbus	Transonic Transport Aircraft
PrADO[99]	Preliminary Aircraft Design and Optimisation	Technical University Braunschweig	Aircraft and AeroSpace Vehicle
PreSST[100]	Preliminary SuperSonic Transport Synthesis and Optimisation	DRA UK	Supersonic Commercial Transport Aircraft
PROFET[50]	(-)	IABG	Missile
RCD[53]	Rapid Conceptual Design	Lockheed Martin Skunk Works	AeroSpace Vehicle
RDS[101]	(-)	Conceptual Research Corporation	Aircraft
Rubber Airplane[95]	(-)	MIT	Aircraft
SENSxx[98]	(-)	DaimlerChrysler Aerospace Airbus	Transonic Transport Aircraft
SSP1[40]	System Synthesis Program	University of Maryland	Helicopter
SSSP[102]	Space Shuttle Synthesis Program	General Dynamics Corporation	AeroSpace Vehicle
SYNAC[103]	SYNthesis of AirCraft	General Dynamics	Aircraft
TASOP[104]	Transport Aircraft Synthesis and Optimisation Program	BAe (Commercial Aircraft) LTD	Transonic Transport Aircraft
TRANSYN[105]	TRANsport SYNthesis	NASA Ames Research Center	Transonic Transport Aircraft
TRANSYS[106]	TRANsportation SYStem	DLR (Aerospace Research)	AeroSpace Vehicle
VDEP[107]	Vehicle Design Evaluation Program	NASA Langley Research Center	Transonic Transport Aircraft
Vehicles[108]	(-)	Aerospace Corporation	Space Systems
VizCraft[109]	(-)	Virginia Tech	Supersonic Commercial Transport Aircraft
WIPAR[110]	Waverider Interactive Parameter Adjustment Routine	DLR Braunschweig	AeroSpace Vehicle (Waverider)
X-Pert[111]	(-)	Delft University of Technology	Aircraft
(-)[112]	Dialog System for Preliminary Design	TsAGI	Transonic Transport Aircraft
(-)[113]	Hypersonic Aircraft Conceptual Design Methodology	Turin Polytechnic	AeroSpace Vehicle
(-)[114]	Design Methodology for Low Speed High Altitude UAV's	Cranfield University (Altman)	Unmanned Aerial Vehicles
(-)[115]	Preliminary Design of Civil Transport Aircraft	ONERA	Transonic Transport Aircraft
(-)[116]	Numerical Synthesis Methodology for Combat Aircraft	Cranfield University (Siegers)	Combat Aircraft
(-)[117]	Synthesis Model for Supersonic Aircraft	Stanford University (Van der Velden)	Supersonic Commercial Transport Aircraft
(-)[118]	Spreadsheet Analysis Program	Loughborough University	Aircraft

Huang's (Huang 2006) assessment of aerospace synthesis systems begins with the Class IV synthesis system, see Table 2-2, listing from Chudoba and focuses on their applicability to Space Access Vehicles (SAVs). Huang assesses 115 aerospace synthesis systems meant for the design of aircraft, helicopters, missiles and launch vehicles, and through a systematic evaluation process, provides an overview of each system and details its applicability towards the SAV problem, see Figure 2-2.

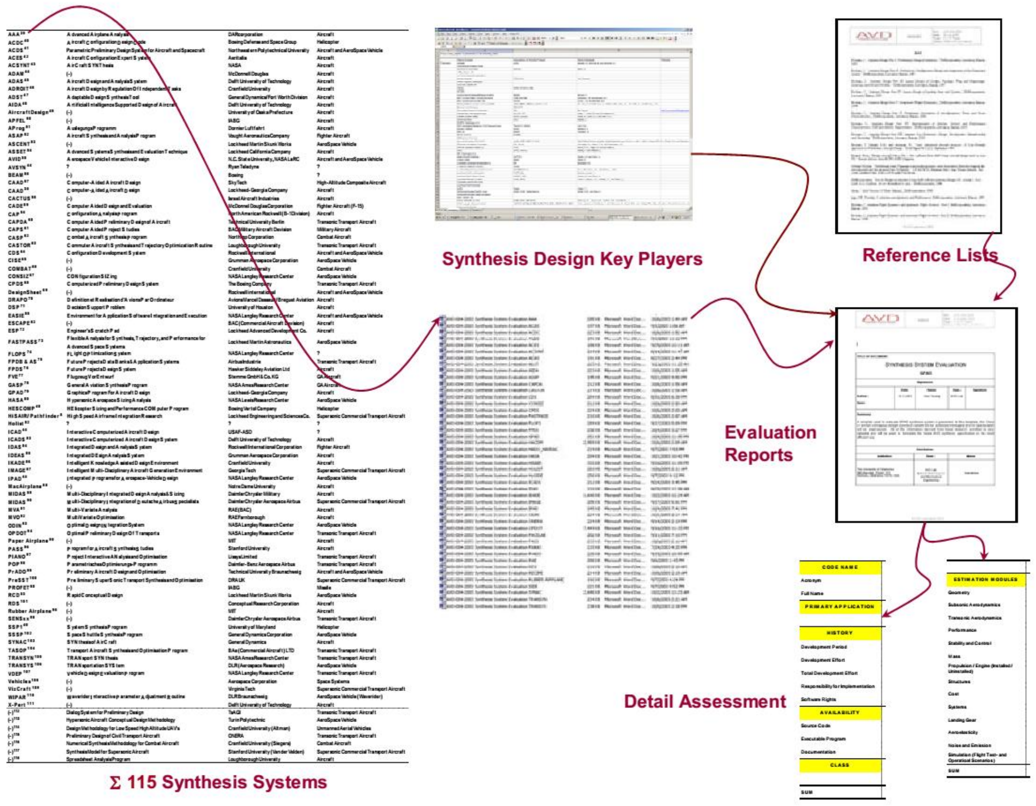
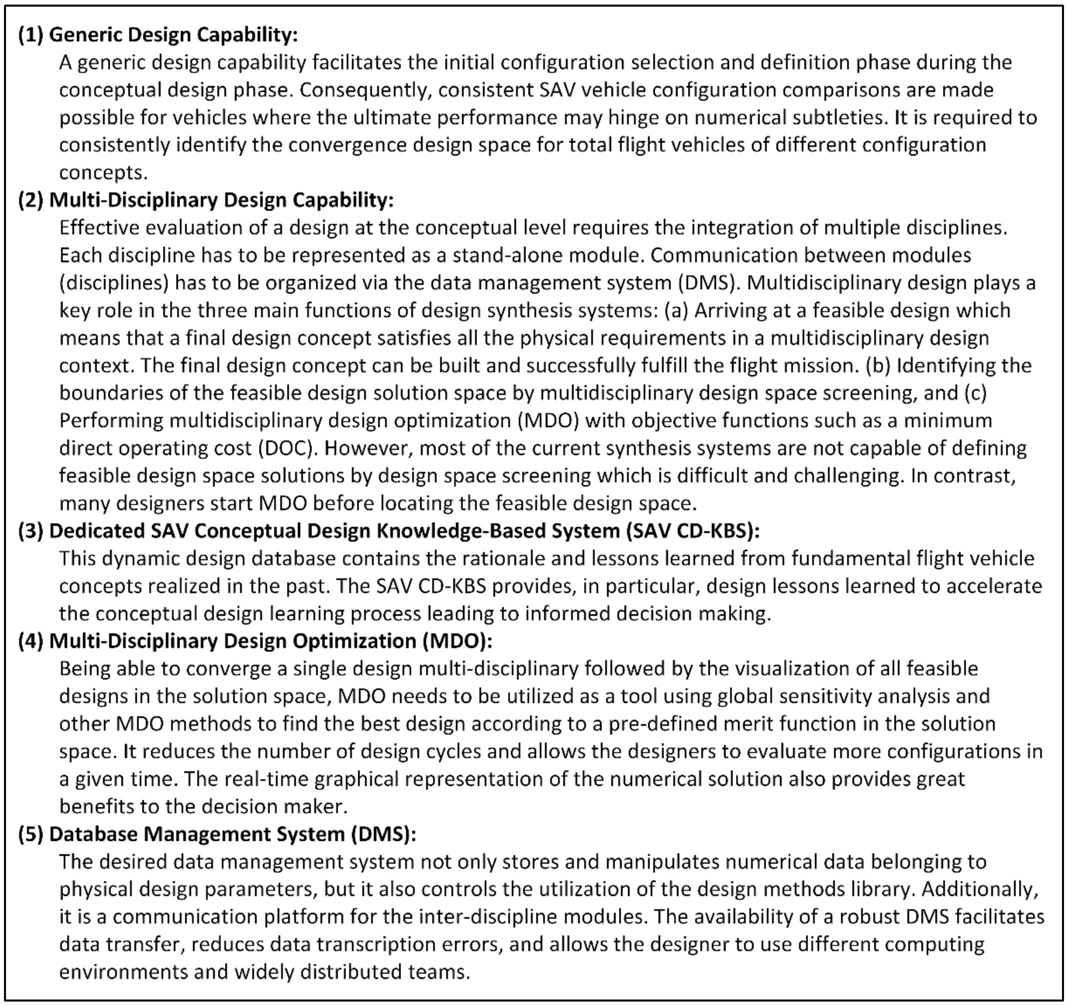


Figure 2-2 Evaluation Process of Design Synthesis Systems (Huang 2006)

Huang categorized each system according to its ability to perform the following: Mathematical Modelling, Multidisciplinary Analysis and Optimization, Knowledge-Based System, and Generic Concepts. The result showed a discrepancy in the ability of the then state of the art, circa 2004, to adequately address the SAV problem in the early stages of conceptual design. This led him to the following specifications for a synthesis system for Space Access Vehicles, see Figure 2-3.



(1) Generic Design Capability:
A generic design capability facilitates the initial configuration selection and definition phase during the conceptual design phase. Consequently, consistent SAV vehicle configuration comparisons are made possible for vehicles where the ultimate performance may hinge on numerical subtleties. It is required to consistently identify the convergence design space for total flight vehicles of different configuration concepts.

(2) Multi-Disciplinary Design Capability:
Effective evaluation of a design at the conceptual level requires the integration of multiple disciplines. Each discipline has to be represented as a stand-alone module. Communication between modules (disciplines) has to be organized via the data management system (DMS). Multidisciplinary design plays a key role in the three main functions of design synthesis systems: (a) Arriving at a feasible design which means that a final design concept satisfies all the physical requirements in a multidisciplinary design context. The final design concept can be built and successfully fulfill the flight mission. (b) Identifying the boundaries of the feasible design solution space by multidisciplinary design space screening, and (c) Performing multidisciplinary design optimization (MDO) with objective functions such as a minimum direct operating cost (DOC). However, most of the current synthesis systems are not capable of defining feasible design space solutions by design space screening which is difficult and challenging. In contrast, many designers start MDO before locating the feasible design space.

(3) Dedicated SAV Conceptual Design Knowledge-Based System (SAV CD-KBS):
This dynamic design database contains the rationale and lessons learned from fundamental flight vehicle concepts realized in the past. The SAV CD-KBS provides, in particular, design lessons learned to accelerate the conceptual design learning process leading to informed decision making.

(4) Multi-Disciplinary Design Optimization (MDO):
Being able to converge a single design multi-disciplinary followed by the visualization of all feasible designs in the solution space, MDO needs to be utilized as a tool using global sensitivity analysis and other MDO methods to find the best design according to a pre-defined merit function in the solution space. It reduces the number of design cycles and allows the designers to evaluate more configurations in a given time. The real-time graphical representation of the numerical solution also provides great benefits to the decision maker.

(5) Database Management System (DMS):
The desired data management system not only stores and manipulates numerical data belonging to physical design parameters, but it also controls the utilization of the design methods library. Additionally, it is a communication platform for the inter-discipline modules. The availability of a robust DMS facilitates data transfer, reduces data transcription errors, and allows the designer to use different computing environments and widely distributed teams.

Figure 2-3 Specification Synthesis System AVDS-SAV (Huang 2006)

Of note in Figure 2-3 is the inclusion of a ‘Database Management System’. This addition to the “Class V Synthesis” specification reveals the necessity of the system to not only connect design parametric data but to also “control utilization of the design methods library”. This insight leads to the idea that Huang’s specification encompasses not only the integration of multi-disciplinary effects, but also the integration of parametric data and design methods into the system.

Coleman (Coleman 2010), investigates synthesis systems applicable to early conceptual design. He segments aircraft conceptual design into three chronological steps namely: Parametric Sizing, Configuration Layout, and Configuration Evaluation, see Figure 2-4.

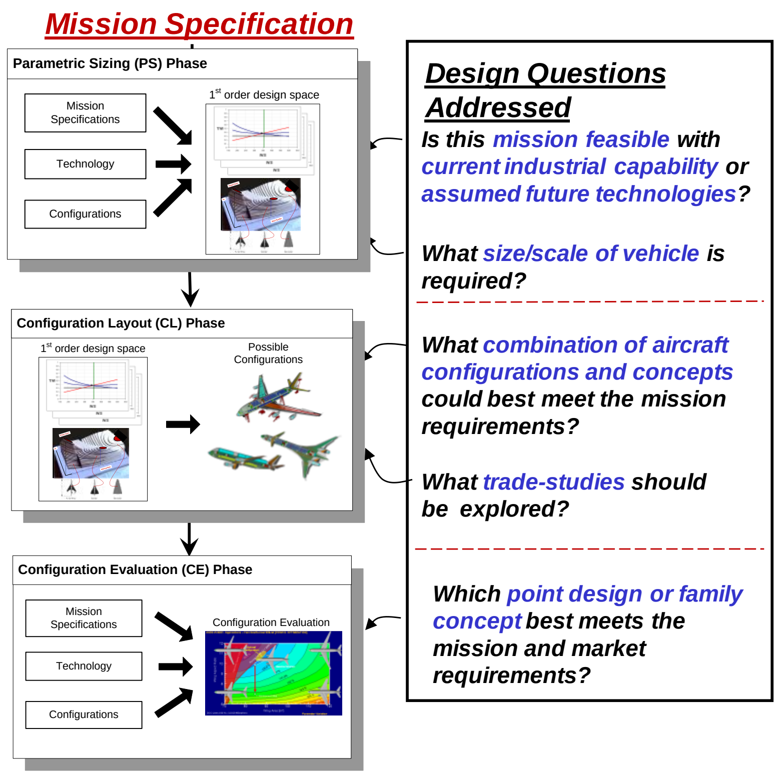

Mission Specification
Parametric Sizing (PS) Phase
Mission Specifications
Technology
Configurations
1st order design space
Configuration Layout (CL) Phase
1st order design space
Possible Configurations
Configuration Evaluation (CE) Phase
Mission Specifications
Technology
Configurations
Configuration Evaluation
Design Questions Addressed
Is this mission feasible with current industrial capability or assumed future technologies?
What size/scale of vehicle is required?
What combination of aircraft configurations and concepts could best meet the mission requirements?
What trade-studies should be explored?
Which point design or family concept best meets the mission and market requirements?

Figure 2-4 Fundamental steps to Aerospace Vehicle Conceptual Design (Coleman 2010)

After reviewing systems meant for each step on the conceptual design process Coleman shows that, “the first step in aircraft conceptual design, parametric sizing, has stagnated or has been ignored in the current literature”. This deficiency is in contrast to the importance Parametric Sizing has in the context of the product development life-cycle (see Figure 1-2). This leads to a specification of objectives meant to increase the then state of the art in parametric sizing for aerospace vehicles: (1) Development of a conceptual design process library, (2) Development of a conceptual design parametric sizing methods library, (3) Development of an integrated and flexible parametric sizing program based on the process and methods library.

2.1.1 Process Library Description

The process library assembled by Coleman is a Microsoft Word document providing information for a collection of aerospace synthesis systems. Each entry contains two parts:

d) Nassi-Schneiderman (NS) diagram (Figure 2-5)

e) Standardized process card (Table 2-3)

The NS diagram provides a standardized condensed form visualization of the logic of a system; including the sequence analytical modules, top-level Boolean operations, and system level deliverables. This provides a method for quickly visualizing complex multi-disciplinary systems. In addition to the standard NS structure, Coleman has added a color scheme to the flow chart, distinctly showing the conceptual design phase applicability for each module.

The process card is separated into three section: Process Overview, Application of Process, and Interpretation. The overview section contains indexing information including authors, publication date (both current and initial), and published reference. The application of processes section provides context towards when and where the process should be used. The last section, Interpretation, discusses how well the process answers the problem it was intended to solve.

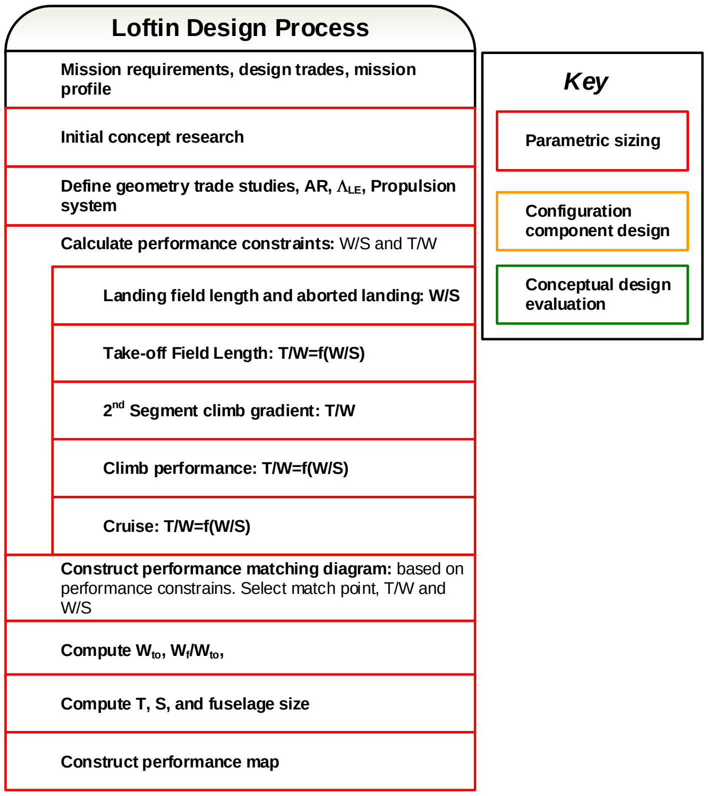

Loftin Design Process
Mission requirements, design trades, mission profile
Initial concept research
Define geometry trade studies, AR, Λ_{LE}, Propulsion system
Calculate performance constraints: W/S and T/W
Landing field length and aborted landing: W/S
Take-off Field Length: T/W=f(W/S)
2nd Segment climb gradient: T/W
Climb performance: T/W=f(W/S)
Cruise: T/W=f(W/S)
Construct performance matching diagram: based on performance constrains. Select match point, T/W and W/S
Compute W_{to}, W_f/W_{to},
Compute T, S, and fuselage size
Construct performance map
Key
Parametric sizing
Configuration component design
Conceptual design evaluation

Figure 2-5 Nassi-Schneiderman diagram for the Loftin design process (Coleman 2010)

The combination of process card and NS diagram together show the constituents of the synthesis system, visualize the connections between them, and allow for judgement of their applicability to differing problem types. An interesting aspect of this type of standardization is the highlighting of common elements between design processes meant for differing vehicle types created in widely different environments. The view of the process in this fashion also allows for the separation of the analytic process from the analytic methods, allowing for a fundamentally modular view of the system.

Table 2-3 Example Process overview card (Coleman 2010)

Processes Overview			
Design Phases Conceptual Design	**Author** Loftin	**Initial Publication Date** 1980	**Latest Publication Date** 1980
Reference: Loftin, L., “Subsonic Aircraft: Evolution and the Matching of Sizing to Performance,” NASA RP1060, 1980			
Application of Processes			
Applicability Primarily focused on parametric sizing of jet powered transports and piston powered general aviation aircraft			
Objective of Processes Determine an approximate size and weight the aircraft to complete the mission from a 1st level approximation of the design solution space			
Initial Start Point The processes begins with mission specification, possible configurations and fixed design variables such as AR.			
Description of basic execution From the mission specification statistics and basic performance relationships are used to determine relationships between T/W and W/S (Performance matching). The aircraft is then sized around this match point			

Interpretation		
CD steps Parametric Sizing	**Synthesis Ladder** Analysis Integrate Iteration of design Visualize design space	**Similar Procedures** Roskam (preliminary sizing) Torenbeek (Cat 1 methods)
General Comments: One of the first published processes utilizing performance matching Where Nicolai compares T/W and W/S after the complete convergence and interaction of the processes, Loftin derives basic relationships between T/W up front to visualize the solution space before intial sizing. Loftin essential short cuts the Nicolai approach to derive an initial design space rather than an initial configuration.		

2.1.2 Methods Library Description

The Methods Library is a Microsoft Word document consisting of disciplinary methods found as either parts of a synthesis system, or as standalone analytic methods found in literature. Each entry in the library is comprised of a Method overview card, see Table 2-4. The overview card contains four types of information (Coleman 2010):

1. Assumptions – detailing all simplifying assumptions used in method
2. Applicability – application validity (configuration/technology packages)
3. Basic Procedure – detailing input requirements, basic analysis procedure and outputs
4. Experience – documentation of design application and lessons learned in terms of accuracy, computation time and general comments

The accumulated disciplinary methods library, allows for the documentation and storage of design experience/knowledge in a centralized location. This results in the ability of the designer to choose which method is best suited for their given problem.

Table 2-4 Example Methods overview card (Coleman 2010)

Method Overview				
Discipline	**Design Phase**	**Method Title**	**Categorization**	**Author**
Aerodynamics	Parametric Sizing	Initial Drag polar estimation	Semi-Empirical	Roskam
Reference: Roskam, J., “Airplane Design Part I: Preliminary Sizing of Airplanes,” DARcorporation, Lawrence, Kansas, 2003				
Brief Description The drag polar is constructed using empirical relationships for parasite drag (based on gross weight), flap and landing gear effects. A classical definition of induced drag is used.				

Assumptions	**Applicability**
Increments of flap and landing gear taken from typical values Parasite drag coefficient is a function of take-off gross weight	Homebuilt aircraft propeller aircraft, single engine propeller aircraft, twin engine propeller aircraft, agricultural aircraft, business jets, regional turboprop aircraft, transport jets, military trainers, fighters, military patrol, bomb and transport, flying boats, supersonic cruise aircraft

Execution of Method
Input Mission profile, type of aircraft, take-off gross weight, AR, e, S estimate
Analysis description Estimate $S_{wet}=f(W_{TO})$ empirical based on type of aircraft Fig 3.22 Estimate $f=f(S_{wet})$ empirical based on type of aircraft Fig 3.21 Assume average value of S Select Flap and landing gear effects for each mission segment Table 3.6 $C_D = f / S + \Delta C_{Dflap} + \Delta C_{DLG} + \frac{C_L^2}{\pi AR \cdot e}$ Assume C_{Lmax} values from Table 3.1
Output: Drag Polar

Experience		
Accuracy	**Time to Calculate**	**General Comments**
Unknown	Unknown	

2.2 Survey of Synthesis Systems in Terms of System Capability

A review of past and present aerospace synthesis systems by Chudoba, Huang and Coleman has provided specifications for capabilities needed by future systems. The previous studies have focused on assessing the ability of a given synthesis system to analyze aerospace problems, as well as the level of disciplinary integration present in said analysis. The current survey benefits from these previous surveys and supplements their finding by attempting to characterize both the mechanism used by each synthesis system to interface disciplinary methods, as well as the ability of each system to create/integrate new disciplinary methods and disciplinary method interfaces.

2.2.1 Review Criterion

The review has been centered on assessing the System Capability of aerospace synthesis systems. For the purpose of the review System Capability has been defined as the capability of synthesis systems to characterize, analyze, and solve classical and new/novel aerospace problems. The categories and subcategories found in Table 2-5 are direct results of, and/or adaptations of the conclusions found in the previous section. The exception to this is section 5.b ‘Data management capability’. The criterion for this section have been derived with the objective of characterizing disciplinary method interfaces in terms of the database management system employed for each synthesis system.

Table 2-5 Literature Survey Criteria – System Capability

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

2.2.2 Representative Synthesis Systems

The synthesis systems reviewed using the criterion detailed in the previous section are listed in Table 2-6 and Table 2-7. Table 2-6 represents by-hand aircraft design processes classically found in design text books and short courses. Table 2-7 represents computer-based synthesis systems. The selected systems range from those developed for use in academia to industry. The listing of both by-hand and computer-based synthesis systems is meant to be a representative cross section of aircraft conceptual design methodologies. A comprehensive listing of the synthesis systems reviewed by Chudoba, Huang and Coleman can be found in Chapter 1.

Table 2-6 Selected By-Hand Synthesis Methodologies

Author	Year	Title
Corning	1979	Supersonic and Subsonic, CTOL and VTOL, Airplane Design
Howe	2000	Aircraft Conceptual Design Synthesis
Jenkinson	1999	Civil Aircraft Design
Loftin	1980	Subsonic Aircraft: Evolution and the Matching of Size to Performance
Nicolai	2010	Fundamentals of aircraft and airship design Volume 1, Aircraft design
Raymer	1999	Aircraft Design: A Conceptual Approach
Roskam	2004	Airplane Design, Parts I-VIII
Schaufele	2000	The Elements of Aircraft Preliminary Design
Stinton	1998	The Anatomy of the Airplane
Torenbeek	1982	Synthesis of Subsonic Airplane Design
Wood	1963	Aerospace Vehicle Design Vol. 1, Aircraft Design

Table 2-7 Selected Computer-Based Synthesis Systems

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
pyOPT	2012-	Python-based object-oriented framework for nonlinear constrained optimization	Royal Military College of Canada
PrADO	1986-	Preliminary Aircraft Design and Optimisation	Technical University Braunschweig
VDK/HC	2001	VDK/Hypersonic Convergence	McDonnell Douglas, Hypertec

A note should be made on the inclusion of the Model Center platform in the review of computer-based synthesis systems. Model Center is an integration platform for aerospace analytic legacy codes. This means that it is fundamentally different from a classical aircraft design codes. ‘Out of the box’ Model Center does not have any specific aerospace disciplinary methods or design processes. What it has is an open platform able to connect disciplinary methods and modules together. It has been included as this type of setting has distinct advantages over classical design codes in terms of its ability to adapt to new problems. From the outset, integration platforms are meant to allow to user to use his/her own disciplinary methods and allows them to be integrated into user-defined analytic processes. This analytic freedom comes with added requirements on the user to know exactly what he plans to implement, and how everything should be connected.

2.2.3 System Capability

The System Capability is a measure of the ability of a synthesis system to characterize, analyze, and solve classical and new/novel aerospace problems. The following six sections follow the six categories found in Table 2-5. Each subsection describes a distinct capability or specification. The result of this analysis has been a visualization of where the current state of the art in aerospace conceptual design stands, what is done well, and where opportunities may exist with further research and refinement.

2.2.3.1 Integration & Connectivity

The first section of the review assesses the capability of each synthesis system to analyze hardware independently while taking into account the multidisciplinary effects a hardware component has on the vehicle, see Figure 2-6.

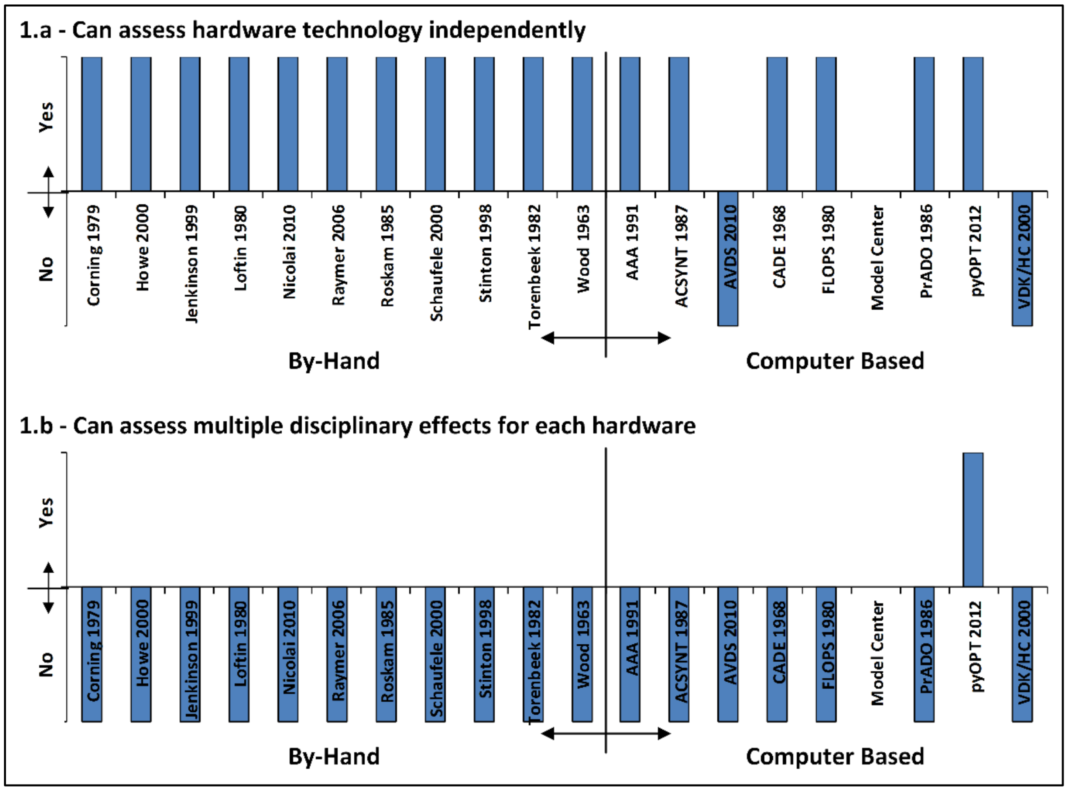


Figure 2-6 System Capability – Integration & Connectivity

a) Can assess hardware technology independently. Does the synthesis have the capability to assess hardware on their own; can a specific hardware component be run outside of the vehicle synthesis analysis (e.g. running the engine alone to create uninstalled thrust and fuel consumption maps)?

b) Can assess multiple effects for each hardware. Can the system, while looking at a piece of hardware independently take into account multiple disciplinary analyses (e.g. analyzing the engine alone to create propulsion performance maps, aerodynamics effects look-up tables, engine weight and volume estimation, etc.)?

One important thing to note, this assessment has been made assuming only hardware components already existing in the system are used. Figure 2-6 shows that the by-hand methods all have the capability to analyze individual hardware components outside of the synthesis process loop. This is a more of a function of the design environment not being assigned or locked down when doing the analysis by-hand. When doing by hand calculations the designer can calculate analyses independently of the synthesis loop, computer-based systems have a prescribed order of operations that must be followed. Due to this, in order for a computer-based system to have this capability it must have an analysis framework which adapts the order of operations to match the input of the user. AVDS and VDK/HC are not set up to run individual hardware performance outside of the main design loop. In these instances, it is necessary to run the full synthesis design loop and look into the individual hardware performance afterwards.

While many of the systems have the capability to assess the performance of an individual hardware component, that analysis is tied to a specific discipline. For example, the PrADO system can run propulsion disciplinary performance analysis of the engine alone but it cannot calculate the aerodynamic effect (e.g. ΔCL and ΔCD caused by the engine hardware and subsequent integration scheme) of the engine alone. In order to see these multidisciplinary effects, the full synthesis loop has to be analyzed. A reason for this is that the plug and play type of multidisciplinary analysis necessitates its own process separate from the synthesis loop. The system would have to take into account the input requirements for each of the separate disciplinary analysis module and create new links between them. Only one of the systems surveyed, pyOPT, has the capability to run user specified multidisciplinary analysis for a specific hardware component. pyOPT adjusts its analytic process through the use of object-oriented programming to set a structured interface between different types of information.

2.2.3.2 Interface Maturity

This section of the review assesses the capability of each synthesis system to combine hardware pieces together and analyze their multidisciplinary effects, see Figure 2-7.

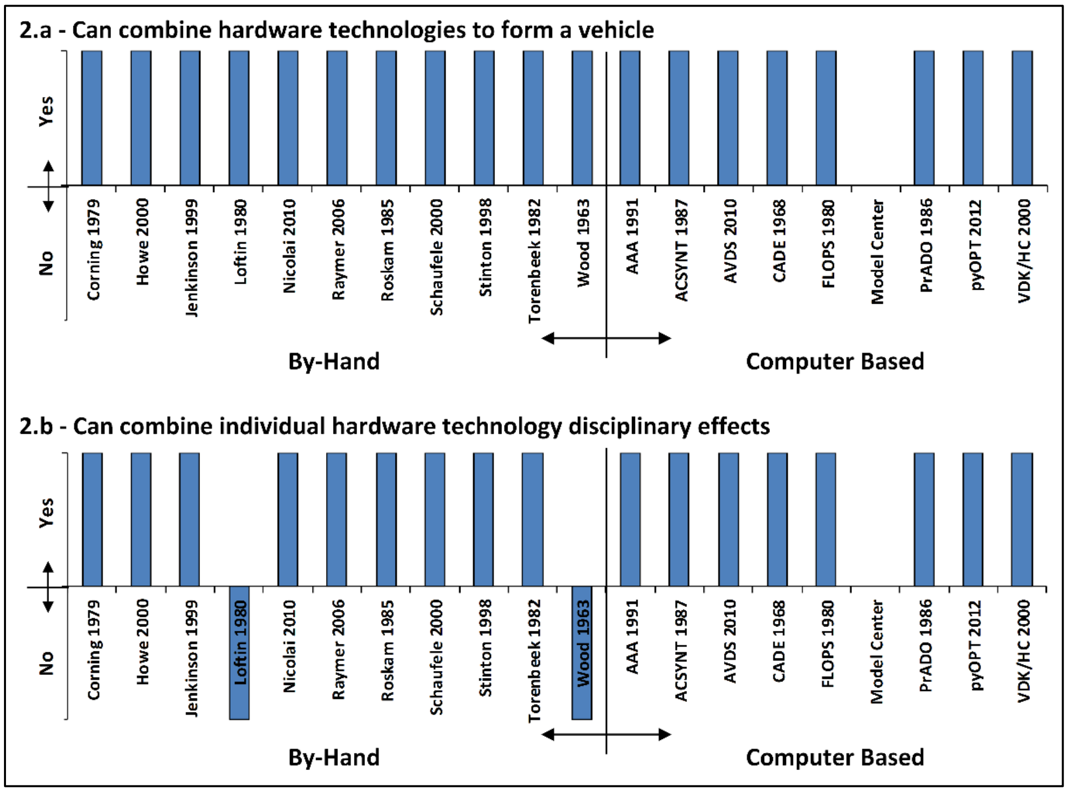


Figure 2-7 System Capability – Interface Maturity

a) Can combine hardware technologies to form a vehicle – Can the system hardware be defined as combinations of hardware components, where each component has its own specification and associated attributes (e.g. Vehicle = Delta wing + turbojet + … + etc. or Turbojet = Inlet + Compressor + …+ etc.)?

b) Can combine hardware technology disciplinary effects – Can the system combine the disciplinary effects of several hardware components to calculate composite hardware performance (e.g. $C_L = C_{l_{wing}} + C_{L_{Tail}} + C_{l_{engine}} + \cdots + etc.$)?

Figure 2-7 shows that all of the by-hand and computer-based systems have the capability to use a buildup methods for vehicle hardware. Each of the systems surveyed represent the vehicle as a composition of hardware pieces.

All of the systems surveyed can combine the effects of hardware pieces to solve for the total vehicle effect. The computer-based systems are more apt for this type of analysis, because buildup capability can be built in for disciplinary analyses of specific hardware types. AAA, AVDS, FLOPS, PrADO, and pyOPT use contributions of individual hardware components to construct total vehicle aerodynamic coefficients. FLOPS, PrADO and pyOPT have the capability to do the same type of buildup for the propulsion system and subsequent disciplinary propulsion analysis. Both Loftin and Wood lack the ability to represent a vehicle disciplinary effect as a composition of individual hardware effects. This is a result of both of these methodologies using solely empirically based statistical methods for their disciplinary analysis.

2.2.3.3 Scope of Applicability

This section of the review assesses the range of applicability of each synthesis system to product type, and phase in the product development life-cycle, see Table 2-8 and Table 2-9.

In both Table 2-8 Table 2-9 Model Center is shown as “N/A”. This result is a function of the ‘out of the box’ functionality present in the Model Center System. As discussed previously, Model Center is an integration platform, meaning that it does not contain a synthesis system analytic process. It is left to the user to use the integration framework to create the analytic process for your given problem. Therefore, Model center has no set applicability toward a specific conceptual design step or to specific products.

Table 2-8 System Capability – Scope of Applicability (CD Phase)

	By-Hand											Computer									
	Corning 1979	Howe 2000	Jenkinson 1999	Loftin 1980	Nicolai 2010	Raymer 2006	Roskam 1985	Schaufele 2000	Stinton 1998	Torenbeek 1982	Wood 1963	AAA 1991	ACSYNT 1987	AVDS 2010	CADE 1968	FLOPS 1980	Model Center	PrADO 1986	pyOPT 2012	VDK/HC 2000
Parametric Sizing	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes
Configuration Layout	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	No	No
Configuration Evaluation	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	No	Yes	Yes	No
N/A	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	Yes	No	No	No

Table 2-9 System Capability – Scope of Applicability (Product)

	By-Hand											Computer								
	Corning 1979	Howe 2000	Jenkinson 1999	Loftin 1980	Nicolai 2010	Raymer 2006	Roskam 1985	Schaufele 2000	Stinton 1998	Torenbeek 1982	Wood 1963	AAA 1991	ACSYNT 1987	AVDS 2010	CADE 1968	FLOPS 1980	Model Center	PrADO 1986	pyOPT 2012	VDK/HC 2000
Homebuilt	No	No	No	Yes	No	Yes	Yes	No	Yes	No	Yes	Yes	No	No	No	Yes	No	Yes	Yes	No
Single Engine	No	No	No	Yes	No	Yes	Yes	No	Yes	No	Yes	Yes	No	No	No	Yes	No	Yes	Yes	No
Twin Engine	No	No	No	Yes	No	Yes	Yes	No	Yes	No	Yes	Yes	No	No	No	Yes	No	Yes	Yes	No
Agricultural	No	No	No	Yes	No	Yes	Yes	No	Yes	No	Yes	Yes	No	No	No	Yes	No	Yes	Yes	No
Business Jet	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes	Yes	No
Regional TBP's	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes	Yes	No
Transport Aircraft	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes	Yes	No
Mil. Trainers	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No
Fighters	No	No	No	No	Yes	Yes	Yes	No	No	No	No	Yes	Yes	No	Yes	Yes	No	No	Yes	No
Mil. Patrol, bombers, transport	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No
Flying boats, Amphibious	No	No	No	No	No	Yes	Yes	No	No	No	No	Yes	No	No	No	No	No	No	Yes	No
Supersonic Cruise	Yes	No	No	No	Yes	Yes	Yes	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	No
Hypersonic P2P	No	No	No	No	No	No	No	No	No	No	No	No	No	Yes	No	No	No	Yes	No	Yes
Launcher (Rocket)	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	Yes
Launcher (A/B)	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	Yes
Reentry	No	No	No	No	No	No	No	No	No	No	No	No	No	Yes	No	No	No	No	No	No
In-Space	No	No	No	No	No	No	No	No	No	No	No	No	No	Yes	No	No	No	No	No	No
N/A	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	Yes	No	No	No

a) Conceptual design phase applicability. What phases of the aerospace vehicle conceptual design phase is the system applicable to ‘out of the box’?

b) Aerospace vehicle applicability. What aerospace vehicle types/configurations can the system analyze ‘out of the box’?

Table 2-8 shows that of the systems reviewed almost all of them are applicable to the parametric sizing phase of conceptual design, with the exceptions being Model Center (previously discussed) and PrADO. PrADO has not been designed for use in the parametric sizing phase. The disciplinary methods and subsequent input data requirements of PrADO are meant for the late conceptual design steps and preliminary design phases of the product development life cycle. Although all of the by-hand methodologies have broader applicability to steps in conceptual design, the computer-based systems are more narrowly focused. The reason for this can be seen in the increase in input data requirements as you progress from early to late conceptual design. The by-hand methodologies do not specify the input data requirements and interdisciplinary data connections of the system, it is left up to the reader to create those links when putting the system together. The computer-based systems have been tailor-made to answer specific problems, subsequently the data connections have been made with that focus in mind. When attempting to move to the next stage in the conceptual design, the data relationships have to be re-derived and re-implemented.

Table 2-9 shows the applicability of the systems reviewed to different aerospace vehicles. There is a larger concentration of systems focusing on commercial transports, from business jet to larger transports. There is also a concentration of systems which have been created to design military fighters.

2.2.3.4 Influence of New Components or Environment

This section of the review assesses the capability of the user to add new hardware, mission types, and disciplinary analysis methods to the synthesis system without the need to augment the analytic framework or source code, see Figure 2-8.

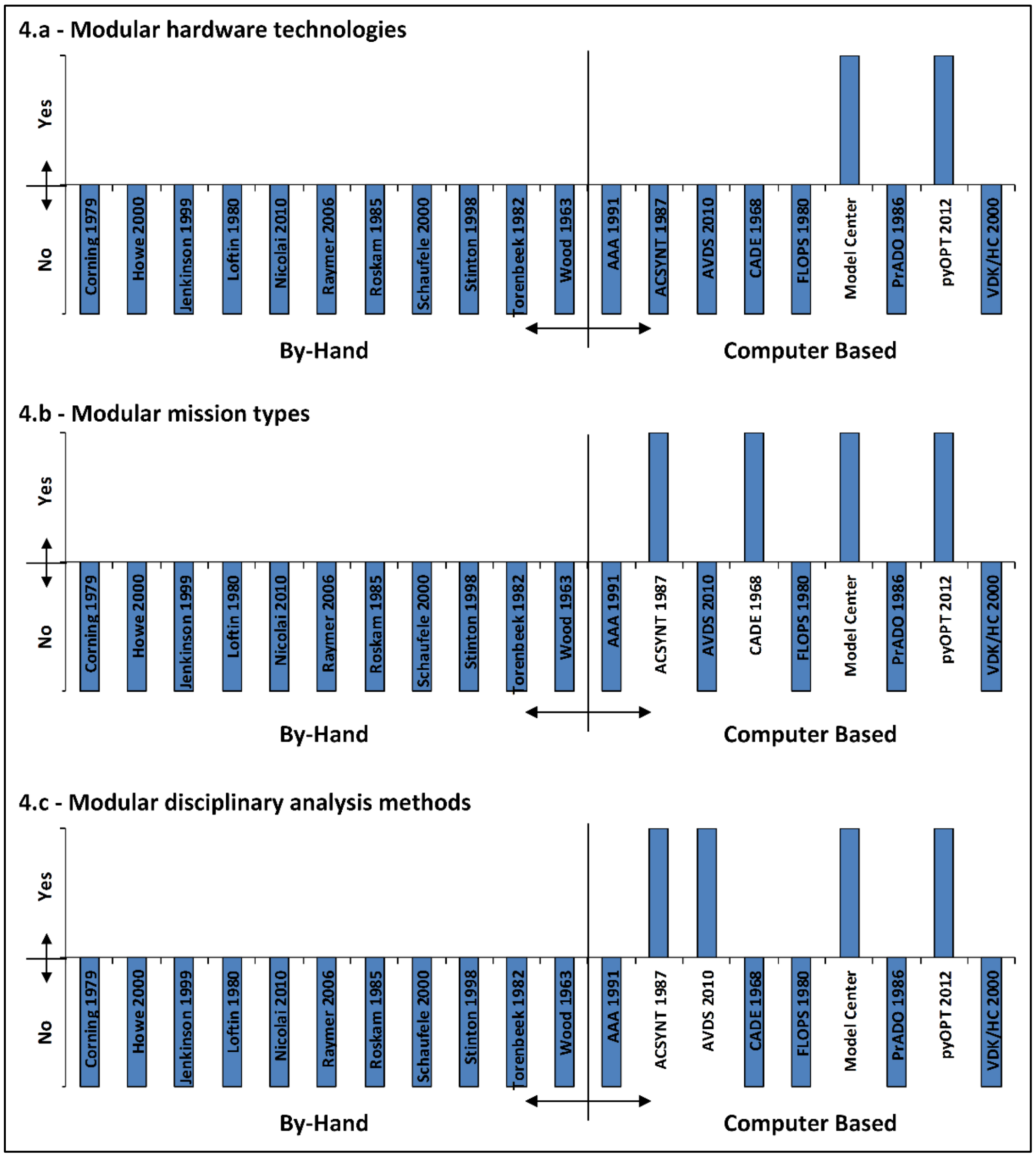


Figure 2-8 System Capability – Influence of New Components or Environment

a) Modular hardware technologies. Can the user add new hardware, at both the vehicle and component level, and integrate that hardware into the analytic framework of the system?

b) Modular mission types. Can the user add new mission types and integrate those mission types into the analytic framework of the system?

c) Modular disciplinary analysis methods. Can the user add new disciplinary methods (empirical, semi-empirical, and analytical) and integrate those disciplinary methods into the analytic framework of the system?

Figure 2-8 shows that two of the surveyed computer-based systems (Model Center, pyOPT) have the capability to add new hardware, mission types, and disciplinary analysis methods. The difficulty with adding new component options to a synthesis system is the need to track the data requirements and relationships in order to properly integrate the new component with the existing system.

Both pyOPT and Model Center are able to add new base components because they have structured data classes for vehicle and subsystem hardware, mission types, and disciplinary methods. pyOPT does this by using object oriented programming and creating objects/classes for each type of information. Model Center similarly uses standard classes of information to categorize system data. One advantage of the Model Center integration platform is the graphical user interface used to create new systems. This provides a setting that allows the user to easily add new hardware, mission types and analytic methods to create a new system.

Model Center is not a synthesis system, it is a platform to integrate new and legacy aerospace codes in order to create synthesis systems. Therefore, Model Center has the capability to provide modular connections for each piece of a system, but the knowledge of what pieces to use and what order to connect them is left to the user. ACSYNT is an early attempt to create a system in this setting, consisting of disciplinary analysis modules created at NASA Langley integrated through the early Model Center framework.

AVDS has been created with the intent of integrating a stand-alone methods library into a synthesis system. The difference between this system and that of pyOPT and Model Center is the analytic process which AVDS adheres to; AVDS follows the process found in the VDK/HC system. This means that the number of disciplinary analysis modules, the order these modules run in and the variables used to mathematically converge the system are constant. This results in the onus being placed on the user to track the data relationships for the given system, and set disciplinary method variable input and output locations manually.

2.2.3.5 Prioritization of Technology Development Efforts

This section of the review assesses the capability of each synthesis system to match the fidelity of disciplinary analysis methods to the given problem requirements, see Figure 2-9.

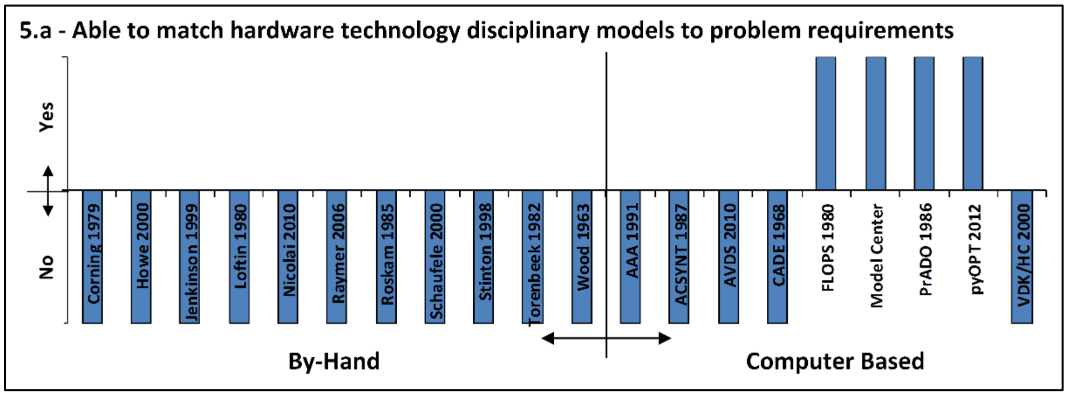


Figure 2-9 System Capability – Prioritization of technology development efforts

a) Able to match hardware technology disciplinary models to problem requirements. Is the process of disciplinary method selection in terms of fidelity effected by the input problem requirements?

b) Data management capability. How is data managed/transferred within the synthesis system?

Figure 2-9 shows that four of the synthesis systems reviewed allow the user to adjust the level of disciplinary fidelity based on the given problem, namely FLOPS, Model Center, PrADO and pyOPT. FLOPS and PrADO have several options for disciplinary analysis methods of varying fidelity. In both cases, the selection of method is made through the selection/omission of options in the system input file. Model Center and pyOPT allow for different level in fidelity using a different mechanism. The modular disciplinary methods, see previous section, allow for selection or use of methods of varying fidelity level. The main difference between this (Model Center, pyOPT) and the former (FLOPS, PrADO) in the case of Model Center and pyOPT only the methods selected appear in the system. In the case of PrADO and FLOPS all of the methods that exist in the system are included and available every time the system is run, the choices made in the input file tell the system what 'route' to take through the code. In this case additional methods fidelity options result in an increase in the size of the system (e.g. lines of source code, data connections, etc.), because each method must be integrated into the system a priori. In the case of Model Center and pyOPT only the methods that are selected for the given problem appear in the system, thus reducing the size of the synthesis system, and decoupling the size of the synthesis system from the number of analytis methods stored.

The need for a database management system (DBMS) in aerospace synthesis has been stated by each of the previous studies. Chudoba alludes to the importance of the DBMS through the mentioning of requirements directly resulting from an integrated DBMS. Coleman describes the DBMS found in the PrADO synthesis system. Highlighting specifically the ability of the PrADO system to analyze each discipline in a modular fashion and integrate these results through the use of the DBMS as the backbone of the analytical framework. Huang (Huang 2006) goes one step further and state the following:

> *The desired data management system not only stores and manipulates numerical data belonging to physical design parameters, but it also controls the utilization of the design methods library. Additionally, it is a communication platform for the inter-discipline modules. The availability of a robust DMS facilitates data transfer, reduces data transcription errors, and allows the designer to use different computing environments and widely distributed teams.*

The resulting listing represents a database management approach meant for integrated data storage, transfer and management, see Table 2-11.

Table 2-10 System Capability – Data Management Survey Criterion

	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)

After reviewing each synthesis system in terms of their System Capability metrics, it can be seen that one of the main difference between the by-hand and computer-based systems is the management of data. The by-hand methodologies layout a framework for an analytic process, but the actual connection of data from discipline to discipline, and discipline to system is left to the synthesis specialist. Due to the nature of computer-based systems, the analytic framework, as well as the data connections have been decided a priori. Each computer-based system is the result of this implementation for a specific problem. The review of database management metrics has been focused only on computer-based systems. Table 2-11 show the results of the review.

Table 2-11 System Capability – Database Management

	AAA 1991	ACSYNT 1987	AVDS 2010	CADE 1968	FLOPS 1980	Model Center	PrADO 1986	pyOPT 2012	VDK/HC 2000
a	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
b	No	No	No	No	No	Yes	No	Yes	No
c	No	No	No	No	No	Yes	No	Yes	No
d	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
e	No	Yes	No	No	No	Yes	No	Yes	No
f	No	Yes	No	No	No	Yes	No	Yes	No
g	No	No	No	No	No	Yes	No	No	No
h	No	No	No	No	No	Yes	No	Yes	No
i	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes
j	No	No	No	No	No	Yes	No	No	No
k	No	No	No	No	No	Yes	No	No	No
l	No	No	No	No	No	Yes	No	No	No
m	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
n	No	No	No	No	No	No	No	No	No
	4 of 14	**6 of 14**	**3 of 14**	**4 of 14**	**3 of 14**	**13 of 14**	**4 of 14**	**9 of 14**	**4 of 14**

There is a clear distinction between classical aircraft design codes and modern implementations. Model Center and pyOPT show the highest degree of data management capability with 13 and 9 of 14 possible criteria. This result can be expected as both of these systems were shown to have the most modularity in terms of adding/assessing aerospace products, mission, and processes (Figure 2-8). The data management review provides an explanation as to how Model Center and pyOPT are able to provide that level of modularity. The database management system in each case has been designed not to connect pieces to solve a specific problem, but instead to connect pieces to solve a user-defined problem. This mindset and subsequent software implementation creates capability not present in classical aircraft synthesis systems.

2.2.3.6 Problem Input Characterization

This section of the review characterizes the input problem requirements for each synthesis system, see Figure 2-10.

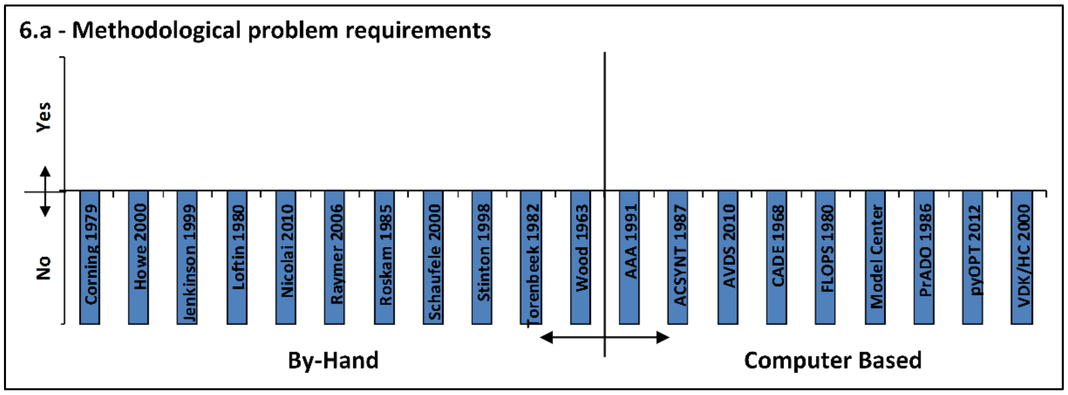


Figure 2-10 System Capability – Problem input characterization

a) Methodological problem requirements. Are the problem input requirements the output of an analytical methodology (Yes) or static inputs to the system (No)?

For the most part, the previous systems have started with the assumption that the problem is a given. Coleman also assumes a given problem but does provide Process and Method overview cards; for a given selection of process and methods, one can refer to the overview card and find the inputs needed to run the system, as well as those needed for each individual method. This helps to define the systematic requirements to analyze a given problem, and allows the user to indirectly assess how these choices affect their system. The aspect that is not involved in these specifications is the methodology to define the problem being solved from the outset. To this end, the input classification criterion has been added with the goal of defining whether a systems input are static, or if they have been developed methodologically.

Figure 2-10 shows that all of the systems reviewed have static inputs. This means that the problem definition must be done outside of the purview of the synthesis system. This disintegrated approach to the problem definition and subsequent analysis is commonly found in early conceptual design for aerospace vehicles. One outcome of this is the lack of a feedback loop between the problem definition and the problem solution.

This eliminates the ability of the decision maker to assess if a problem should be solved, or if the problem definition in and of itself has been ill-formed.

2.3 Solution Concept Specification

The original goal of the research project as described in the previous chapter, was the creation of a system for engine selection in early conceptual design. A review of propulsion system analysis and integration has led to the realization that the most prudent method to achieve this task is the improvement in the efficacy of the synthesis process. A subsequent review of aerospace synthesis systems has highlighted the effect of System Capability on the system's ability to solve a variety of problems in aerospace conceptual design. One of the major takeaways has been that the systems able to model the widest variety of problems have a database management system that is able to adapt its structure for a given problem, Model Center as the prime example. The open and adaptable nature of integration frameworks like Model Center while allowing for easy connection between new and legacy tools, do not have any structure or format for analysis in and of themselves. They are created with the requirement that a synthesis specialist knows from the outset what he wants to model, how he wants to model it, and how everything should be connected. This means that while data connections can be easily made between analysis modules, the question of which modules to choose for a given problem is still solely a function of user experience.

Two different trains of thought have been found to 'book end' the problem of conceptual design synthesis, the classical structured synthesis system and the open ended integration platform. It is the intent of this research to bridge the gap between these types of systems, creating 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. This leads to the creation of a system with the following 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

To meet these requirements, the research will focus on the creation of an environment with the purpose of tailor making synthesis systems for aerospace vehicle conceptual design. The deliverable of the setting is not the design of a vehicle or solution to an aerospace problem, the output is the integrated system designed to solve a given aerospace problem.

Chapter 3

Solution Concept

Aerospace Synthesis Systems are specialized aircraft design tools for evaluating total vehicle performance resulting from multidisciplinary effects. These systems are combinations of analysis methods for multiple disciplines, run in specific orders to solve for specific metrics. The research objectives define the need for a framework with the capability to tailor make synthesis systems based on user need. This deliverable of this framework is a stand-alone integrated analysis setting made to solve a user specified problem. To this end a framework has been created which is analogous to an automated assembly line; where an assembly line is comprised of categorized and ordered storage compartments, machinery meant to pick and assemble parts, and an interface between them that controls the building process based on user input. Applying this setting to the composition of aerospace synthesis systems leads to the requirement to have a repository of synthesis system parts, a mechanism to put those parts together, and an interface to control the generation procedure through user input.

The solution concept is comprised of two tasks: the process of breaking down aerospace synthesis systems into their constituent parts, and the process of combining those parts to create new synthesis systems. Taking principals from both Systems Engineering (SE) and Modelling and Simulation (M&S), and applying them to the problem of aerospace synthesis allows for a unique building block approach to synthesis system composition.

The chapter is divided into three sections:

- Description of non-aerospace techniques for system composition / decomposition
- Derivation of aerospace synthesis system building blocks
- Derivation of aerospace synthesis system generation methodology

3.1 System Composition and Decomposition Techniques

The concept of system decomposition and synthesis is an inherently interdisciplinary problem with many techniques originating from outside the realm of aerospace engineering. Two specific fields of study focusing on this task are Systems Engineering, and Modelling and Simulation. The following sections serve to define an aerospace synthesis system in terms of these non-aerospace fields, as well as to present how these other disciplines have addressed this type of problem.

3.1.1 Complex Multidisciplinary Systems

The term 'system' in Systems Engineering has been shown to have multiple meanings. Kline (Kline 1995) gives three definitions of a 'system':

- The object of study, what we want to discuss, define, think about, write about, and so forth.
- A picture, equation, mental image, conceptual model, word description, etc., which represents the entity we want to discuss, analyze, think about, write about.
- An integrated entity of heterogeneous parts which acts in a coordinated way.

The third definition gives the idea that a system is a composition of unique parts acting in an ordered manner; this shows similarity to that of an aerospace synthesis system. Additionally, this viewpoint allows the introduction of the concept of system complexity.

Ryan (Ryan 2007) defines complexity as "the amount of information needed to describe a process, a system, or an object". Bar-Yam (Bar-Yam 1997) characterizes complexity by the system elements, their number, the interactions, their strength, formation/operation and their time scales, diversity/variability, environment and its demands, activities and their objectives. This shows that system complexity changes with types of disciplines considered in the representation of the system. It also speaks to the multidisciplinary nature of complex systems. A simple system can be made complex if it is to be studied from multiple disciplinary points of view. Therefore, if a problem necessitates the accounting of multiple disciplinary effects, a complex system model is required to solve it. Ryan (Ryan 2007) proves that complex systems retain definitions across philosophy, theory and application. Ryan also shows that Complex Systems they can be used to answer a broad range of problems if a multidisciplinary approach is used to their design.

In the context of this research, the term complex multidisciplinary system (CMDS) is used. The goal of this research to build a methodology for generating aerospace synthesis systems; these systems are complex because of their numerous highly integrated analytic methodologies. The word 'multidisciplinary' has been added to Complex Systems in order to emphasize that they need to be studied from more than one disciplinary perspective.

3.1.2 Systems Engineering Process

Systems Engineering (SE) is defined in the MIL-STD-499A (United States. 1974):

> *"The application of scientific and engineering efforts to (a) transform an operational need into a description of system performance parameters and a system configuration through the use of an iterative process of definition, synthesis, analysis, design, test, and evaluation; (b) integrate related technical parameters and ensure compatibility of all physical, functional, and program interfaces in a manner that optimizes the total system definition and design; (c) integrate reliability, maintainability, safety, survivability, human engineering, and other such factors into the total engineering effort to meet cost, schedule, supportability, and technical performance objectives"*

The Systems Engineering Process (SEP) is the generic process applied to systematically achieve the Systems Engineering requirements specified in the MIL-STD-499A. The IEEE further defines the SEP as "*a generic problem-solving process that provides the mechanisms for identifying and evolving the product and process definitions of a system*". Figure 3-1 is a graphical representation showing both the SEP task sequence as well as the iterative nature of the SEP procedure. The purpose of the SEP is to take user requirements and iteratively create the technical and managerial processes required to realize a system that meets those requirements.

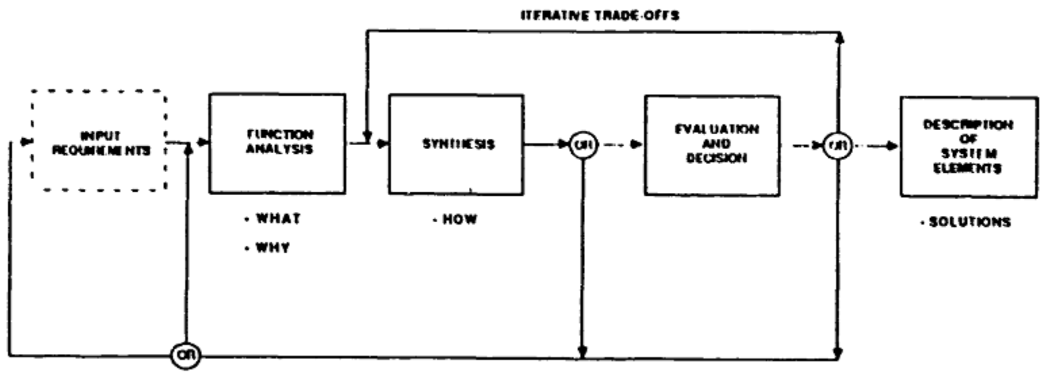


Figure 3-1 The Systems Engineering Process (Kockler et al. 1990)

Ryan (Ryan 2007) summarizes the SEP stages as follows:

1. [Input Requirements] - Customer needs are captured in precise, quantified requirements specifications.
2. [Functional Analysis] - System requirements are decomposed into requirements for subsystems, until each subsystem requirement in sufficiently simple.
3. [Synthesis] - Design synthesis integrates subsystems.
4. [Evaluation and Decision] - Test and evaluation identifies unintended interactions between subsystems, which may generate some additional requirements for subsystems. If there are unintended consequences (i.e. unplanned emergent properties), the process returns to stage 2, and repeats until the system meets the requirements.

The SEP has historically been applied to aerospace vehicle product development (the ‘system’ being defined is the vehicle); Bell Labs and the Western Electric Company applied SE techniques during the development of the NIKE air defense system, and NASA applied the SEP during the design, planning and manufacturing of Project Apollo. In this context the SEP defines the system in terms of its requirements throughout the product life cycle, see Figure 3-2. This is done through the iterative assessing input requirements and defining system components meant to meet those requirements; meaning the subcomponents of the top level system can also be considered systems and decomposed into their constituent parts. The current research objective is focused on the generation of an analysis framework for modelling a given aerospace vehicle. This objective can be seen as a subfunction of the overall vehicle 'system', where the input requirements are assumed to have been defined through an earlier application of the SEP.

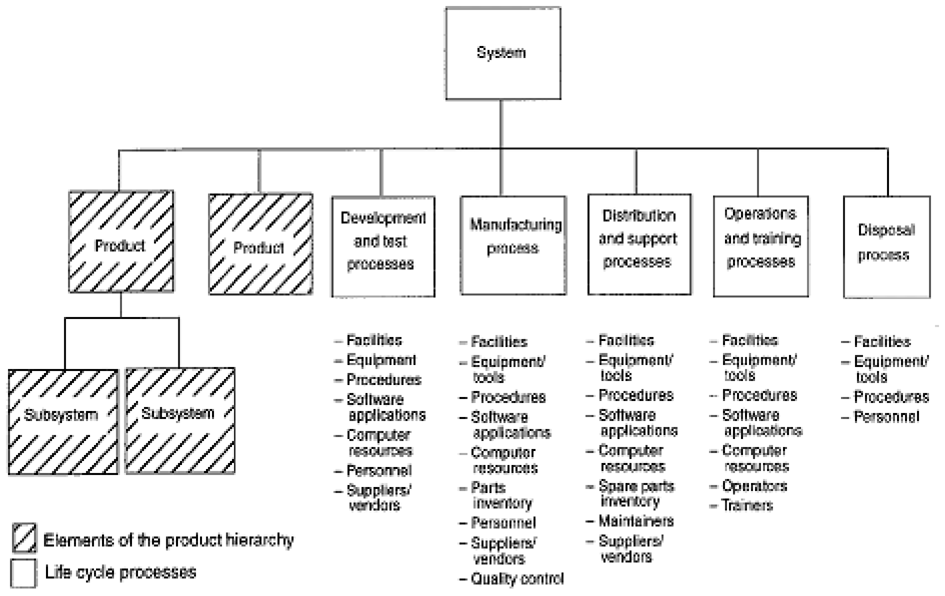


Figure 3-2 Life cycle process definition (IEEE 2007)

3.1.2.1 Functional Analysis

The purpose of the functional analysis stage of the SEP is to assess the input requirements of the system and define subsystems to meet those requirements. This process is highly iterative as each output subsystem can be further broken down into its parts, see Figure 3-3. IEEE (IEEE 2007) defines functional analysis as the process used to meet two objectives: a) To describe the problem defined by requirements analysis in clearer detail, b) To decompose the system functions to lower-level functions that should be satisfied by elements of the system design (e.g., subsystems, components, or parts).

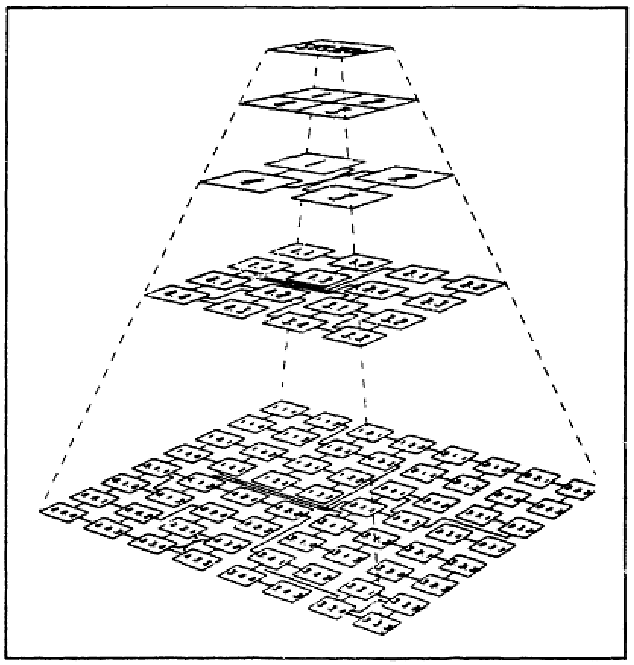


Figure 3-3 Top-Down Approach to Functional Decomposition (Kockler et al. 1990)

The NASA Systems Engineering Handbook (NASA 2007) describes functional analysis techniques used to perform these tasks: Product Breakdown Structure (PBS), Functional Flow Block Diagram (FFBD), and N^2 Diagrams (see Chapter 1).

The PBS is a hierarchical breakdown of the product including hardware, software, and information items (documents, databases, etc.). The goal of the PBS is to create a top-down hierarchical relationship "*carried down to the lowest level for which there is a cognizant engineer or manager*". Figure 3-4 shows an example PBS for a an in-space system. This example shows three distinct product subsystems meant to function during a given flight segment. Also worth noting is the connection between the Spacecraft and Payload interfaces; this provides a visual cue for inter-hardware dependencies.

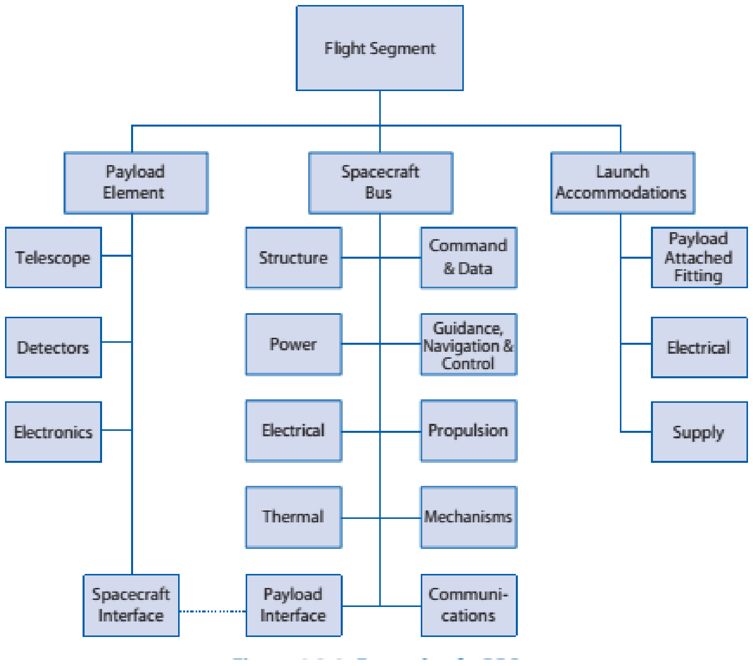


Figure 3-4 Example Product Breakdown Structure (NASA 2007)

The FFBD is the most often described technique used for functional analysis. The purpose of the FFBD is to provide sequential relationships for all functions required by the system to accomplish given requirements. In a more general since, the FFBD tries to answer "what" must happen without defining "how" it is to occur. As with other steps in the SEP, the task of completing an FFBD for a given system is highly iterative. The top-level functional blocks are first defined, then for each of those blocks a new sequence of functions are defined in order to answer "what" must happen to achieve that higher level function. This process continues until each block has been sufficiently described. Figure 3-5 shows an example FFBD for the flight mission of a spacecraft. The first level shows the total mission trajectory sequence. The 2nd level expands on the subfunction required to achieve the "Perform Mission Operations" function of the first level, the 3rd level expands on the subfunction required to achieve the "Acquire Payload Data" function of the 2nd level.

Each level not only shows the sequence in which function must occur, but also describes Boolean relationships between functions at a given level.

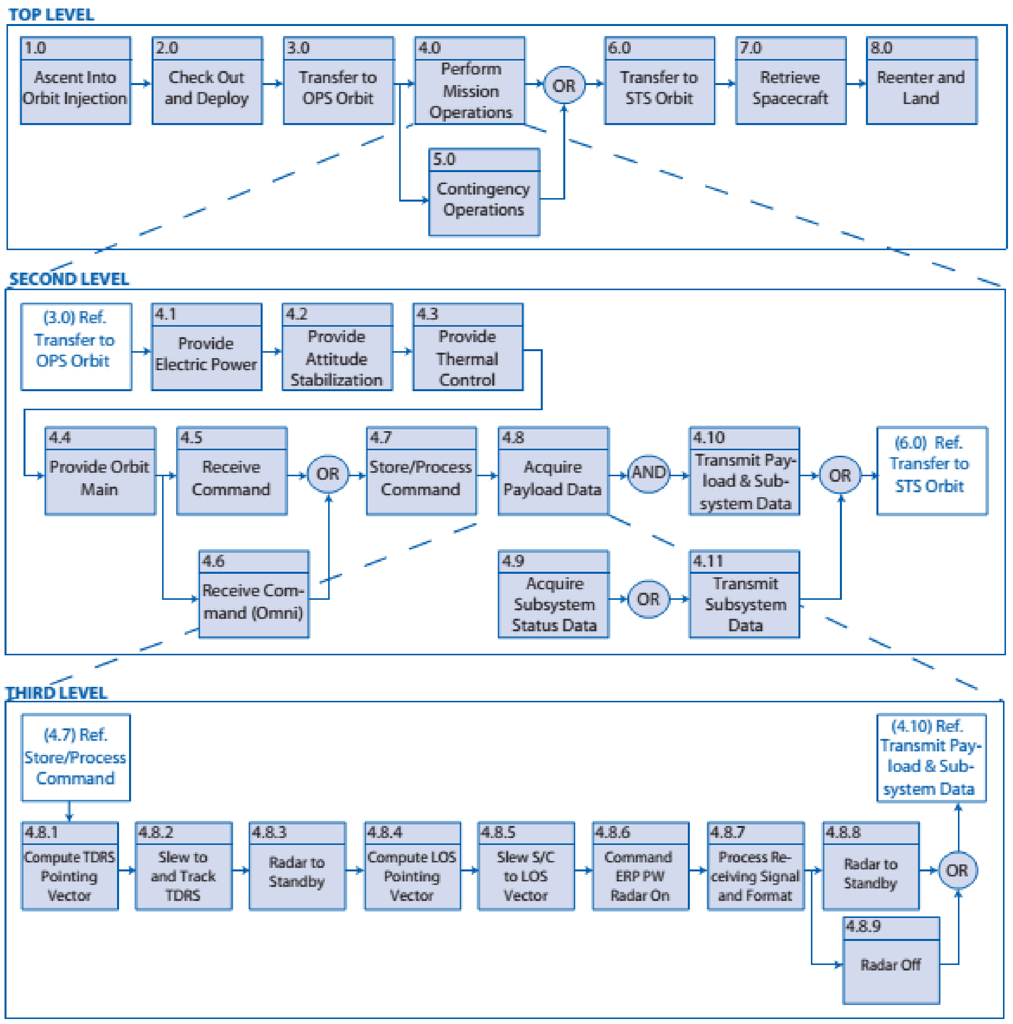

TOP LEVEL
1.0 Ascent Into Orbit Injection
2.0 Check Out and Deploy
3.0 Transfer to OPS Orbit
4.0 Perform Mission Operations
OR
6.0 Transfer to STS Orbit
7.0 Retrieve Spacecraft
8.0 Reenter and Land
5.0 Contingency Operations
SECOND LEVEL
(3.0) Ref. Transfer to OPS Orbit
4.1 Provide Electric Power
4.2 Provide Attitude Stabilization
4.3 Provide Thermal Control
4.4 Provide Orbit Main
4.5 Receive Command
OR
4.7 Store/Process Command
4.8 Acquire Payload Data
AND
4.10 Transmit Payload & Subsystem Data
OR
(6.0) Ref. Transfer to STS Orbit
4.6 Receive Command (Omni)
4.9 Acquire Subsystem Status Data
OR
4.11 Transmit Subsystem Data
THIRD LEVEL
(4.7) Ref. Store/Process Command
(4.10) Ref. Transmit Payload & Subsystem Data
4.8.1 Compute TDRS Pointing Vector
4.8.2 Slew to and Track TDRS
4.8.3 Radar to Standby
4.8.4 Compute LOS Pointing Vector
4.8.5 Slew S/C to LOS Vector
4.8.6 Command ERP PW Radar On
4.8.7 Process Receiving Signal and Format
4.8.8 Radar to Standby
OR
4.8.9 Radar Off

Figure 3-5 Example Function Flow Block Diagram (NASA 2007)

The functional analysis stage of the SEP provides guidance and techniques to logically decompose a system into its constituent parts. This breakdown centers on the ability to take a given set of input requirements and define the hardware and function needed to fulfill those requirements. Functional analysis answers the question of “what” is needed to meet requirements, it does not attempt to answer the question of “how”. Moving on with the assumption that our application of the SEP has provided a breakdown and definition of the components of a CMDS, a methodology to compose those components into a custom-tailored CMDS is required. This methodology is meant to answer the “how”, left open by the functional analysis.

3.1.3 Simulation Composability

Simulation Composability is a Modelling and Simulations (M&S) concept describing the “capability to select and assemble simulation components in various combinations into valid simulation systems to satisfy specific user requirements” (Petty and Weisel 2003). A notional representation of composability is presented in Figure 3-6. The power of this type of system comes into the ability to re-use components previously built for other applications. The components are stored in a repository, where the choice of components and the order which they run are based on user need.

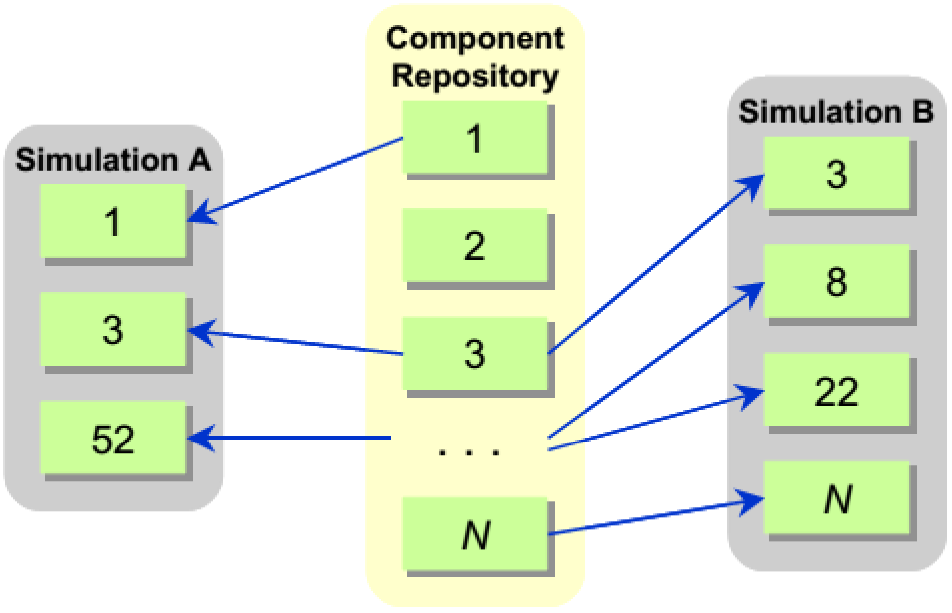


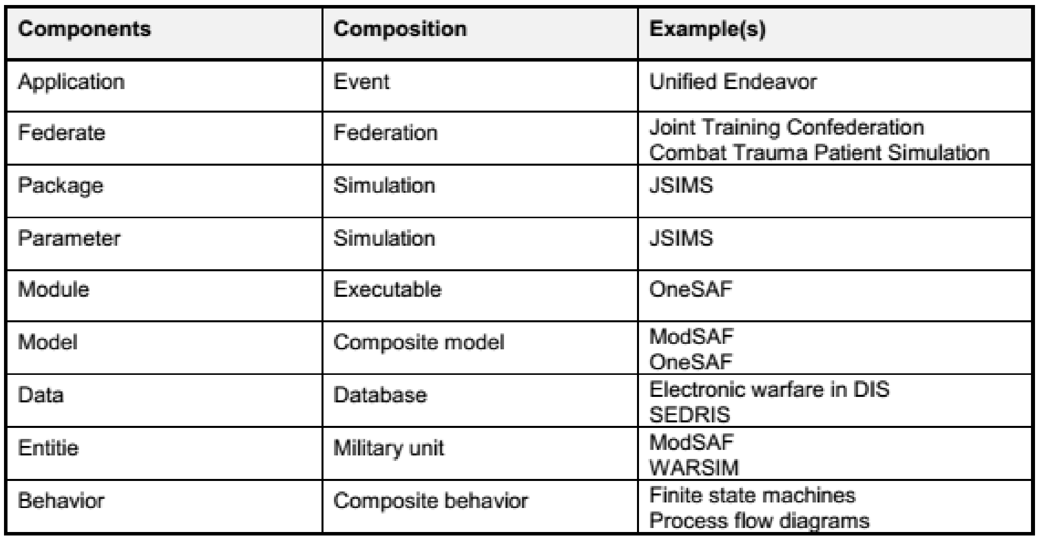
Figure 3-6 Notional Example of Composability (Petty and Weisel 2003)

The main benefit of system composability come in the form of time savings towards the development of new models. Shaw (Shaw 1995) asserts that "*most applications devote less than 10% of their code to the overt function of the system; the other 90% goes into system or administrative code*". Simulation Composability is the M&S response to this problem, where a reduction in the time to create and integrate new simulations is a direct result.

3.1.3.1 Level and Type of Simulation Composability

There is not a common definition or application of the term composability in M&S literature. To create a basis for description Petty links the level of composability through the definition of the individual components, as well as the composition of those components., see Table 3-1. A full description of each level can be found in (Weisel 2004).

Table 3-1 Levels of Composability (Petty and Weisel 2003)



Components	Composition	Example(s)
Application	Event	Unified Endeavor
Federate	Federation	Joint Training Confederation Combat Trauma Patient Simulation
Package	Simulation	JSIMS
Parameter	Simulation	JSIMS
Module	Executable	OneSAF
Model	Composite model	ModSAF OneSAF
Data	Database	Electronic warfare in DIS SEDRIS
Entitie	Military unit	ModSAF WARSIM
Behavior	Composite behavior	Finite state machines Process flow diagrams

The Model/Composite Model level has been defined as "*Separate models of smaller-scale processes or objects are composed into composite models of larger-scale processes or objects. For example, models of platform/entity sub-systems, such as* *sensors and weapons, may be composed into composite models of platforms/entities, such as aircraft* ". This level is most appropriate to the current research task of synthesis system generation, where these systems are combinations of individual disciplinary models.

In addition to the levels of composability based on application, a composable system can be described by the type of composability it is implementing. Weisel (Weisel 2004) describes two types of composability:

- Syntactic Composability - Requires that the composable components be constructed so that their implementation details, such as parameter passing mechanisms, external data accesses, and timing assumptions are compatible for all of the different configurations that might be composed. The question in engineering (syntactic) composability is whether the components can be connected
- Semantic Composability - Addresses whether the models that make up the composed simulation system can be meaningfully composed, i.e., if their combined computation is semantically valid

These definitions serve to make distinction when assessing whether a system composable from the viewpoint of the system generator, as opposed to the system user. In the first case, Syntactic Composability, measures the ability of the system to create data connections between individual components regardless of the order in which they are assembled. From this perspective, the goal would be to assess whether the system can 'run' given any combination of components. The second case, Semantic Composability, measures the capability of the given system to answer the prescribed problem. The goal in this sense to answer the question of whether a can assess a given problem; are the components applicable, do the results 'make sense'.

3.1.3.2 Syntactically Composable Systems

The concept of syntactic composability has been implemented in the construction if several real-world systems, see Table 3-2, whereas a fully semantically composable system has yet to be designed and implemented.

Table 3-2 Example Syntactically Composable Systems

Acronym	Full Name	Reference
JMASS	Joint Modeling and Simulation System	(Weisel 2004; OFFICE OF THE UNDER SECRETARY OF DEFENSE 1997)
OneSAF	ONE SemiAutomated Forces Objective System	(Wittman, Robert, and Harrison 2001)
BOM	Base Object Model	(SISO 2004)
CODES	COmposable Discrete-Event scalable Simulation	(Szabo and Teo 2007)

JMASS is an M&S architecture designed for simulation component creation, combination, and reuse. JMASS uses a Common library approach to syntactic composability. This means that in order for a model to exist in the JMASS system, they must be created in a JMASS environment and use JMASS Application Programming Interfaces (APIs) (Weisel 2004). An interesting aspect of the JMASS system is the mechanism used to connect models. Each model contains information concerning its interfaces. When a new simulation is created this data is referenced and new code is generated to handle the interfaces. This provides automatic enforcement of model interfaces, and ensures syntactic composability in terms of data connections between selected models.

OneSAF is an entity based simulation system with the capability to create simulations overs a range of domains: Advanced Concept Requirements, Training, Exercise and Military Operations, and Research Development and Acquisition. OneSAF employs the Product Line Architecture Framework (PLAF) to "*guide the definition of individual components, their services, and interfaces so that they can be independently developed and then combined to support a variety of products and system configurations*" (Wittman, Robert, and Harrison 2001). PLAF uses a hierarchical composition process to create user defined systems, where systems are composed of products and products are composed of components. Using the PLAF as the standard for components definition, OneSAF supports an array of nine products types for system generation, see Figure 3-7.

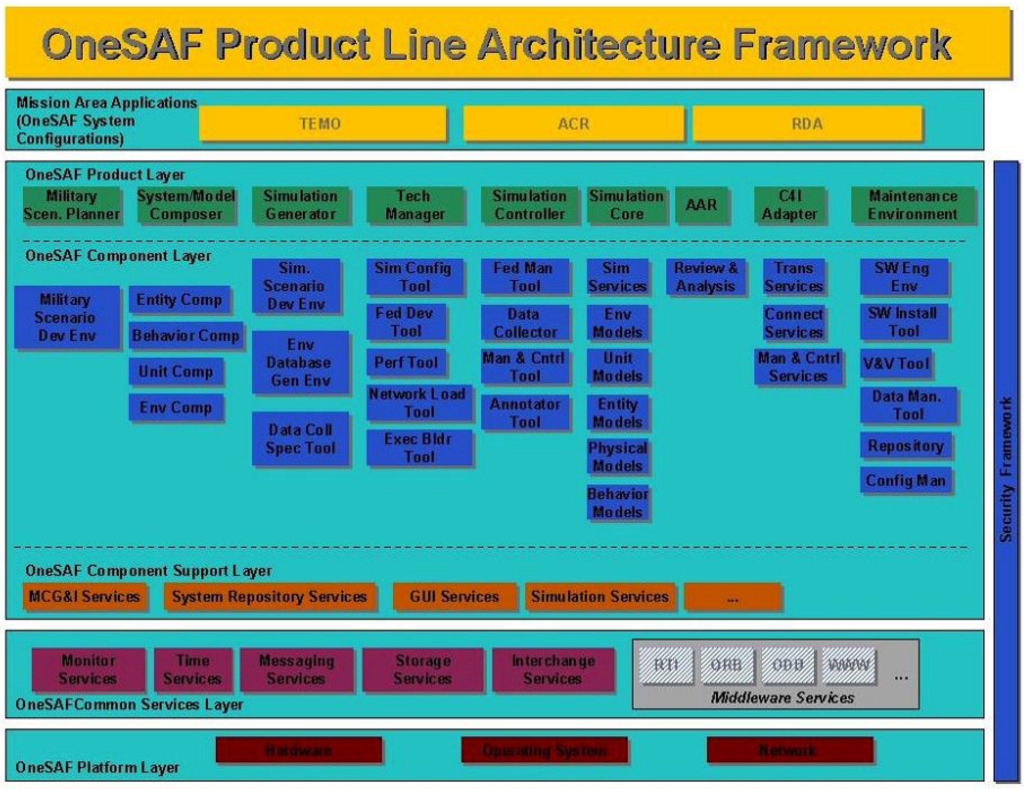


Figure 3-7 OneSAF PLAF (Wittman, Robert, and Harrison 2001)

BOM is an object model implementation that aims to "*provide a key mechanism in facilitating interoperability, reuse, and composability*." (SISO 2004) A key to BOM is the idea that a simulation can be broken into parts, and those parts can be "*extracted and* *reused as modeling building-blocks or components*." The mechanism for this is the definition of the Base Object Models (BOMs) follows a standard template identified in the IEEE 1516.3 HLA Federation Development and Execution Process (FEDEP). There are two types of BOMs specified, see Figure 3-8:

- Interface BOMs - Contains the essential elements needed to represent a reusable pattern of interplay, which is characterized by messages and/or triggers related to one or more object classes
- Encapsulated BOMs - Represents a manifestation of an Interface BOM. It is a manifestation, because it details how the BOM can carry out the pattern elements defined by the Interface BOM. This includes behavioral information for modeling what was identified by the Interface BOM, and additional meta-data to better support composability such as validation, level-of-fidelity, and graphical meta-data used for visual rendering.

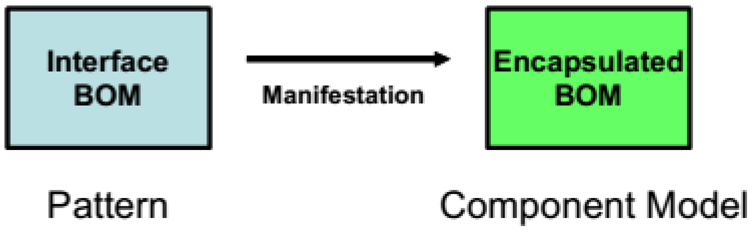


Figure 3-8 BOM Composability View (SISO 2004)

Compositions of BOMs are referred to as a Mega-BOM, see Figure 3-9. A Mega-BOM contains all of the metadata from the individual BOMs, additionally it contains the dependency and relational data defining the composition of the BOMs. In this same manner Mega-BOMs can be composed to create more complex simulations.

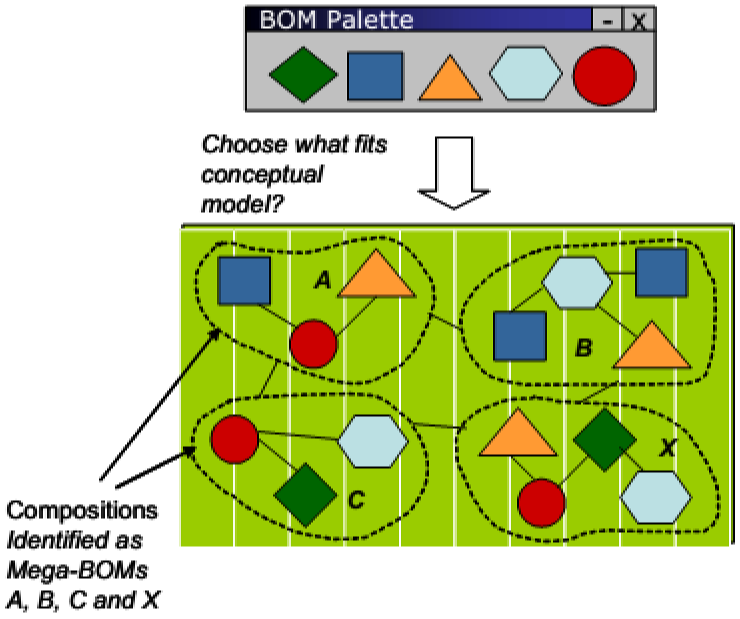

BOM Palette
Choose what fits conceptual model?
A
B
C
X
Compositions Identified as Mega-BOMs A, B, C and X

Figure 3-9 Creating BOM Compositions (SISO 2004)

CODES is a "*hierarchical component framework to support component-based modelling and simulation*" (Szabo and Teo 2007). The CODES framework is centered around a four step process toward building component-based simulations: component discovery, model validation, model execution and model deployment. The CODES framework has six modules, see Figure 3-10, which support these steps:

- Model Composer - Responsible for component discovery and for model validation
- Model Repository - A database for models or model components from which one may compose other models
- Locator –Searches the Model Repository given criteria from Model Composer
- Validator - Checks model against input syntactic composability criterion
- Actuator- Executes validated model
- Distributor - places the validated simulators in the model repository according to the deployment scheme to facilitate model discovery

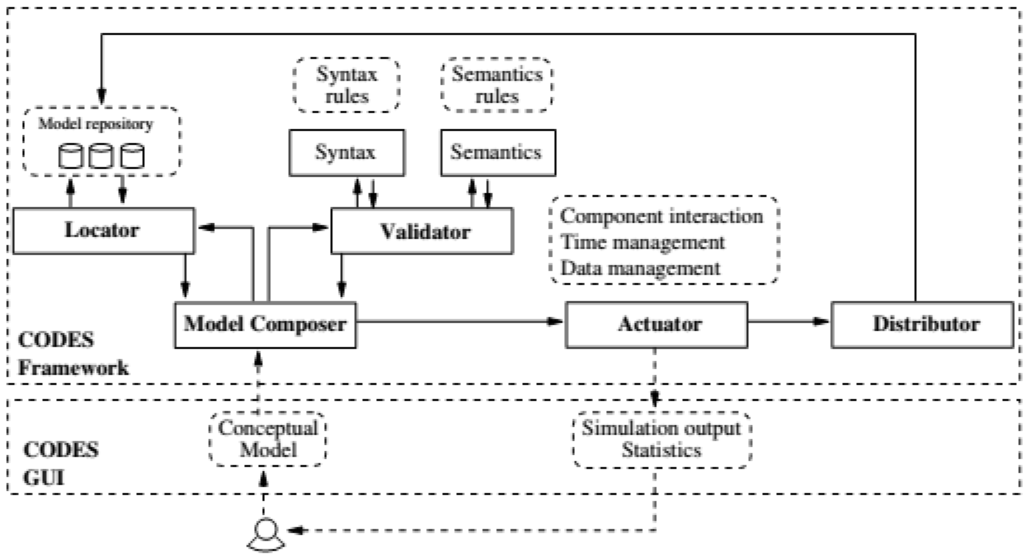


Figure 3-10 Component-based Model Simulation Development (Szabo and Teo 2007)

Syntactic composability rules are input using the Extended Backus–Naur Form (EBNF) grammar notation. The use of EBNF based grammars to specify model composition rules supports syntactic composability verification as well as aids in the discovery of shared models and model components (Szabo and Teo 2007).

3.1.4 Conclusions

In order to facilitate in the creation of a methodology for generating aerospace synthesis systems several non-aerospace concepts have been reviewed. The implementation of interdisciplinary methodologies allows for a new and unique solution concept for an aerospace problem.

The SEP provides systematic guidelines for the decomposition of a system into its constituent functional components. Due to its generic nature, the SEP can be applied to problems in various fields of study. The SEP is meant to be applied throughout the product development life cycle. Of this total a focus has been placed on the steps in the SEP that were found to benefit the aerospace synthesis system ‘decomposition’ activity. To this end the ‘Functional Analysis’ procedure has shown merit, and has been applied to create CMDS Building blocks.

Once those Building blocks are assumed to have been created, it became necessary to define a methodology for the automated composition of those building blocks to create a user defined CMDS. Simulation Composability is a field specializing in this task. A review of syntactically composable systems has highlighted several mechanisms to ensure composability and the ways in which they differ. A combination of these characteristics has been applied to create a syntactically composable framework for the automatic generation of a user defined CMDS.

The following sections define the application of these techniques towards the problem of aerospace synthesis system decomposition and generation.

3.2 Aerospace Synthesis System Decomposition – CMDS Building Blocks

The process of creating CMDS building blocks, see Figure 3-11, results from combining the SEP Functional Analysis procedure with knowledge gained from a review of current and past aerospace synthesis systems.

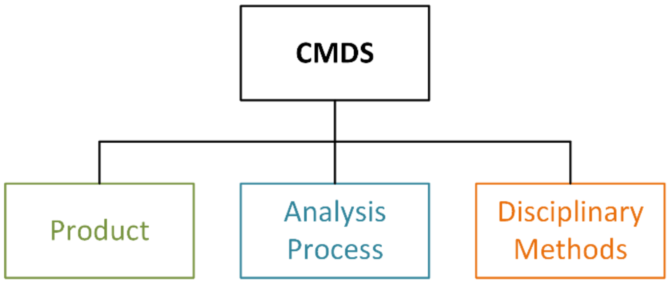


Figure 3-11 CMDS Top-Level Decomposition Blocks

Coleman (Coleman 2010) shows that aerospace synthesis systems are comprised of disciplinary methods as well as an analytic process. The disciplinary methods serve as the analysis modules of the system, whereas the analytic process serves as the system blueprint and controls the order and integration of the analysis modules. Coleman further explains that the choice of disciplinary methods is a function of the aircraft configuration, design mission and operational constraints defined for the problem. Accordingly, a CMDS is comprised of three classes of information: a description of the product being modelled, a definition of the analytic process being used to order and integrate the model, and a permutation of disciplinary methods performing the analysis of the model.

3.2.1 Product Blocks

The Product refers to physical representation of what is being modelled/solved for; this is defined here within as a combination of functional subsystems, operational events and operational requirement, see Figure 3-12.

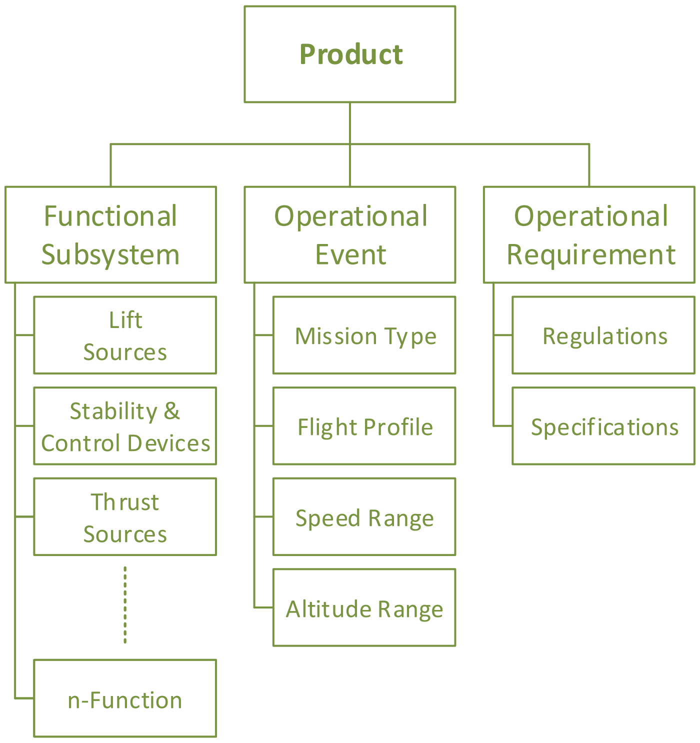


Figure 3-12 Product Block Decomposition

- Functional Subsystem - Individual hardware components added in order to achieve one or more primary functions
- Operational Event - Operational attribute that is time dependent
- Operational Requirement - Operational attribute that is time independent

3.2.1.1 Functional Subsystem

It is common is classical aerospace synthesis systems to define vehicle hardware through the selection of disciplinary methods. An example of this can be seen in the PrADO (Heinze 1994), AVDS (Coleman 2010) and FLOPS (McCullers 1987) synthesis systems. In each case, the selection of methods in the input file defines the vehicle hardware being modelled. The sequence of tasks to define vehicle hardware in this manner is:

1. Create list of hardware for vehicle to be examined; separate from synthesis system
2. Look through synthesis system input file
3. Select disciplinary methods in input file that match the hardware list
4. Re-examine hardware listing and selected methods to ensure all required hardware are being modelled

This puts the onus entirely on the synthesis specialist to both keep a listing of hardware inputs (separate from synthesis system), and to select disciplinary methods that represent that hardware (in synthesis system). An outcome of this setting is the coupling of the definition of the vehicle hardware meant to be modelled with disciplinary analysis meant to model it; the vehicle hardware is defined by the disciplinary methods selected.

In order to decouple the definition of the vehicle hardware from the analysis model, a hardware build-up methodology has been derived. Each hardware component is first defined by the function(s) that component provides to the vehicle (Figure 3-13).

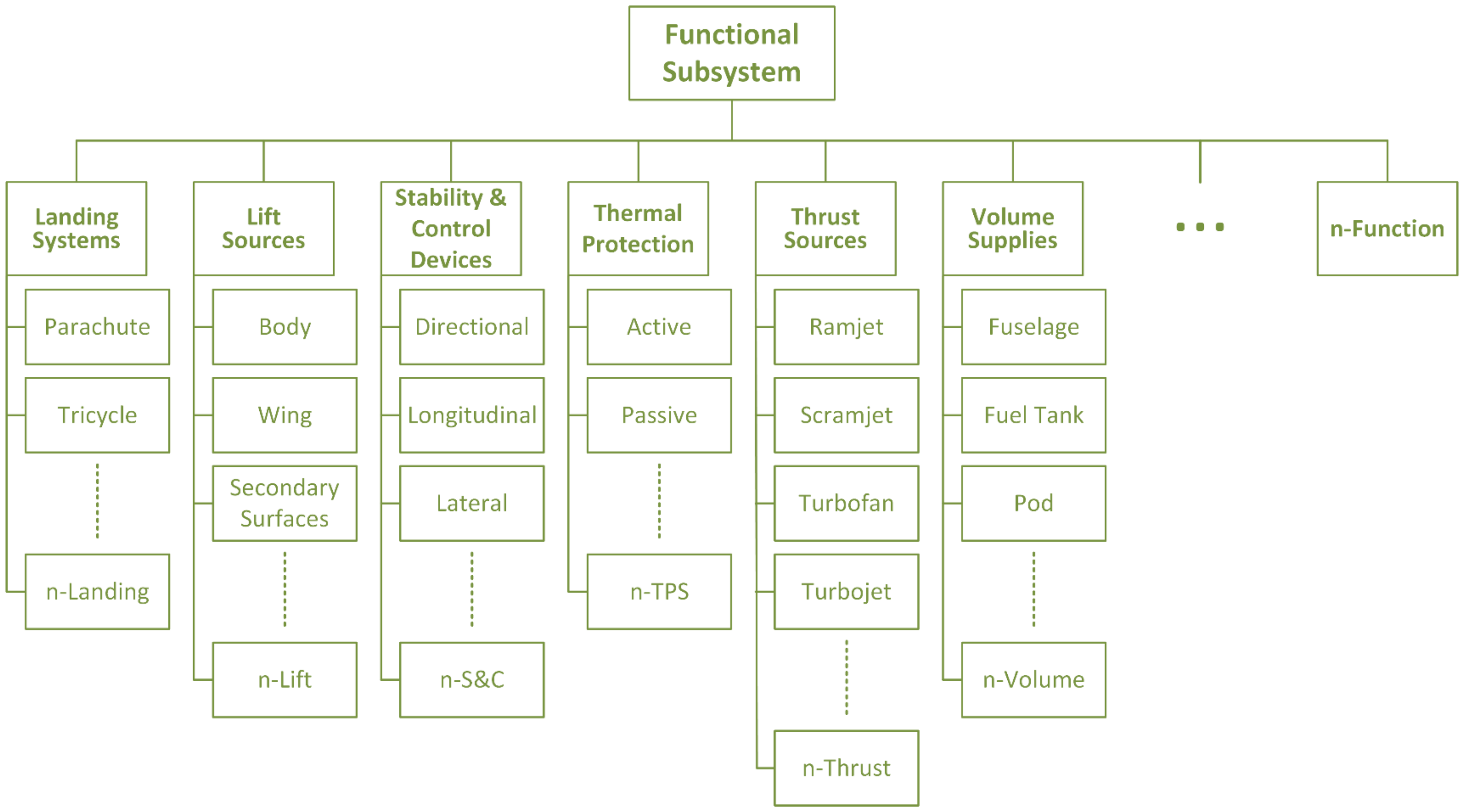


Figure 3-13 Functional Subsystem Block Decomposition

Table 3-3 shows a listing of functional categories; this is representative and is not intended to be a complete listing.

Table 3-3 Description of Hardware Function Categories

Function	**Purpose of Hardware**	**Example(s)**
Drag Source	Provide drag force	Parachute, Autogyro, etc
Landing System	Provide capability to land/recover	Tricycle Gear, Skids, etc
Lift Source	Provide lift force	Wing, Wing Flap, Lifting Body, etc
Stability & Control	Provide stability and/or control	Aileron, Elevon, etc
Thermal Protection	Provide thermal protection	Ablator, Heat Shingle, Heat Pipe, etc
Thrust Source	Provide thrust force	Turbojet, Turbofan, Scramjet, etc
Volume Supply	Supply internal volume	Fuselage, Fuel Tank, Pod, etc

The task sequence for vehicle hardware definition then becomes:

1. Create list of hardware for vehicle to be examined
2. Specify the function(s) that each hardware component, or component group is performing on the vehicle
3. Create Product Breakdown Structure (PBS) with component function information

The functional hardware build-up process creates a buffer between the vehicle hardware definition and the analysis being done to model that vehicle through the implementation of a stand-alone product build-up methodology. Vehicle hardware combinations can be built-up without the need to match and select applicable analysis methods. Also, this process makes it compulsory to define the function of every hardware component attached to the vehicle.

3.2.1.2 Operational Event

Operational Events are non-hardware product attributes that change during the designed use of the product. In terms of aerospace vehicles, this category comprises the design mission type, flight profile, speed range, and altitude range, see Figure 3-14.

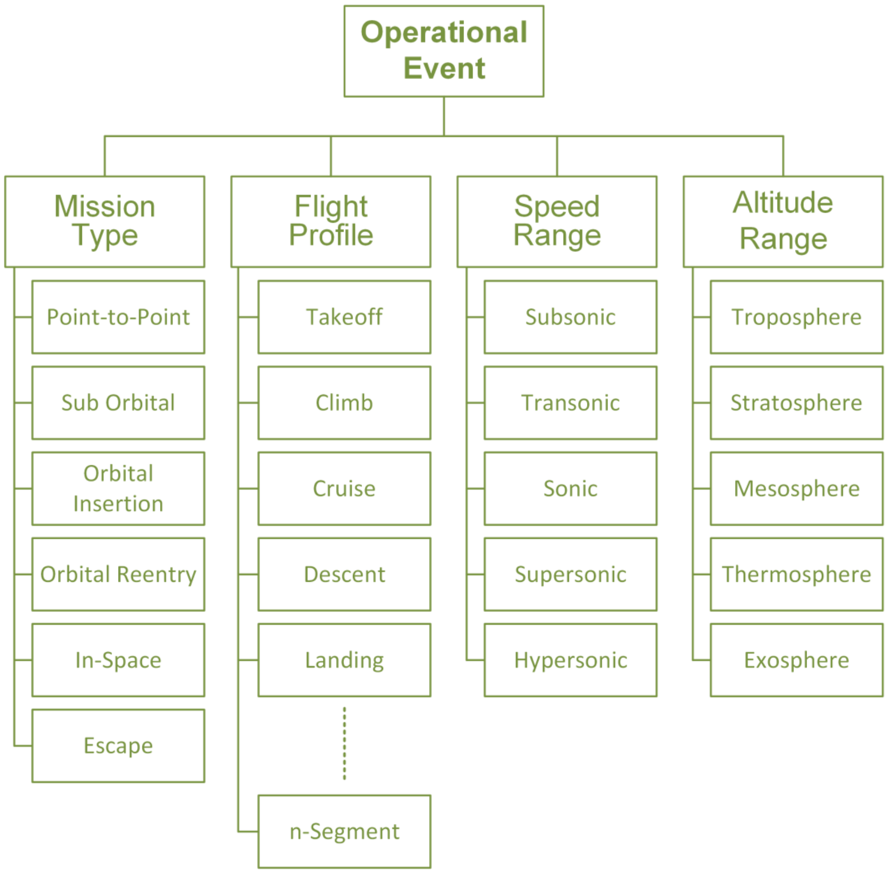


Figure 3-14 Operational Event Block Decomposition

The mission type defines the top-level function of the vehicle. There are six mission type elements, see Table 3-4.

Table 3-4 Description of Mission Types

Type	**Objective of Vehicle**	**Example Vehicles**
Point-to-Point	Move vehicle or payload from one point to another	B747, A320, F22, C-5
Sub Orbital	Reach space (>100 km) without sufficient energy to complete one orbital revolution	Spaceship 2
Orbital Insertion	Reach space (>100km) with sufficient energy to remain at a specific altitude for more than one orbital revolution	Saturn V, Falcon 9
Orbital Reentry	Enter from orbital altitude through planet's atmosphere	Apollo Capsule, Dragon Capsule
In-Space	Perform mission objectives in planetary orbit	ISS
Escape	Provide sufficient energy to escape planetary gravity well	Voyager 1&2

If a vehicle encompasses more than one Mission Type, multiple selections can be made. This case is most often seen by high-speed vehicles, although other exceptions can occur. For example, a Single Stage to Orbit (SSTO) vehicle would be considered an Orbital Insertion and an Orbital Reentry Vehicle. Additionally, if the SSTO vehicle is meant to perform operations while in orbit it would also fall under the In-Space moniker.

The Illustrated Dictionary of Aviation (Kumar, De Remer, and Marshall 2005) defines flight profile as, "*A graphic representation of the flight path of an aircraft in the vertical plane, giving altitude, speed, range, and maneuver of the aircraft as observed from the side*". When assessing flight profiles Vihn (Vinh 1981a) explains, "*It is customary to investigate separately the different phases of a flight profile to assess the respective performances*"; these phases are defined as Trajectory Segments. Each trajectory segment represents a section of the total flight path with a specific objective, see Figure 3-15. Multiple trajectory segment blocks can be selected in order to build up a vehicles flight profile.

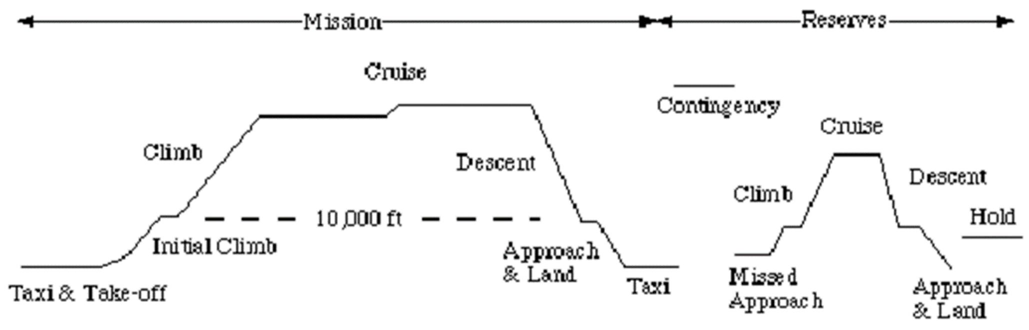


Figure 3-15 Example Flight Profile (Kroo 2006)

The altitude and speed range building blocks are meant to give a representation of flow phenomenon the vehicle is expected to encounter. The speed range has been divided into Mach Number flow regimes, (Figure 3-16), while the Altitude Range is divided into Earth's atmospheric layers (Figure 3-17). In both cases multiple selections can be made according to the expected flight profile and mission type of the vehicle.

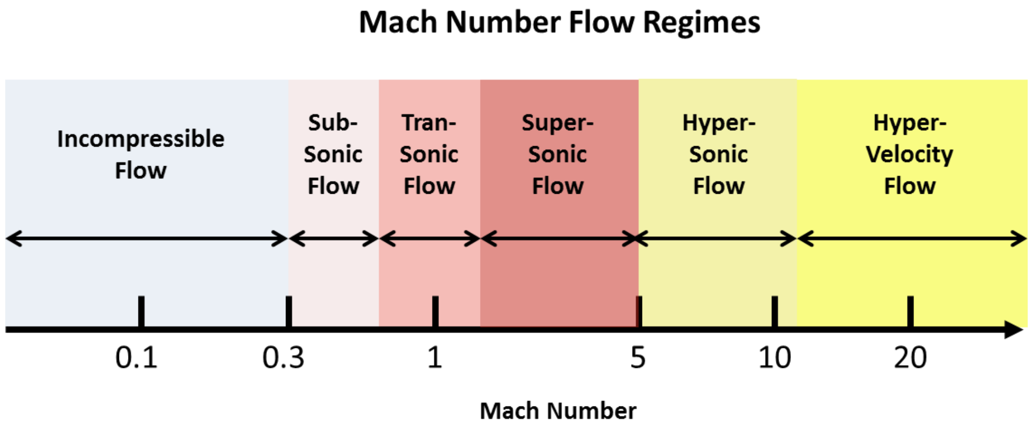


Figure 3-16 Mach Number Flow Regimes (Rchisena92 / CC-BY-SA-3.0)

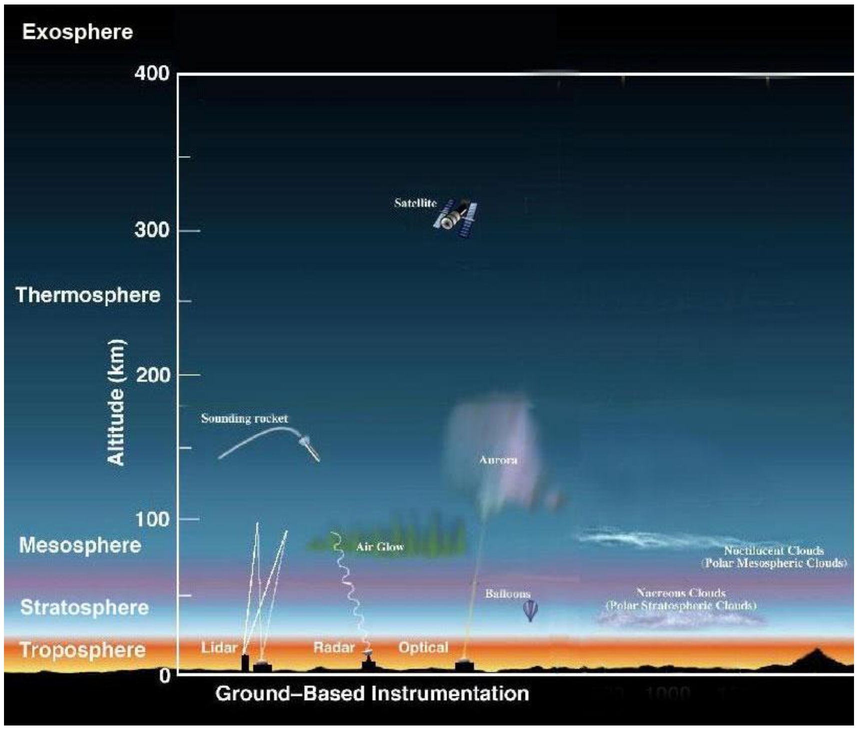


Figure 3-17 Earth's Atmospheric Layers [Adapted from (NASA and Zell 2015)]

3.2.1.3 Operational Requirement

Operational Requirements are non-hardware product attributes that are constant throughout the designed use of the product. In terms of aerospace vehicles these requirements can take the form of Regulations or Specifications, see Figure 3-18.

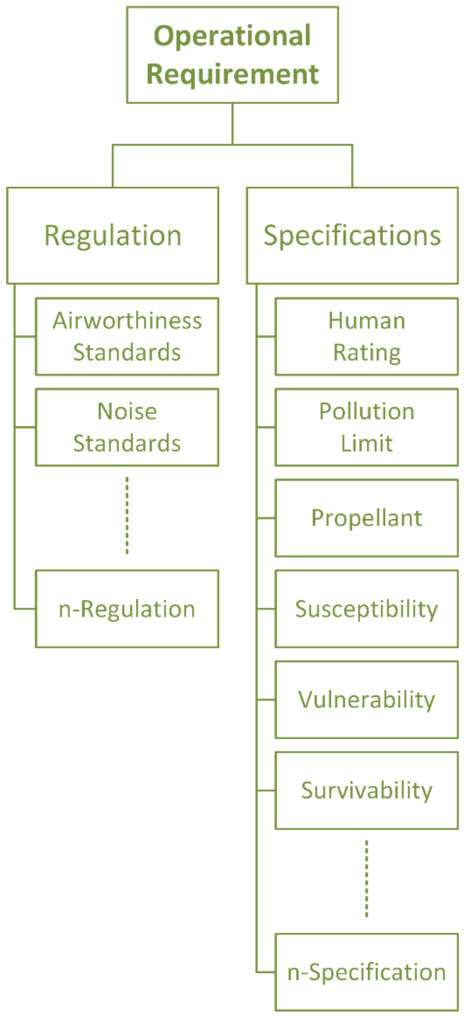


Figure 3-18 Operational Requirement Block Decomposition

Regulations specify any constraints imposed on the vehicle by a governing body (Federal Aviation Administration, International Civil Aviation Authority, etc). Regulations can limit the size (ICAO/FAA 80-meter box rule) or operation (FAA Prohibits Supersonic flight over land) of a vehicle. Regulations tie vehicle parameters to constraints that must be achieved, they do not deal with the optimization of those parameters.

A Specification can be seen as a non-hardware design parameter of interest to the user, or stakeholder. These parameters can be design choices such as propellant type, or Human Rating. Other Specifications are inherently tied to a constraint or optimization where a level or magnitude for the parameter must be specified. For example, a Pollution Limit must be accompanied by a defined set of chemicals to measure as well a limit or a goal for the concentration of those chemicals.

There are scenarios where a Specification parameter can be that same as one defined in a Regulation. In these case, the user or stakeholder has made a decision to not only meet a regulation standard for a parameter, but to exceed it.

3.2.2 Analysis Process Blocks

The Analysis Process is defined as any information relating to the overall organization and integration of an Aerospace Synthesis System. The Analysis Process is broken into two classes of information: System Elements, and Disciplinary Elements, see Figure 3-19.

The System Elements describe the top-level objective function of the Synthesis System. An objective function is generally referred to as a function whose value is meant to be maximized, minimized, or driven to zero. An objective function is comprised of independent and dependent variable. In the context of aerospace synthesis systems, the independent variables are the initial guesses to start the process; the AVDS sizing process begins with an assumption of Planform Area and Vehicle Wing Loading. The dependent variable in this case are the other variable in the objective function. These variable should be solved for throughout the synthesis process and represent the output of disciplinary analysis. Again referring back to AVDS Sizing process, the dependent variables are

Takeoff Gross Weight, Operating Weight Empty-Weight, and Operating Weight Empty-Volume.

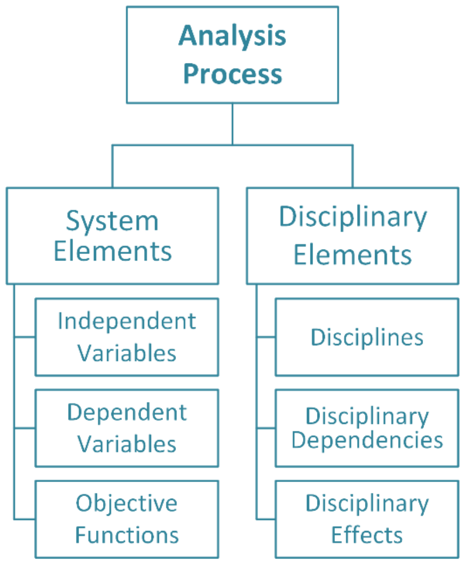


Figure 3-19 Analysis Process Block Decomposition

The Disciplinary Elements describe the how a system is integrated in terms of its disciplinary requirements. This description contains three type of information:

- Disciplines – A listing of disciplinary analysis modules and the run order contained in the analysis process
- Disciplinary Dependencies – The input parameters that define the degrees of freedom of a disciplinary analysis module; e.g. if the Aerodynamics module dependencies are Altitude, Velocity and Angle of Attack, then any Aerodynamic lookup table would be a function of those three parameters only.
- Disciplinary Effects – The output variables that are solved for by the disciplinary module; e.g. if the Aerodynamics module effects are C_L, C_D, and C_M, then any aerodynamic analysis being done in the module must, at minimum, estimate values for those three effects.

3.2.3 Disciplinary Method Blocks

Disciplinary Methods are defined as any analytic function meant to solve for the disciplinary effects defined in the analysis process. Disciplinary methods are broken into three classes of information: Product Model, Variable, Analysis, see Figure 3-20.

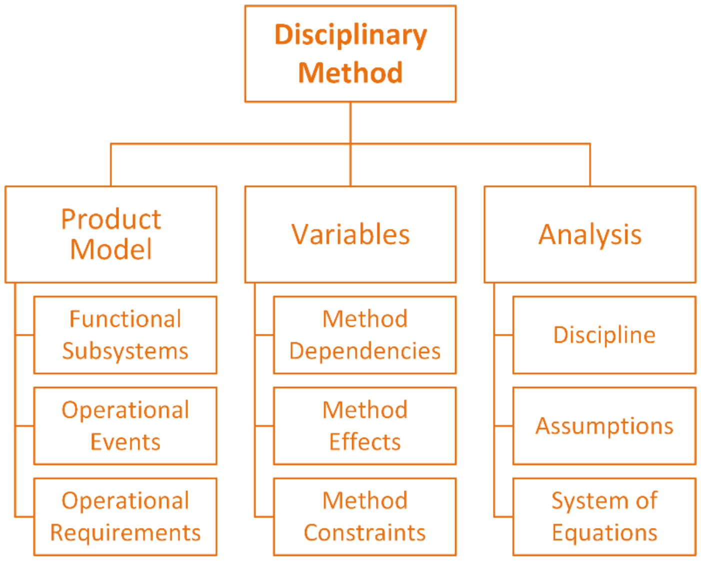


Figure 3-20 Disciplinary Method Block Decomposition

The Product Model is analogous to the Product building blocks described previously. In the case of the Product building blocks the intent has been to characterize the vehicle in terms of its hardware, operational events and operational requirements in a holistic sense. In this case the Product Model is meant to describe the applicability of a given Disciplinary Method toward specific hardware, operational events, or operational requirements; e.g. a given propulsion method might only be applicable to model scramjet engines.

The Variables element of the disciplinary method is similar to that of the disciplinary element in the analysis process. Three type of variables are described:

- Method Dependencies – Defines the input needed for the disciplinary method; the knowns.
- Method Effects – The output variables that are solved for by the disciplinary method; the unknowns
- Method Constraints – A listing of variables and associated magnitude ranges that the disciplinary method is valid over; e.g. an Aerodynamic method might only be valid when the Mach Number is less than 0.7.

The analysis element contains the systems of equations defining the disciplinary analysis. The system of equations can be comprised of empirical relationships, lookup tables, nomograms, etc. Additionally, the analysis element contains data pertaining to the classification of the method in terms of discipline and assumptions.

3.2.4 Decomposition Process

The description to this point has been focused on the definition of the components required to build a CMDS. The application of these definitions allows for a systematic capability to review and capture synthesis system knowledge from literature; Figure 3-21 shows a general concept for the process of CMDS decomposition.

A notional synthesis system is first separated into its three constituent building block types: Product, Analysis Process, and Disciplinary Methods. These three types of data are then separately recorded into a Database through a Database Management System (DBMS). The result of this process is a Database (DB) containing Individual Product, Analysis Process, and Disciplinary Method Data. Additionally, there is a System Architecture layer that stores the connection between these three building blocks. This means that once input into the DB, a selection can be made for either the Product, Analysis Process, or Disciplinary Methods associated with the CMDS, or the CMDS itself can be reconstructed as a combination of the three building blocks.

Another aspect of the CMDS decomposition process is the visibility of uniqueness of the parts that comprise a CMDS. The bottom right corner of Figure 3-21 shows that “Specific” and “Common” components are deciphered before entry into the DBMS. This prevents the duplication of building block data found in several CMDSs. Additionally, the tracking of common building blocks with allows the tracking of specific information as to how and where these components have been implemented in the past.

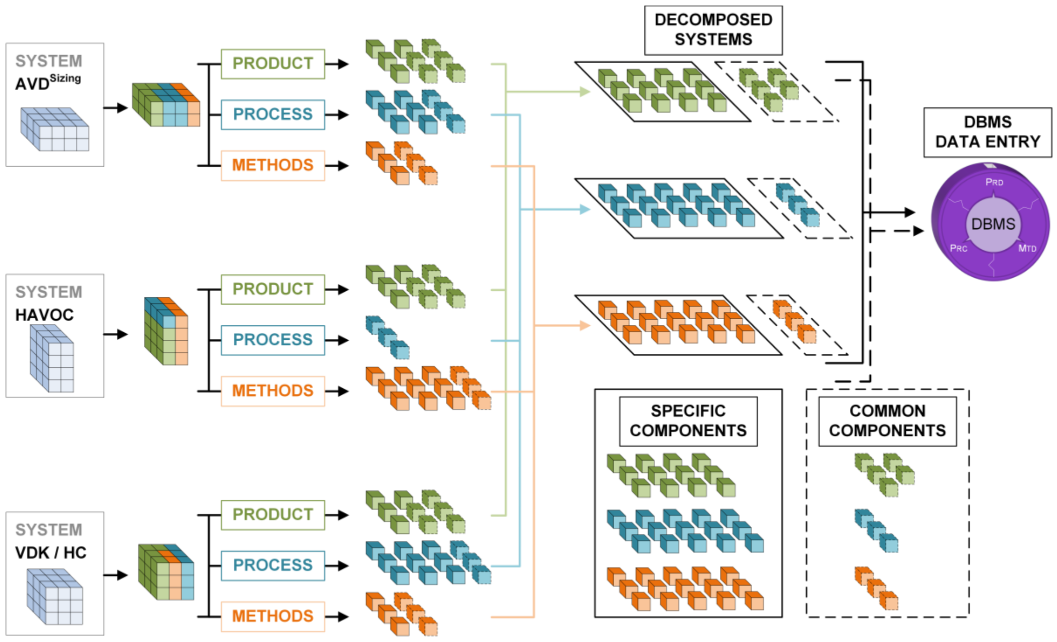


Figure 3-21 CMDS Decomposition Process

3.3 Aerospace Synthesis System Generation - CMDS Composition

The result of CMDS decomposition yields the building blocks for each of the parts of the synthesis system: Product, Analysis Process, and Disciplinary Methods. Each building block adheres to a standardized interface specific to the data type. This modular approach allows for building blocks of each type to be selected and integrated together to create new CMDSs.

The generation of a CMDS begins with the decision of a Product to be modeled and the Analysis Process used as the framework for analysis. The Product and Analysis Process breakdown results from a coupling of the stakeholder requirements and expected deliverables. This process is iterative, as the development lifecycle progresses and alters the scope of the problem, the Product and Analysis Process will need to be changed. The strategy and/or methodology used to create Product Breakdown and an Analysis Process as a function of Stakeholder Requirements is beyond the scope of the current research endeavor. As so, the CMDS Generation assumes that a Product and Analysis Process have been defined a priori.

The methodology for the generation of a CMDS follows four sequential actionable steps: Matching, Selecting, Arranging and Generation. Each of these steps describe the action taken to on the Disciplinary Method building blocks, in order to compose them with pre-defined Product and Analysis Process building blocks. It should be noted that the following description of the CMDS Generation Methodology assumes that a database management system exists with the capability to store the building block information from the decomposition process. The remainder of this section describes the use of this capability to gather individual Product, Analysis Process, and Disciplinary Methods building blocks and integrate them in order to create a tailor made CMDS.

3.3.1 Matching

The Matching phase queries and returns all disciplinary methods that are applicable to the problem requirements, namely the product and analytics process. Table 3-5 shows the details the Product and Analytic Process input attributes as well as their accompanying Method attributes. The resulting list of disciplinary methods contains all of the attribute information for each method; see earlier discussion of disciplinary method building blocks. Figure 3-22 details the sequence used in the method matching process.

Table 3-5 CMDS Matching – Method Matching Attributes

Product Matching		**Analytic Process Matching**	
Product Element	*Method Element*	*Process Element*	*Method Element*
Functional Subsystem	Functional Subsystem	Discipline	Discipline
Operational Event	Operational Event	Disciplinary Dependencies	Method Dependencies
Operational Requirement	Operational Requirement	Disciplinary Effects	Method Effects

The first step in the process queries all disciplinary methods that are applicable to the hardware selected for the product. This process is iterative as it cycles through each hardware function and subsequently through each hardware piece and returns methods having a matching hardware attribute; this creates a listing of method candidates. The second step in the process applies the mission and operational requirement inputs from the product definition and reduces the number of candidate disciplinary methods. Next the input requirements from the analytic process definition are applied. The discipline attribute of each method is compared to the listing of disciplines found in the analytic process definition. The last step in the process compares the input and outputs variables of each method. The methods that contain input and output variables matching the defined disciplinary process input and outputs variable requirements are the output of the matching process. This new listing contains all possible disciplinary methods applicable to the product and analytic process defined.

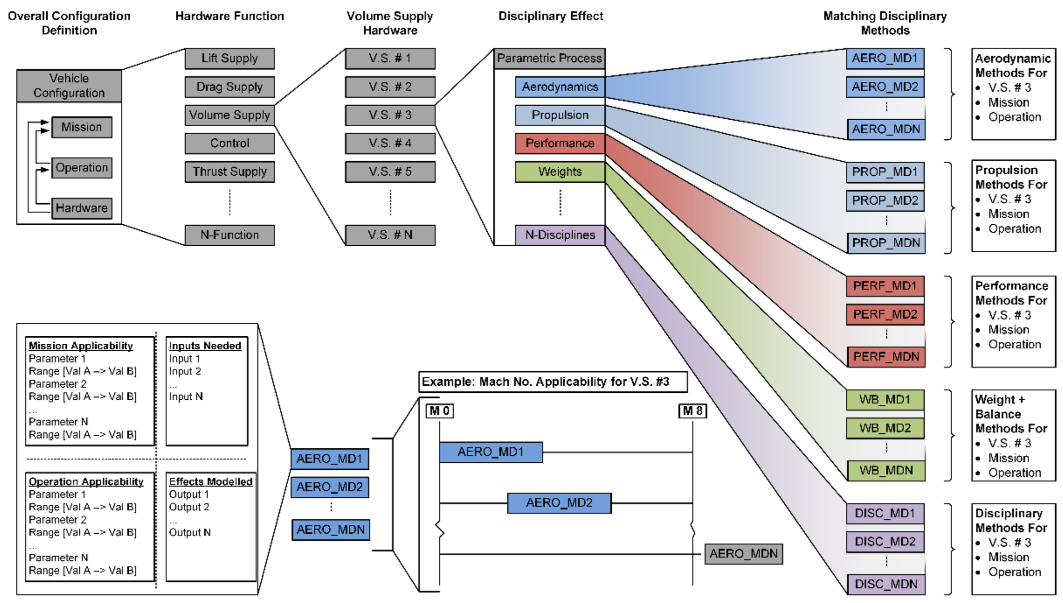


Figure 3-22 CMDS Matching

3.3.2 Selecting

The Selecting phase reviews all disciplinary methods returned from the Matching phase, and selects those that will be integrated into the CMDS. This step in the process is highly user-inclusive and is not meant to be done in an automated fashion. The engineer creating the CMDS selects the methods he/she feels best represent the problem they are trying to solve. That being said, the selection of disciplinary methods can be aided through the visualization of method specific information and the cross referencing of that information to the problem input requirements.

The DBMS must keep track of each method selected and show in some form what selections still need to be made in order to complete a CMDS, see Figure 3-23. An example of this can be seen in the bottom left hand corner of Figure 3-23, where three aerodynamic methods have been selected (AERO_MD1, AERO_MD2, AERO_MD3). In this case each of the aerodynamic methods have a constraint associated with the range of Mach numbers they are applicable to. This information must be shown to the user during the method Selecting phase to avoid a CMDS that does not syntactically cover the specified problem requirements. If a CMDS is being created to model a vehicle that flies from Mach 0.0 to Mach 8, methods should be selected that at minimum cover this range without gaps in applicability.

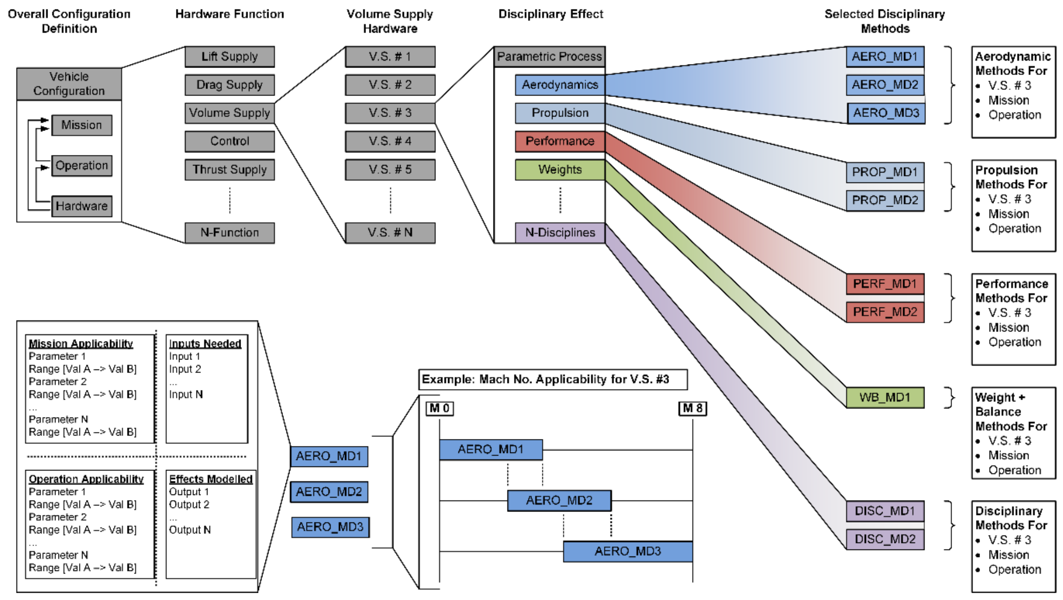


Figure 3-23 CMDS Selecting

It should be noted that this example only covers the selection of Disciplinary Methods for a single discipline (Aerodynamics), for a given hardware (Volume Supply #3). The process of selecting methods and checking each method specific constraint to that of the vehicle input requirements is iterative, and must be done for all combinations of hardware and disciplines. In order to ensure a syntactic composability a query must be made to ensure this requirement has been rigorously adhered to. The rules for this query are as follows:

- Product Hardware – There must be at minimum one Disciplinary Method selected per Product Hardware defined in the Matching phase.
- Disciplines – There must be at minimum one Disciplinary Method selected per Discipline defined in the Matching phase.
- Trajectory Segments – There must be only one Disciplinary Method selected per Trajectory Segment defined in the Matching phase.
- Atmospheric Model – An Atmospheric Model must be selected

3.3.3 Arranging

The Arranging phase assesses the combination of Product, Analysis Process and Selected Disciplinary Methods, and creates an integration blueprint for the DBMS. The integration blueprint is comprised of a Run Order for the selected Disciplinary Methods, and a listing of all variables input into and created by the DBMS.

Up to this point the Matching and Selection phases have focused on individual hardware components and the assignment of disciplinary methods to model that components; Figure 3-23 shows the selection of Disciplinary Methods from multiple disciplines meant to model the hardware component VS #3. The arranging phase takes this information and re-organizes it in order to group the selected methods by discipline; e.g. all of the selected aerodynamics methods meant to model all of the vehicle components are grouped together, see Figure 3-24. This change is necessary in order to provide a blueprint that adheres to the discipline run order defined in the Analysis Process.

Each group of disciplinary methods must be arranged in an order that guarantees that each method has the correct input information available. This means that the input and out variables for each method must be catalogues and cross referenced in order to assess where each of its input variables will be coming from. Variables input into a disciplinary method can originate from two sources: the input file, or run-time generation. For the latter case, run-time generation, it is necessary to track where the variable is created and where it is stored so that it can be retrieved appropriately when needed later in the run-time setting.

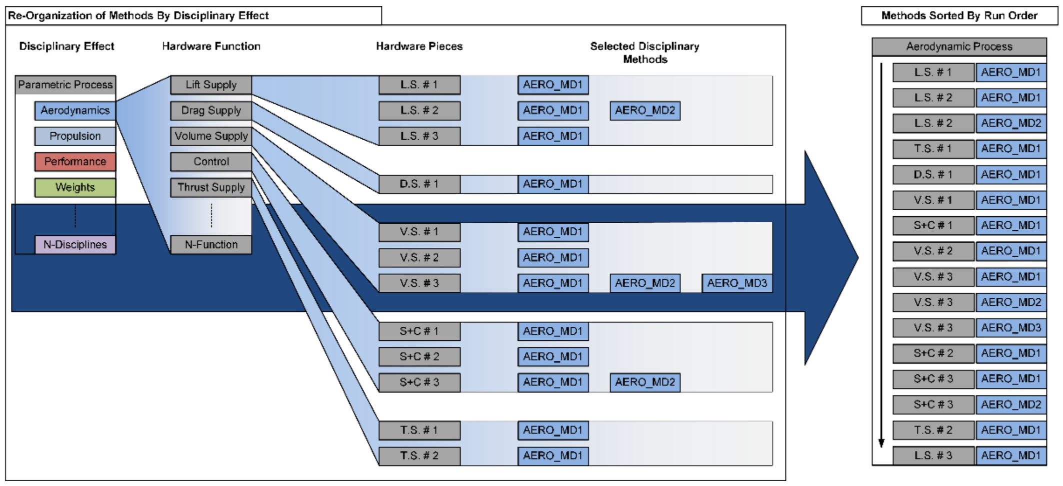


Figure 3-24 CMDS Arranging

An additional task of the Arranging phase is the tracking of variables needed for the Objective Function defined in the Analysis Process. As stated earlier, the Objective Function contains independent and dependent variables. The initial guess for the values of the independent variables are found in the input file; an initial guess is used as the independent variables are used to drive the objective function, and change throughout run-time. The dependent variables are created during disciplinary analysis and are nominally grouped with the other disciplinary methods outputs. In order ensure syntactic composability the Arranging phase verifies that all objective function dependent variables are created during run-time and are accessible by the objective function.

3.3.4 Generation

The Generation phase creates an analysis architecture based on the analysis blueprint created in the arranging phase. Up to this point every phase in the CMDS

Generation process has been wholly contained in the DBMS setting. The Generation phase differs in this respect as its output is meant to be a self-contained executable, where the execution setting is not in the purview of the CMDS. There are two main components of the CMDS Generation phase: Input Parameter Listing and Analysis Architecture, see Figure 3-25.

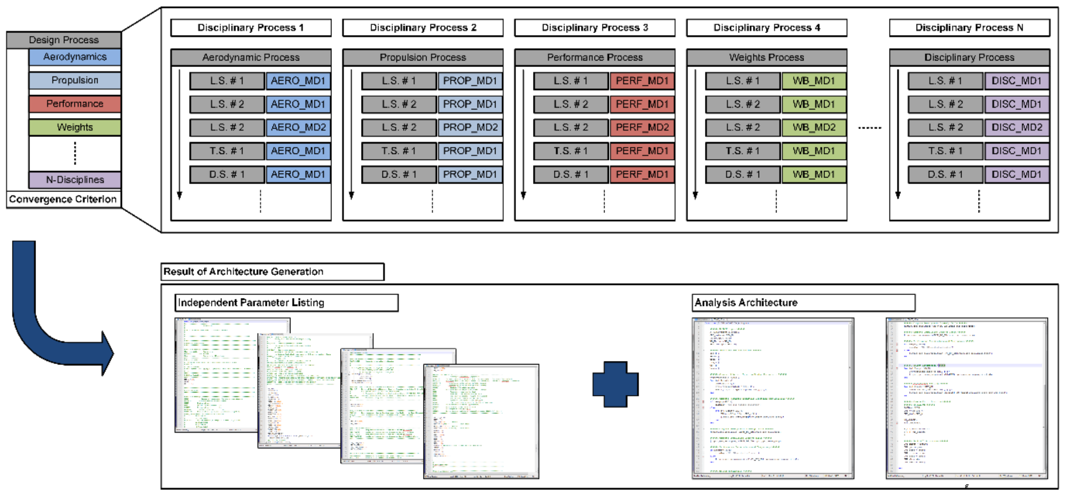


Figure 3-25 CMDS Generation

The Input Parameter Listing is the input file for the generated CMDS, it contains the system level input variables as well as input variables required by each disciplinary method in the CMDS. The Analysis Architecture is comprised of three classes of files:

- System Process – Objective function, and orders disciplinary process function calls
- Disciplinary Process – Orders disciplinary method function calls
- Disciplinary Method – Contains disciplinary analysis function calls

The purpose of the Generation phase is to convert the CMDS blueprint into source code to create a tailor-made CMDS. Every execution of the CMDS Generation process will yield a new CMDS source code only containing the components needed to solve the given problem.

3.4 Solution Concept Summary

The goal of this research has been defined as the creation of an environment with the purpose of tailor making synthesis systems for aerospace vehicle conceptual design. A review of aerospace synthesis systems led to the realization that a solution would need to include techniques not traditionally used in aerospace conceptual design.

The Systems Engineering Process has been applied to create a decomposition methodology aimed at reducing an aerospace synthesis system into its constituent building blocks, namely Product, Analysis Process, and Disciplinary Methods. M&S Simulation Composability has been applied to create a composition methodology, with the capability to create a tailor-made CMDS through the composition of those building blocks. The CMDS Composition process has been shown to contain four sequential steps: Matching, Selecting, Arranging, and Generation.

A generic methodology has been defined for CMDS Composition. This methodology has been made with a focus on what tasks must be achieved and with the assumption that DBMS exists with the capability to perform these tasks. Due to the nature of the derivation the environment for both the DBMS and the resulting analysis framework has been left open. In order to transition the CMDS Composition methodology to a functioning system, the methodology must be implemented in a specified software environment.

Chapter 4

Software Implementation

The solution concept has called for systematic process to convert user input and problems specification into specific analysis frameworks. The CMDS Composition specification derived in Chapter 3 is comprised of a four step process (Matching, Selecting, Arranging, and Generation) meant to systematically evolve a user's problem description into an analysis framework meant to solve said problem. As a mean of assessing the validity of the CMDS Composition process a prototype system (AVD^{DBMS}) has been developed. AVD^{DBMS} is comprised of three distinct layers: The Graphical User Interface (GUI), the database layer, and the analysis layer. Each layer is not only distinct in its application but also the software used to create it and programming language in which it is written, see Table 4-1 and Figure 4-1.

Table 4-1 AVD^{DBMS} Software Used

Layer	Software	Programming Language
GUI Layer	Microsoft Access	Microsoft Visual Basic with Applications (VBA)
Database Layer	Microsoft Access	Search Query Language (SQL)
Analysis Layer	MATLAB	MATLAB Script

The GUI Layer has been created using Microsoft Access relational database program, and is implemented using the VBA programming language. This layer serves as the front end for the database, and is the only part of AVD^{DBMS} that the user can directly input or adjust data. The database layer has also been created in Microsoft Access relational database program, and serves as the backend for AVD^{DBMS} where all data is stored. SQL has been used to both create tables as well as facilitate data transfer between the GUI and Database layers. The user never directly adds to or adjusts data in the back end table, any data entry must be done through input forms contained in the GUI Layer.

The final part of AVD^{DBMS} is the Analysis Layer; which has been implemented in the MATLAB workspace environment. The Analysis Layer contains the source code for each of the Disciplinary Methods in the Methods Library as well as each CMDS output from the CMDS Composition process.

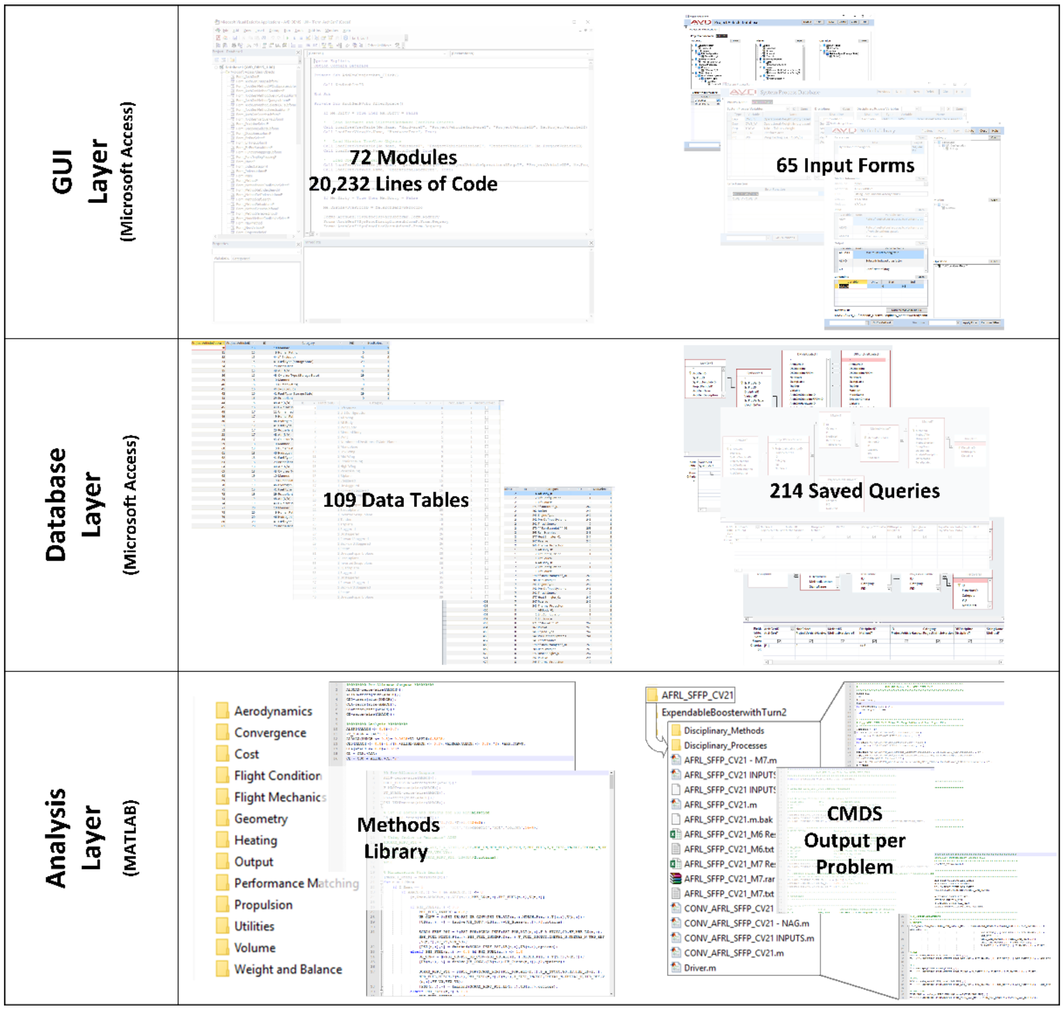


Figure 4-1 AVD DBMS Three Layer Architecture

The description of the AVD^{DBMS} software is broken into three sections: Utility Modules, Building Block Input Mechanisms, and CMDS Composition Framework. Each section will provide a functional description based on the implementation of each of the three AVD^{DBMS} Layers.

4.1 Utility Modules

The main components of the AVD^{DBMS} system are the Building Block Input Mechanisms and the CMDS Composition Framework. In the process of creating those modules it became necessary to create segments of code that are shared between both of them. These modules have been termed utility modules and will be discussed in the next few sections.

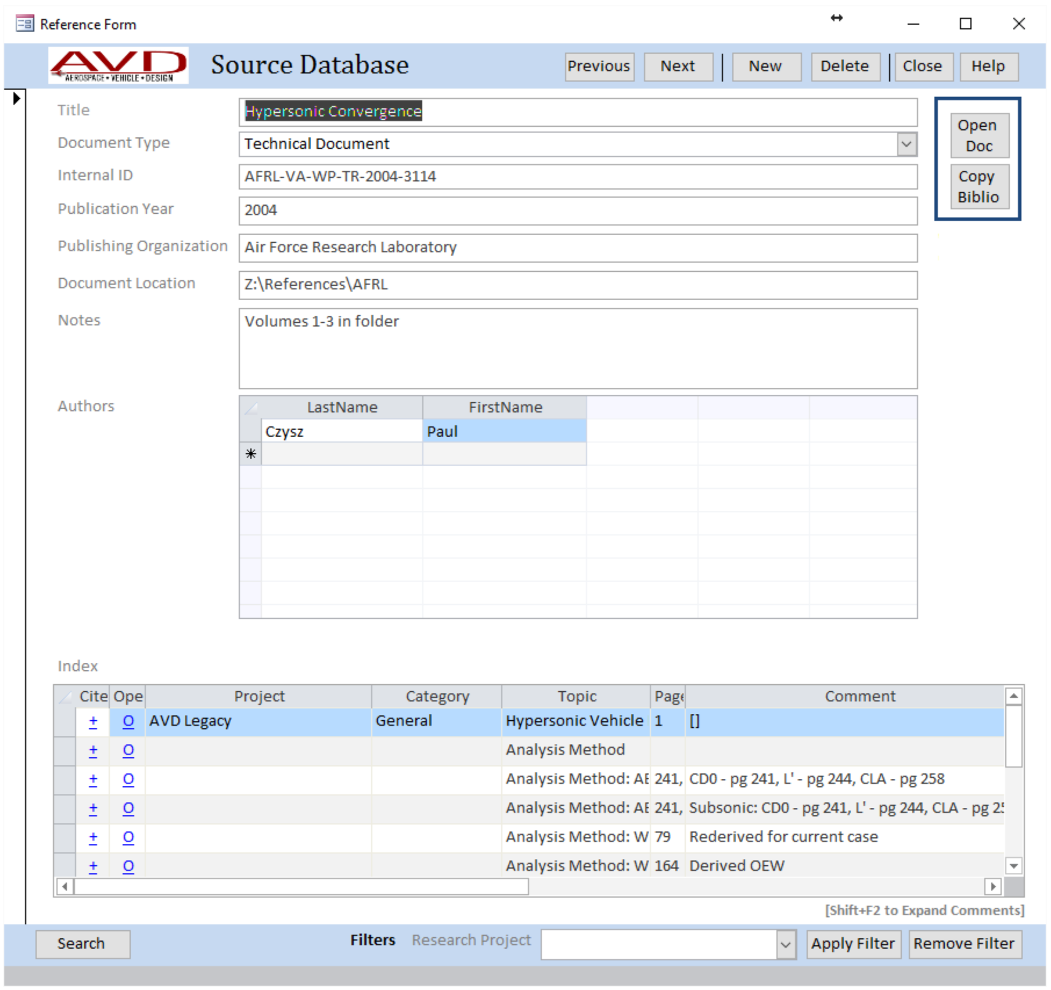
4.1.1 References Input Form

The Reference Input Form is the mechanism enabling the capture of data and knowledge from source material and preparing it for use in the CMDS composition process. The form is separated into 2 input sections, the first half deals with citation data meant to describe the reference, the second deals with index data meant to describe information held within the reference. A listing of Reference Form Input parameters and an example of the Reference form user interface can be seen in Table 4-2 and Figure 4-2 respectively.

Table 4-2 Reference Form Input Parameters

Input Field	**Type**	**Description**	**Table**
Title	String	Title of Reference	MainT
Document Type	String	Describes class of reference: Book, Press Release, Technical Document, Contract Report, Thesis/Dissertation, Presentation, Website, Patent	MainT
Internal ID	String	Any identification number associated with reference	MainT
Publication Year	Integer	Year reference was published	MainT
Publishing Organization	String	Organization where reference was published	MainT
Document Location	String	Location of document at AVD Laboratory	MainT

Notes	String	Notes or keywords associated with reference. This field is used as a means to search for references	MainT
Authors	Table	First and last name of reference authors	AuthorT
Index	Subform	An index refers to a specific piece of data found on one or many pages in the reference. Indexes are analogous to a post-it note placed on a page in a book. Figure 4-1 shows several Index examples; the indexes highlight aerodynamic methods, show a brief description of the method ouput variables and point to the page numbers where the methods are found	IndexT



Title	Hypersonic Convergence
Document Type	Technical Document
Internal ID	AFRL-VA-WP-TR-2004-3114
Publication Year	2004
Publishing Organization	Air Force Research Laboratory
Document Location	Z:\References\AFRL
Notes	Volumes 1-3 in folder

Authors

LastName	FirstName
Czysz	Paul

Index

Cite	Ope	Project	Category	Topic	Page	Comment
+	O	AVD Legacy	General	Hypersonic Vehicle	1	[]
+	O			Analysis Method		
+	O			Analysis Method: AE	241,	CD0 - pg 241, L' - pg 244, CLA - pg 258
+	O			Analysis Method: AE	241,	Subsonic: CD0 - pg 241, L' - pg 244, CLA - pg 25
+	O			Analysis Method: W	79	Rederived for current case
+	O			Analysis Method: W	164	Derived OEW

[Shift+F2 to Expand Comments]

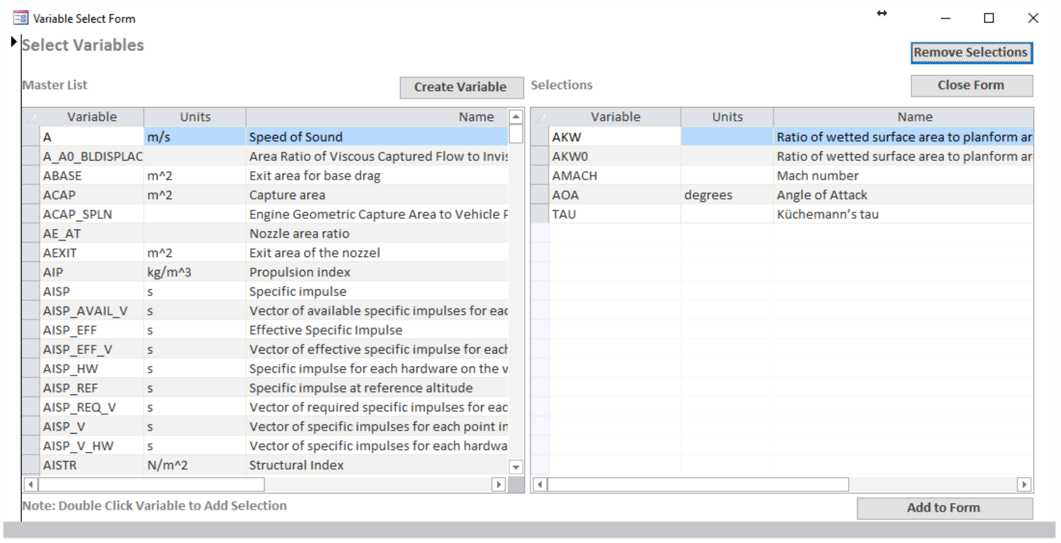
Research Project

Figure 4-2 Reference Input Form

4.1.2 Variable Input Form

A cornerstone of the CMDS Composition process is the ability to track and classify input and output variables throughout the analysis framework. In order to facilitate this action and to ensure that duplicate variables are not created, a variable input form has been created, see Figure 4-3. A variable is defined in our system as containing three types of information:

- Variable syntax used in analysis source code. This is how the variable will appear in disciplinary methods.
- Units associated with the variable *Note standard metric units are used in AVD^{DBMS} whenever possible.
- A brief description of the variable



Variable	Units	Name
A	m/s	Speed of Sound
A_A0_BLDISPLAC		Area Ratio of Viscous Captured Flow to Invis
ABASE	m^2	Exit area for base drag
ACAP	m^2	Capture area
ACAP_SPLN		Engine Geometric Capture Area to Vehicle F
AE_AT		Nozzle area ratio
AEXIT	m^2	Exit area of the nozzel
AIP	kg/m^3	Propulsion index
AISP	s	Specific impulse
AISP_AVAIL_V	s	Vector of available specific impulses for eac
AISP_EFF	s	Effective Specific Impulse
AISP_EFF_V	s	Vector of effective specific impulse for each
AISP_HW	s	Specific impulse for each hardware on the v
AISP_REF	s	Specific impulse at reference altitude
AISP_REQ_V	s	Vector of required specific impulses for eac
AISP_V	s	Vector of specific impulses for each point in
AISP_V_HW	s	Vector of specific impulses for each hardwa
AISTR	N/m^2	Structural Index

Variable	Units	Name
AKW		Ratio of wetted surface area to planform ar
AKW0		Ratio of wetted surface area to planform ar
AMACH		Mach number
AOA	degrees	Angle of Attack
TAU		Küchemann's tau

Note: Double Click Variable to Add Selection

Figure 4-3 Variable Subform

The Variable Input form has two sections: Master Variable List and Selected Variable List. The Master Variable List contains every variable that exists in the AVD^{DBMS} library. If a Variable is needed that does not exist, it can be added by clicking on "Create

Variable”; this bring up the “Add New Variable” subform, Figure 4-4. In order to select a variable, the user double clicks on the variable name; this action adds the variable to the Selected Variable list. The Variable Subform is used throughout the Building Block Input forms as well as the CMDS Composition forms and is implemented as a pop-up Subform when needed.

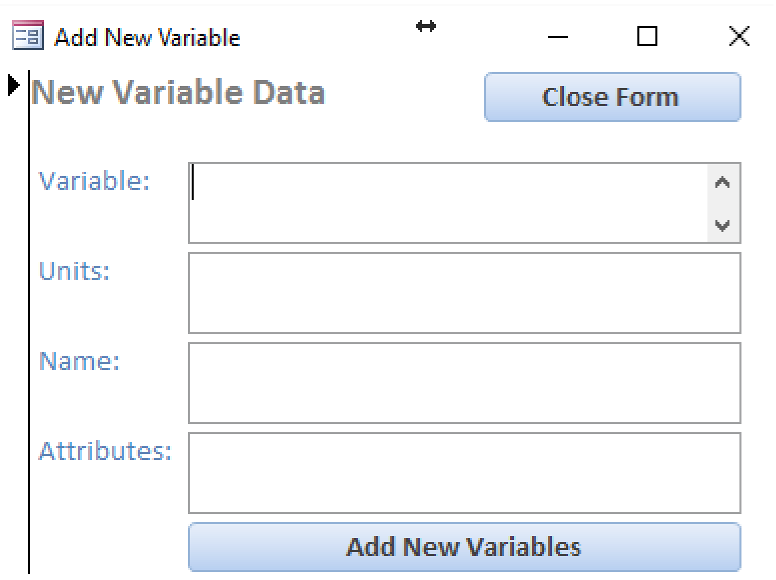


Figure 4-4 Add New Variable Subform

4.1.3 Input Tree Diagrams

The application of the SEP towards the decomposition of aerospace products has led to the definition of three classes of information: Hardware, Operational Events and Operations Requirements. Each class of information contains its own set of possible input data, and set of dependencies. The solution found in the SEP is the use of Functional Analysis in the definition of each of these product categories. The main feature of functional analysis is the hierarchical structure that the data conforms to. In order to implement this type of setting, an input mechanism was needed that allowed for quick and easy building of hierarchical relationships.

The solution implemented uses a pop-up Subform containing the Microsoft TreeView Control. Where the “TreeView control displays a hierarchical list of Node objects, each of which consists of a label and an optional bitmap. A TreeView is typically used to display the headings in a document, the entries in an index, the files and directories on a disk, or any other kind of information that might usefully be displayed as a hierarchy” (MSDN 2016). Figure 4-5 shows examples of input forms for each of the three Product classes.

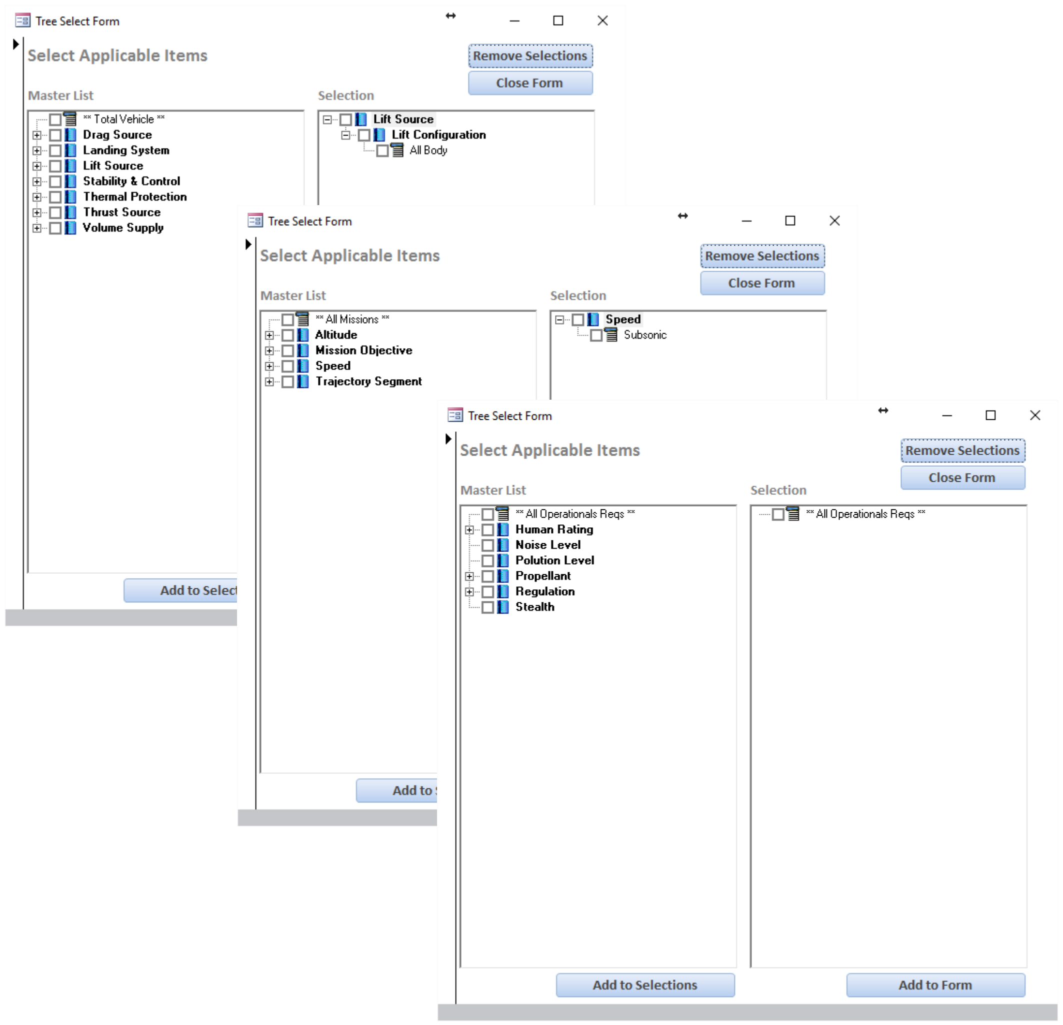


Figure 4-5 Product Input Tree Subform

The Input Tree Subform has two parts: Master List, and Selected List. The Master List contains all entries for the given Product class (Hardware, Operational Even, or Operational Requirement) contained in AVD^{DBMS}. There is a check box next to each entry, checking this box selects the node; this process is repeated until all required nodes are selected. With all of the required nodes checked, clicking on the “Add to Selections” button moves the selected nodes to the “Selections” tree on the right side of the subform. This process is repeated until the “Selections” tree contains all required nodes. Clicking on the “Add to Form” button takes the “Selection” tree and augments the form that originally called the tree Subform.

4.2 Building Block Input Mechanism

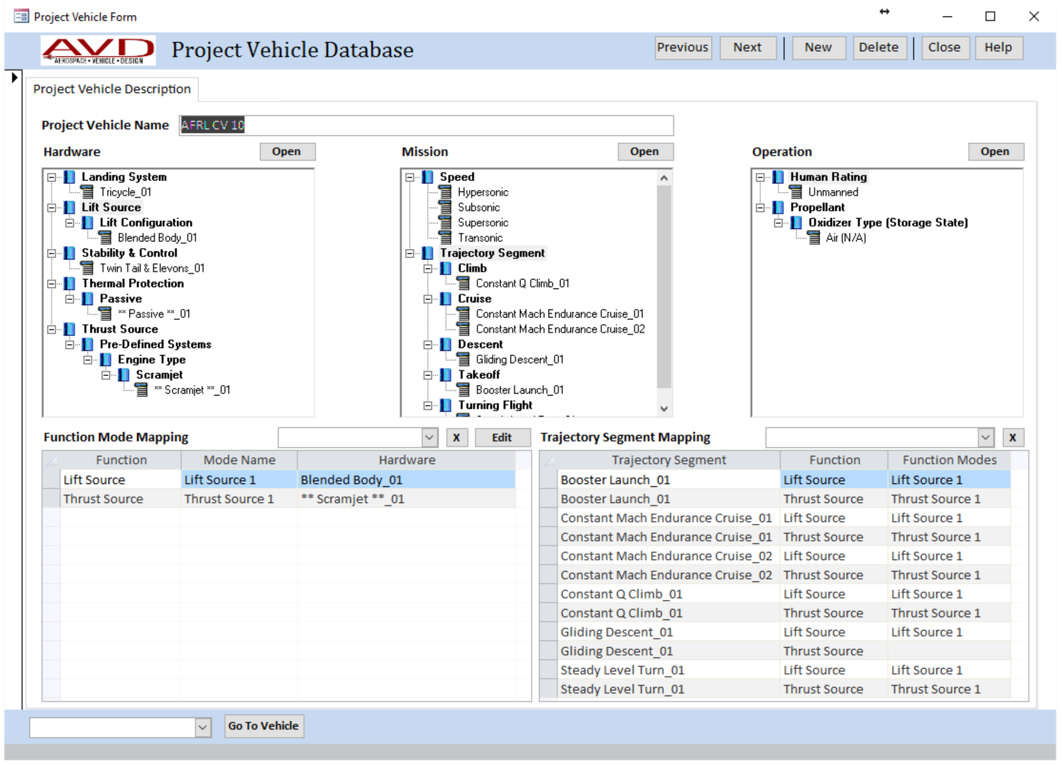
The decomposition effort in Chapter 3 has yielded three separate classes of information necessary to characterize an Aerospace Synthesis system, namely the Product being modelled, Analysis Process guiding the analysis and the Disciplinary Methods used to model the product. Each of these classes of information contains a specific breakdown of constituent parts and interdependencies. In order to facilitate a prototype system for CMDS Composition, it is required to first have the capability to input and store data for each building block class. The following section describe the input mechanism for the Product, Analysis Process, and Disciplinary Methods building blocks.

4.2.1 Product

The Product building block is comprised of three parts: Hardware, Operational Events, and Operational Requirements. A listing of Product Form Input parameters and an example of the Product Input Form can be seen in Table 4-3 and Figure 4-6 respectively.

Table 4-3 Product Form Input Parameters

Input Field	Type	Description	Table
Project Vehicle Name	String	Name of Product being modelled	ProjectVehicleT
Hardware	Tree	Listing of hardware components of product	ProjectVehicleHardwareT
Mission	Tree	Listing of Operational Events being modelled	ProjectVehicleMissionT
Operation	Tree	Listing of Operational Requirements being modelled	ProjectVehicleOperationsIReqsT
Function Mode Mapping	Subform	Definition of function modes for product. A function mode is defined as a group of hardware components of the same function type which are active/working at the same time	ProjectVehicleFunctionModeT
Trajectory Segment Mapping	Subform	Assignment of specific function modes for each trajectory segment	ProjectVehicleTrajSegT



Project Vehicle Name: AFRL CV 10

Hardware
- Landing System
  - Tricycle_01
- Lift Source
  - Lift Configuration
    - Blended Body_01
- Stability & Control
  - Twin Tail & Elevons_01
- Thermal Protection
  - Passive
    - ** Passive **_01
- Thrust Source
  - Pre-Defined Systems
    - Engine Type
      - Scramjet
        - ** Scramjet **_01

Mission
- Speed
  - Hypersonic
  - Subsonic
  - Supersonic
  - Transonic
- Trajectory Segment
  - Climb
    - Constant Q Climb_01
  - Cruise
    - Constant Mach Endurance Cruise_01
    - Constant Mach Endurance Cruise_02
  - Descent
    - Gliding Descent_01
  - Takeoff
    - Booster Launch_01
  - Turning Flight

Operation
- Human Rating
  - Unmanned
- Propellant
  - Oxidizer Type (Storage State)
    - Air (N/A)

Function Mode Mapping

Function	Mode Name	Hardware
Lift Source	Lift Source 1	Blended Body_01
Thrust Source	Thrust Source 1	** Scramjet **_01

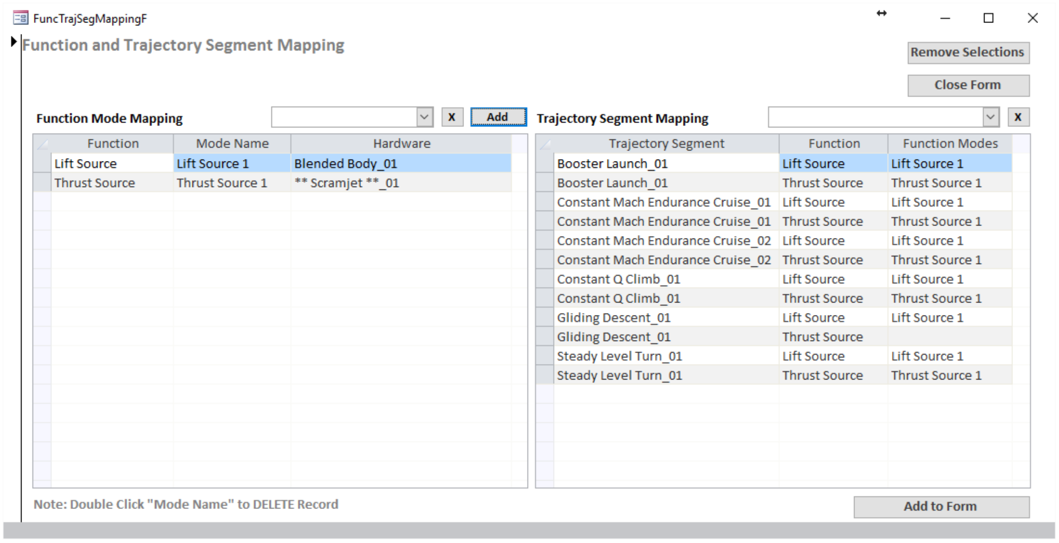
Trajectory Segment Mapping

Trajectory Segment	Function	Function Modes
Booster Launch_01	Lift Source	Lift Source 1
Booster Launch_01	Thrust Source	Thrust Source 1
Constant Mach Endurance Cruise_01	Lift Source	Lift Source 1
Constant Mach Endurance Cruise_01	Thrust Source	Thrust Source 1
Constant Mach Endurance Cruise_02	Lift Source	Lift Source 1
Constant Mach Endurance Cruise_02	Thrust Source	Thrust Source 1
Constant Q Climb_01	Lift Source	Lift Source 1
Constant Q Climb_01	Thrust Source	Thrust Source 1
Gliding Descent_01	Lift Source	Lift Source 1
Gliding Descent_01	Thrust Source	
Steady Level Turn_01	Lift Source	Lift Source 1
Steady Level Turn_01	Thrust Source	Thrust Source 1

Figure 4-6 Product Input Form

The Hardware, Operational Event, and Operational Requirements input mechanism is the Input Tree subform described previously. When either of the “Open” buttons are clicked, the Input Tree subform is opened, and the corresponding hierarchical information is loaded. Once the correct information is input using the Input Tree subform, it is loaded in the Product Form. This process is repeated for each of the three Trees.

In addition to the selection of individual components using the Input Tree subform, it is also necessary describe the dependencies between the components. Two such relationships are defined using the Function Mode and Trajectory Segment Mapping Subform, see Figure 4-7.



Function and Trajectory Segment Mapping

Function Mode Mapping

Function	Mode Name	Hardware
Lift Source	Lift Source 1	Blended Body_01
Thrust Source	Thrust Source 1	** Scramjet **_01

Trajectory Segment Mapping

Trajectory Segment	Function	Function Modes
Booster Launch_01	Lift Source	Lift Source 1
Booster Launch_01	Thrust Source	Thrust Source 1
Constant Mach Endurance Cruise_01	Lift Source	Lift Source 1
Constant Mach Endurance Cruise_01	Thrust Source	Thrust Source 1
Constant Mach Endurance Cruise_02	Lift Source	Lift Source 1
Constant Mach Endurance Cruise_02	Thrust Source	Thrust Source 1
Constant Q Climb_01	Lift Source	Lift Source 1
Constant Q Climb_01	Thrust Source	Thrust Source 1
Gliding Descent_01	Lift Source	Lift Source 1
Gliding Descent_01	Thrust Source	
Steady Level Turn_01	Lift Source	Lift Source 1
Steady Level Turn_01	Thrust Source	Thrust Source 1

Note: Double Click "Mode Name" to DELETE Record

Figure 4-7 Function Mode and Trajectory Segment Mapping Subform

The first dependency mapping, Function Mode, describe groups of hardware from the same functional category that are active at the same time. For example, if there are two separate engines (Rocket and Scramjet) on a vehicle, then there are three possible Function Modes:

- Thrust Source 1 – Rocket Only
- Thrust Source 2 – Scramjet Only
- Thrust Source 3 – Rocket + Scramjet

Each Function Mode describe a different operationally scenario available for the given functional hardware. This classification is necessary as some vehicles require complex operational schedules.

The second dependency defines the relationship between Function Modes and Trajectory Segments. This mapping takes the Function Modes and assigns them to specific Trajectory Segments. Continuing with the Function Modes from the previous example, if a vehicle has a flight profile containing 3 segments (Acceleration, Cruise, and Descent), then an example Function Mode – Trajectory Segment mapping scenario is:

- Acceleration – Thrust Source 1 Or Thrust Source 3
- Cruise – Thrust Source 2
- Descent - None

During the Acceleration segment there are two Function Modes available. This means that either Thrust Source 1 (Rocket Only) or Thrust Source 2 (Rocket and Scramjet) will be active. In this case more information would be needed to decide which of these two choices are active at any given point during the acceleration phase; e.g. Thrust Source 1 when Mach Number < 3, Thrust Source 3 when Mach Number >= 3. The cruise segment only has one Function Mode association; this means that at every point during this segment Thrust Source 2 will be active. During the Descent segment there are no Thrust Source association; there is no thrust producing hardware active for the duration of the segment.

4.2.2 Analysis Process

The Analysis Process building block is comprised of two parts: The System Elements and Disciplinary Elements. A listing of Analysis Process Form Input parameters and an example of the Analysis Process Input Form can be seen in Table 4-4 Table 4-3and Figure 4-8 respectively.

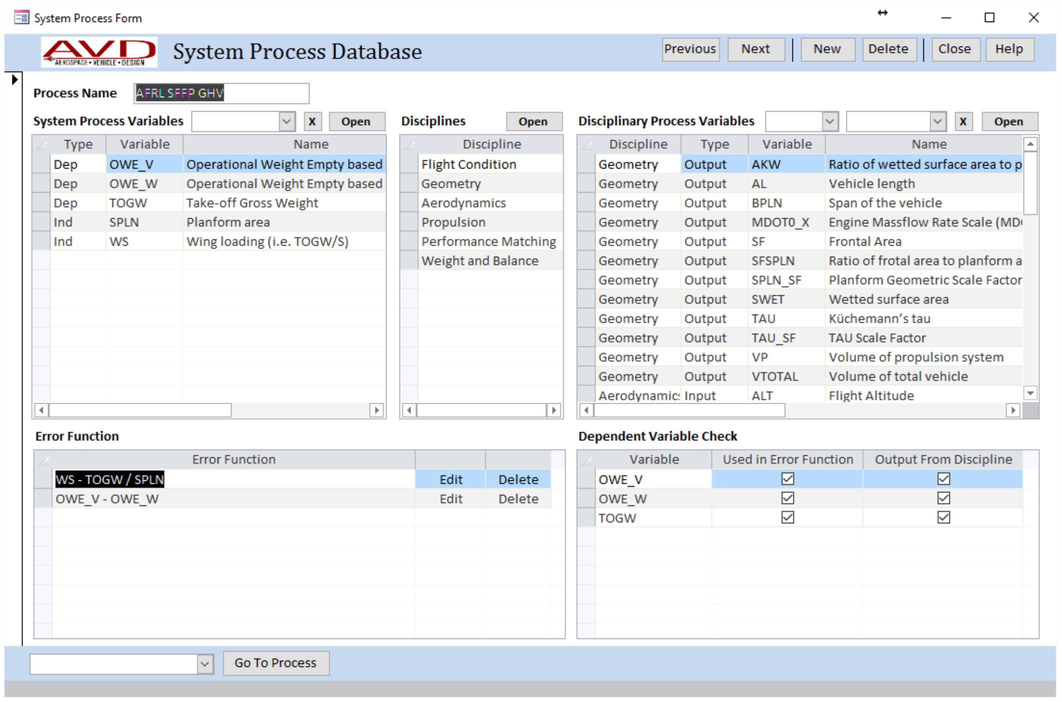
Table 4-4 Analysis Process Form Input Parameters

Input Field	**Type**	**Description**	**Table**
Process Name	String	Name of Analysis Process	SysProcT
System Process Variables	Subform	Listing of Independent and Dependent variables found in system error function	SysProcVarT
Disciplines	Subform	Listing of disciplines, and the order in which they are sequentially run	SysProcDisciplineT
Disciplinary Process Variables	Subform	Listing of output variables required for each discipline. If a discipline is meant to serve as a lookup table (Aerodynamics creating aerodynamic databook) then the lookup table independent variables are specified	SysProcVarT
Error Function	Subform	Listing of objective function for system	SysProcErrFncT

The System Process Variables input mechanism is initiated by clicking the “Open” button and implemented through the use of the Variables subform. Variables needed for all system objective function(s) are selected and classified as either an independent or dependent variable. As stated in Chapter 3, independent variables are the variables that are changed in order to drive the objective function towards the desired goal (zero, maximum, minimum). These variables need initial guesses in order to start the synthesis process. The dependent variables are calculated values that are outputs from the synthesis process.

The Error Function input section allows for the definition of system objective functions. The number of system objective function is a result of the number of independent variables defined in the System Process Variables section. There is a one-to-one ratio of System Independent Variables to System Objective Functions. Clicking on the “Edit” cell brings up the “Error Function” (see Figure 4-9) subform for that objective function. The

Error Function subform allows the user to build up an objective function from System Process Independent and Dependent Variables, and arithmetic operations.


System Process Form

System Process Database

Process Name: AFRL SFFP GHV

System Process Variables

Type	Variable	Name
Dep	OWE_V	Operational Weight Empty based
Dep	OWE_W	Operational Weight Empty based
Dep	TOGW	Take-off Gross Weight
Ind	SPLN	Planform area
Ind	WS	Wing loading (i.e. TOGW/S)

Disciplines

Discipline
Flight Condition
Geometry
Aerodynamics
Propulsion
Performance Matching
Weight and Balance

Disciplinary Process Variables

Discipline	Type	Variable	Name
Geometry	Output	AKW	Ratio of wetted surface area to p
Geometry	Output	AL	Vehicle length
Geometry	Output	BPLN	Span of the vehicle
Geometry	Output	MDOT0_X	Engine Massflow Rate Scale (MD
Geometry	Output	SF	Frontal Area
Geometry	Output	SFSPLN	Ratio of frotal area to planform a
Geometry	Output	SPLN_SF	Planform Geometric Scale Factor
Geometry	Output	SWET	Wetted surface area
Geometry	Output	TAU	Küchemann's tau
Geometry	Output	TAU_SF	TAU Scale Factor
Geometry	Output	VP	Volume of propulsion system
Geometry	Output	VTOTAL	Volume of total vehicle
Aerodynamic	Input	ALT	Flight Altitude

Error Function

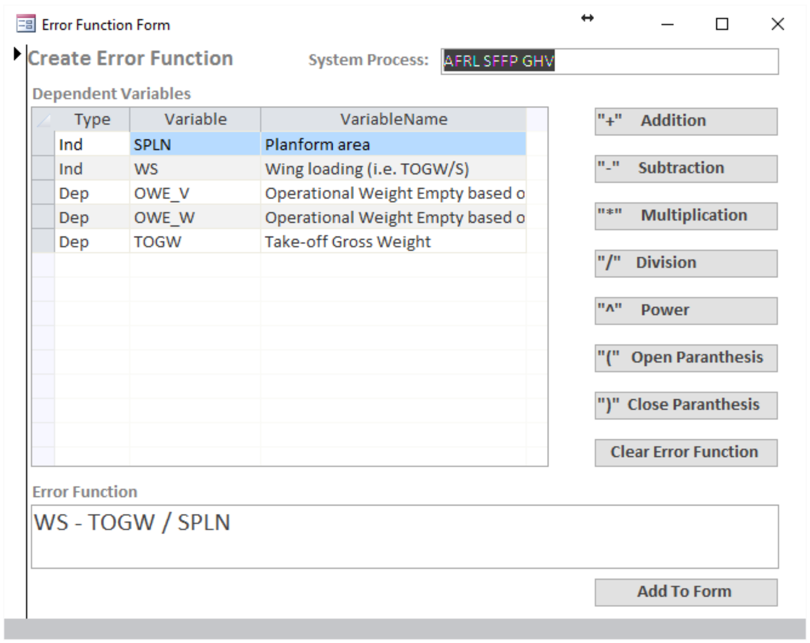
Error Function		
WS - TOGW / SPLN	Edit	Delete
OWE_V - OWE_W	Edit	Delete

Dependent Variable Check

Variable	Used in Error Function	Output From Discipline
OWE_V	☑	☑
OWE_W	☑	☑
TOGW	☑	☑

Figure 4-8 Analysis Process Input Form

The next section of the System Process form is the selections of disciplinary analysis modules. Clicking on “Open” initiates the “Discipline Select” subform, see Figure 4-10. The purpose of this form is the selection of all required disciplinary analysis modules as well the definition of their run order within the system. Clicking on a discipline in the “Master List” subform adds it to the “Selected List” subform. After selecting all required disciplinary modules, clicking the “Add to Form” updates the Disciplines section of the System Process Form.



System Process: AFRL SFFP GHV

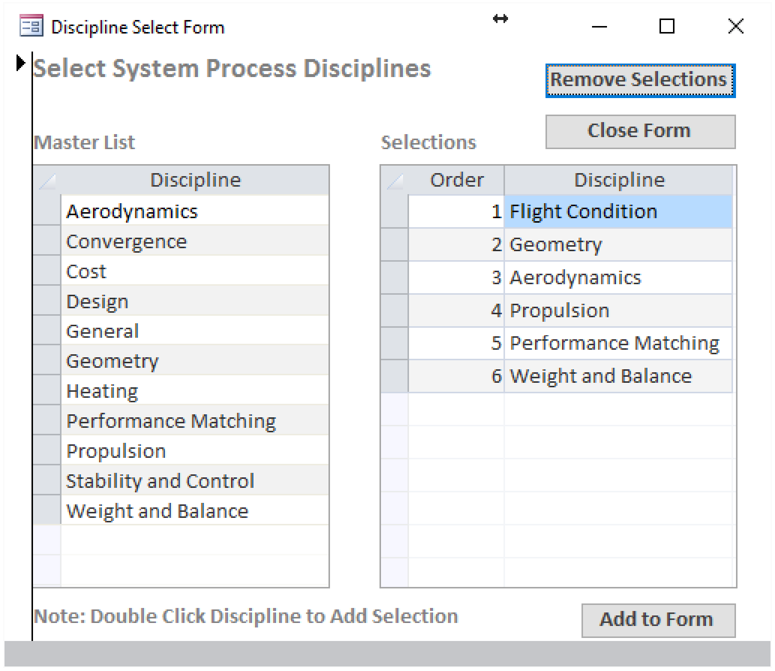
Dependent Variables

Type	Variable	VariableName
Ind	SPLN	Planform area
Ind	WS	Wing loading (i.e. TOGW/S)
Dep	OWE_V	Operational Weight Empty based o
Dep	OWE_W	Operational Weight Empty based o
Dep	TOGW	Take-off Gross Weight

Error Function

WS - TOGW / SPLN

Figure 4-9 Objective Function Subform



Master List

Discipline
Aerodynamics
Convergence
Cost
Design
General
Geometry
Heating
Performance Matching
Propulsion
Stability and Control
Weight and Balance

Selections

Order	Discipline
1	Flight Condition
2	Geometry
3	Aerodynamics
4	Propulsion
5	Performance Matching
6	Weight and Balance

Note: Double Click Discipline to Add Selection

Figure 4-10 Disciplinary Selection and Order Subform

The Disciplinary Process Variables input mechanism is initiated by clicking the "Open" button and implemented through the use of the Variables subform. Disciplinary Process Variables are defined by the Disciplinary Module they are associated with as well as the classification of whether the variable is an output variable or a lookup table input variable.

Classifying a variable as a disciplinary output variable means that any analysis performed for that discipline must calculate that variable. As an example, the first disciplinary process output variable in Figure 4-8 is AKW (ratio of vehicle wetted area to vehicle planform area) and is associated with the Geometry disciplinary module. This definition means that when using this System Process, any Geometry Disciplinary Methods must calculate and output the AKW variable.

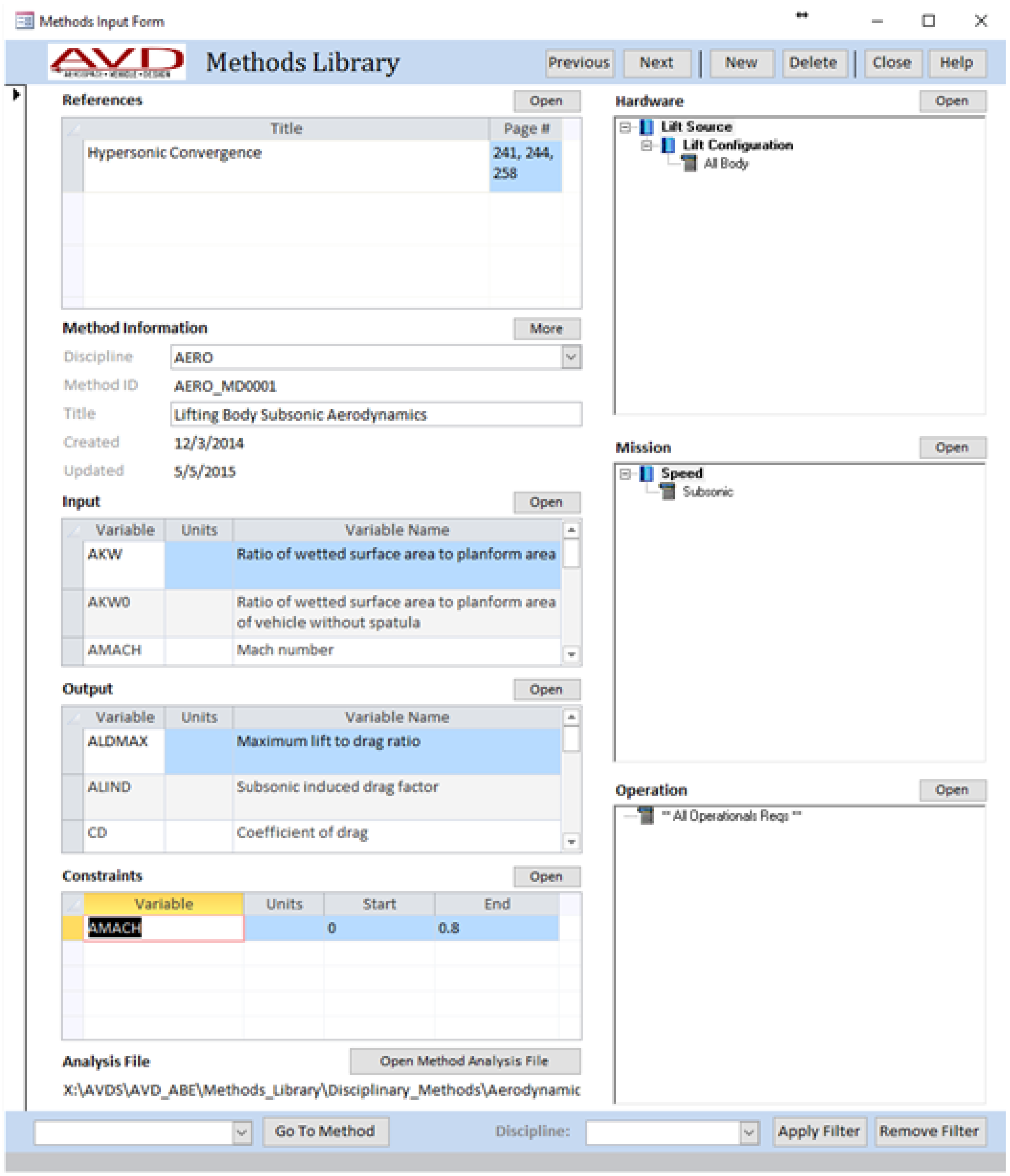
Disciplinary Input variables are defined as variable needed by disciplinary modules whose outputs are not single values, but rather look-up tables. An example would be the Aerodynamics discipline. Assuming aerodynamic performance parameters (C_L, C_D, C_M) are a function of flight condition as well as geometric parameters, means that for a given geometry, the aerodynamic performance will change throughout the design mission. Disciplinary input variables are selected to account for this. Any disciplinary input variable, is a variable that the disciplinary module look-up table is a function of. If Velocity and Altitude are selected as Aerodynamic input variables, then any Aerodynamic performance value (C_L, C_D, C_M) would be a function of those variables. The definition of look-up table input variable sets up the framework that will later be used when writing interpolation functions for each specific disciplinary module.

4.2.3 Disciplinary Method

The Disciplinary Method building block is comprised of three parts: Product Model, Variables and Analysis. A listing of Disciplinary Method Form Input parameters and an example of the Disciplinary Method Input Form can be seen in Table 4-5 and Figure 4-11 respectively.

Table 4-5 Disciplinary Method Form Input Parameters

Input Field	**Type**	**Description**	**Table**
References	Subform	Listing of references describing disciplinary method. There may be several references which comprise a single method.	MethodIndexT
Discipline	String	Discipline associated with method	MethodT
MethodID	Automated	Methods Library identifier. This is auto generated once a new method and discipline have been chosen.	MethodT
Title	String	Name of disciplinary method	MethodT
Created	Automated	Date the method was created in the AVD-DBMS. This is auto generated.	MethodT
Updated	Automated	The last date that changes have been made to the disciplinary method data	MethodT
Input Variables	Subform	Listing of input variables required by the disciplinary method	MethodVarT
Output Variables	Subform	Listing of output variables required by the disciplinary method	MethodVarT
Constraints	Subform	Listing of variable constraints associated with the disciplinary method. The variable name is accompanied by the range of applicability (e.g. 0 < Mach No. < 2).	MethodVarT
Analysis File	MATLAB	Directory location of the MATLAB m-file where the analysis script is located. This is auto generated.	MethodT
Hardware	Tree	Listing of hardware components the disciplinary method is applicable to model	MethodHardwareT
Mission	Tree	Listing of operational events that the disciplinary method is applicable to model	MethodMissionT
Operation	Tree	Listing of operational requirements that the disciplinary method is applicable to model	MethodOperationalReqsT



References

Title	Page #
Hypersonic Convergence	241, 244, 258

Method Information

Discipline	AERO
Method ID	AERO_MD0001
Title	Lifting Body Subsonic Aerodynamics
Created	12/3/2014
Updated	5/5/2015

Input

Variable	Units	Variable Name
AKW		Ratio of wetted surface area to planform area
AKW0		Ratio of wetted surface area to planform area of vehicle without spatula
AMACH		Mach number

Output

Variable	Units	Variable Name
ALDMAX		Maximum lift to drag ratio
ALIND		Subsonic induced drag factor
CD		Coefficient of drag

Constraints

Variable	Units	Start	End
AMACH		0	0.8

Analysis File

X:\AVDS\AVD_ABE\Methods_Library\Disciplinary_Methods\Aerodynamic

Hardware

- Lift Source
  - Lift Configuration
    - All Body

Mission

- Speed
  - Subsonic

Operation

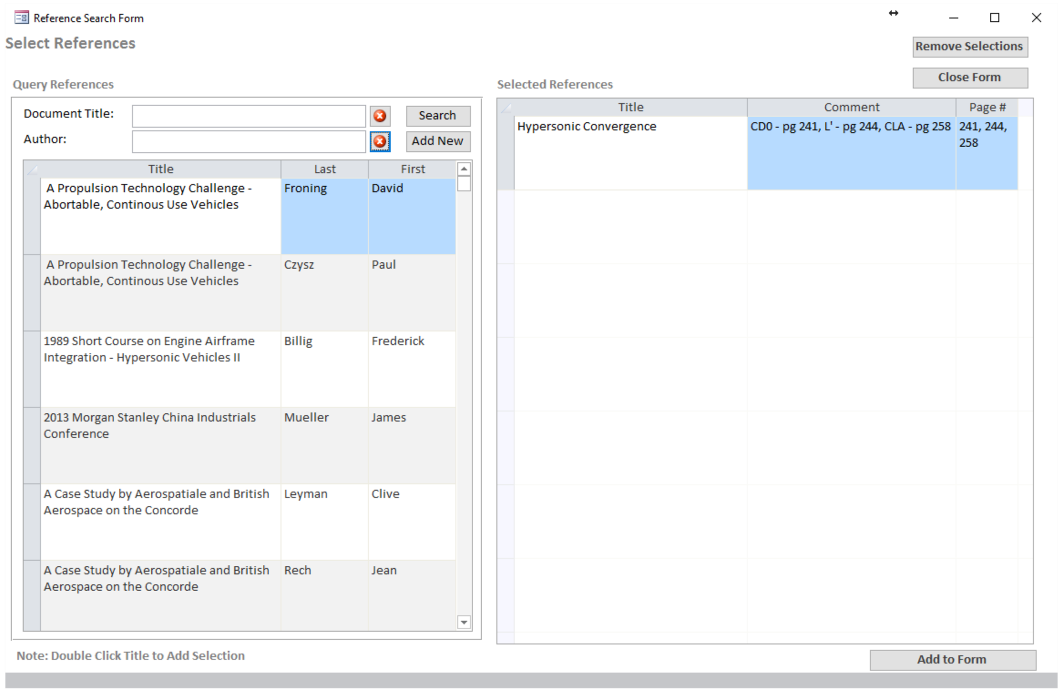
- ** All Operationals Reqs **

Discipline:

Figure 4-11 Disciplinary Method Input Form

In order to add a new Disciplinary Method through the Disciplinary Method input form, a discipline and reference must be chosen. A disciplinary method can have multiple reference associated with it. References are added by clicking the "open" button in the reference section of the disciplinary methods form; this pulls of the Disciplinary Method –

Reference Matching subform, see Figure 4-12. Along with the title of the reference, individual pages in the reference can be associated with a disciplinary method; Figure 4-12 shows the indexing of methods from pages 241, 244 and 248 of one reference (Hypersonic Convergence) and associates them with one method (AERO_MD0001).



Reference Search Form

Select References

Remove Selections

Close Form

Query References

Document Title:

Author:

Search

Add New

Title	Last	First
A Propulsion Technology Challenge - Abortable, Continous Use Vehicles	Froning	David
A Propulsion Technology Challenge - Abortable, Continous Use Vehicles	Czysz	Paul
1989 Short Course on Engine Airframe Integration - Hypersonic Vehicles II	Billig	Frederick
2013 Morgan Stanley China Industrials Conference	Mueller	James
A Case Study by Aerospatiale and British Aerospace on the Concorde	Leyman	Clive
A Case Study by Aerospatiale and British Aerospace on the Concorde	Rech	Jean

Selected References

Title	Comment	Page #
Hypersonic Convergence	CD0 - pg 241, L' - pg 244, CLA - pg 258	241, 244, 258

Note: Double Click Title to Add Selection

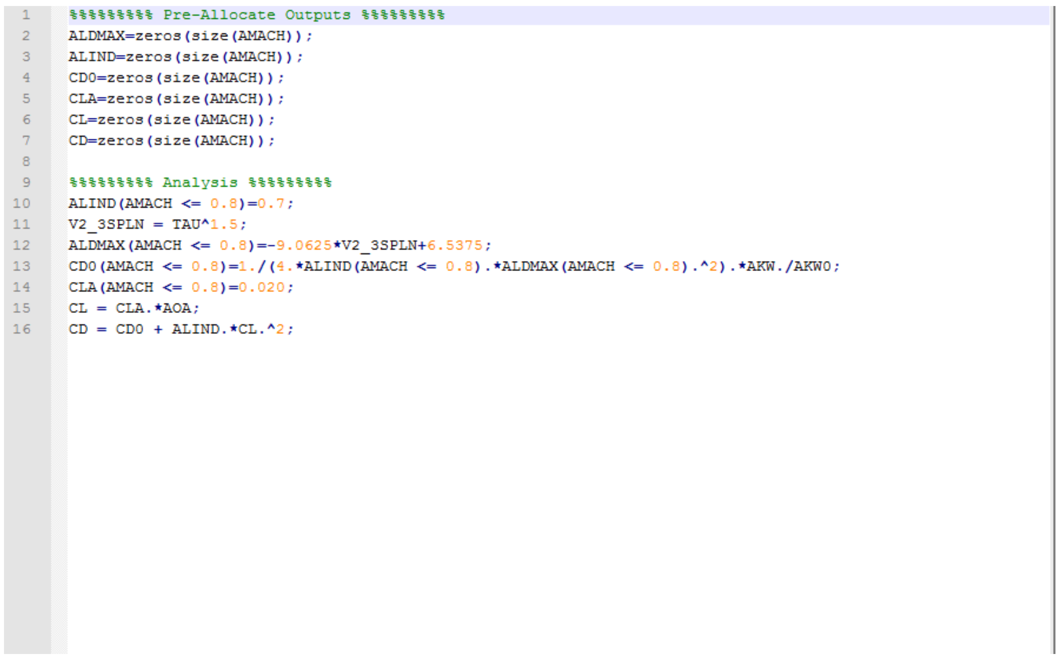
Add to Form

Figure 4-12 Disciplinary Method - Reference Mapping Subform

Method input, output and constraint variables are entered using the variable input subform. While input and outputs variables are selected through the subform with no additional information, method constraint variable require the range of applicability to entered. This is done by directly updating the constraints subform “Start” and “End” cells for each constraint variable selected.

The Hardware, Operational Event, and Operational Requirements input mechanism is the Input Tree subform described previously. When either of the “Open” buttons are clicked, the Input Tree subform is opened, and the corresponding hierarchical information is loaded. Once the correct information is input using the Input Tree subform, it is loaded in the Product Form. This process is repeated for each of the three Trees.

The Disciplinary Method form is meant to provide details of the analysis methods, those details are input into data tables and indexed in order to be used later for various queries. This description encompasses the rationale for the GUI and Database Layers as described in Figure 4-1. The disciplinary methods form also provides the first look into the Analysis layer of the AVD^{DBMS} system. Clocking the “Open Analysis Method File” button open up a text file containing the analysis source code for the given method. If no file exists, then a new blank text file is created and saved according the automatically assigned Method Title. AVD^{DBMS} uses MATLAB as its analysis platform, as so all analysis methods and subroutines are written in MATLAB script, see Figure 4-13.



```
%%%%%%%%% Pre-Allocate Outputs %%%%%%%%%
ALDMAX=zeros(size(AMACH));
ALIND=zeros(size(AMACH));
CD0=zeros(size(AMACH));
CLA=zeros(size(AMACH));
CL=zeros(size(AMACH));
CD=zeros(size(AMACH));

%%%%%%%%% Analysis %%%%%%%%%
ALIND(AMACH <= 0.8)=0.7;
V2_3SPLN = TAU^1.5;
ALDMAX(AMACH <= 0.8)=-9.0625*V2_3SPLN+6.5375;
CD0(AMACH <= 0.8)=1./(4.*ALIND(AMACH <= 0.8).*ALDMAX(AMACH <= 0.8).^2).*AKW./AKW0;
CLA(AMACH <= 0.8)=0.020;
CL = CLA.*AOA;
CD = CD0 + ALIND.*CL.^2;
```

Figure 4-13 Example Methods Library Entry MATLAB m-file (AERO_MD0001.m)

One key to the AVD^{DBMS} implementation is the re-structuring of analysis file data input and output requirements. When writing a new analysis file for a new method, it is not necessary to include the description of any input variables in the analysis file. Any new analysis method is made with the assumption that any input variable that has been selected using the disciplinary method input form exists in the workspace for that file. This means that when writing a new method file it is only necessary to include lines of code dealing with the analysis meant to be performed. In other words, the burden of tracking where input variables have been created or how they are connected in the system is not placed on the user/creator of the method but rather the onus is on the system itself to correctly track and implement these connections.

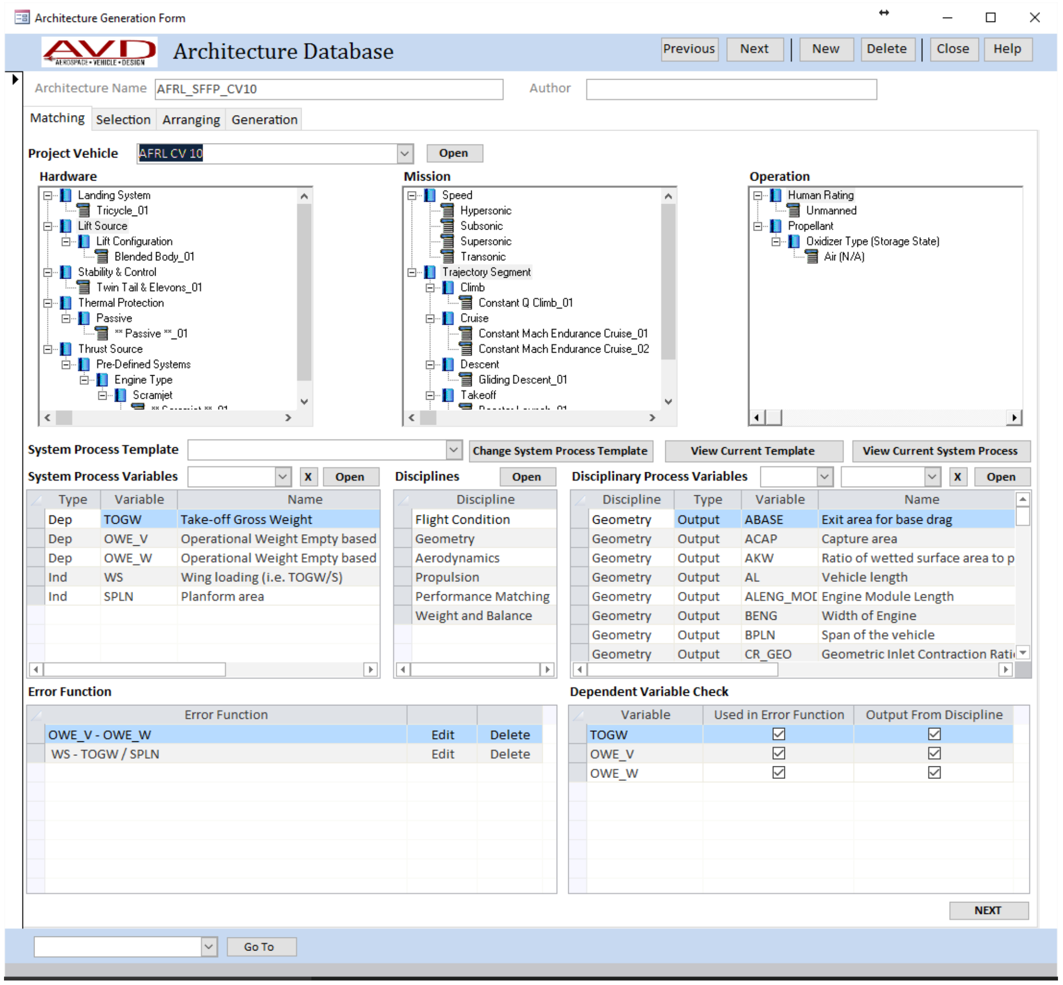
4.3 CMDS Composition Framework

A sequence of four actionable steps for CMDS Composition have been defined in Chapter 3, namely Matching, Selecting, Arranging, and Generation. The impetus for the creation of the CMDS Composition process has been the need to systematically combine groups of CMDS building blocks into a stand-alone CMDS. The CMDS Composition Framework leverages the implementation of a mechanism to input the building blocks into the DBMS. All four of the CMDS Composition steps are contained in CMDS Composition Input Form where each step in represented by individual tabs. The following section describe the input mechanisms and implementation of the CMDS Composition Input Form as well as the structure of the MATLAB CMDS output.

4.3.1 Matching

The purpose of the Matching phase of the CMDS Composition process is to find all disciplinary methods that match the given problem, in other words all disciplinary methods that are compatible with a selected Product and Analysis Process. To this end, the first step in the Matching phase is the selection of both the Product and Analysis

Process to be matched. The Matching tab of the CMDS Composition form has a drop down menu for the selection of both a Product and an Analysis Process, see Figure 4-14.



Architecture Name: AFRL_SFFP_CV10

Author:

Project Vehicle: AFRL CV 10

Hardware

- Landing System
  - Tricycle_01
- Lift Source
  - Lift Configuration
    - Blended Body_01
- Stability & Control
  - Twin Tail & Elevons_01
- Thermal Protection
  - Passive
    - ** Passive **_01
- Thrust Source
  - Pre-Defined Systems
    - Engine Type
      - Scramjet

Mission

- Speed
  - Hypersonic
  - Subsonic
  - Supersonic
  - Transonic
- Trajectory Segment
  - Climb
    - Constant Q Climb_01
  - Cruise
    - Constant Mach Endurance Cruise_01
    - Constant Mach Endurance Cruise_02
  - Descent
    - Gliding Descent_01
  - Takeoff

Operation

- Human Rating
  - Unmanned
- Propellant
  - Oxidizer Type (Storage State)
    - Air (N/A)

System Process Template:

System Process Variables

Type	Variable	Name
Dep	TOGW	Take-off Gross Weight
Dep	OWE_V	Operational Weight Empty based
Dep	OWE_W	Operational Weight Empty based
Ind	WS	Wing loading (i.e. TOGW/S)
Ind	SPLN	Planform area

Disciplines

Discipline
Flight Condition
Geometry
Aerodynamics
Propulsion
Performance Matching
Weight and Balance

Disciplinary Process Variables

Discipline	Type	Variable	Name
Geometry	Output	ABASE	Exit area for base drag
Geometry	Output	ACAP	Capture area
Geometry	Output	AKW	Ratio of wetted surface area to p
Geometry	Output	AL	Vehicle length
Geometry	Output	ALENG_MOD	Engine Module Length
Geometry	Output	BENG	Width of Engine
Geometry	Output	BPLN	Span of the vehicle
Geometry	Output	CR_GEO	Geometric Inlet Contraction Rati

Error Function

Error Function		
OWE_V - OWE_W	Edit	Delete
WS - TOGW / SPLN	Edit	Delete

Dependent Variable Check

Variable	Used in Error Function	Output From Discipline
TOGW	✓	✓
OWE_V	✓	✓
OWE_W	✓	✓

Figure 4-14 CMDS Composition Input Form - Matching

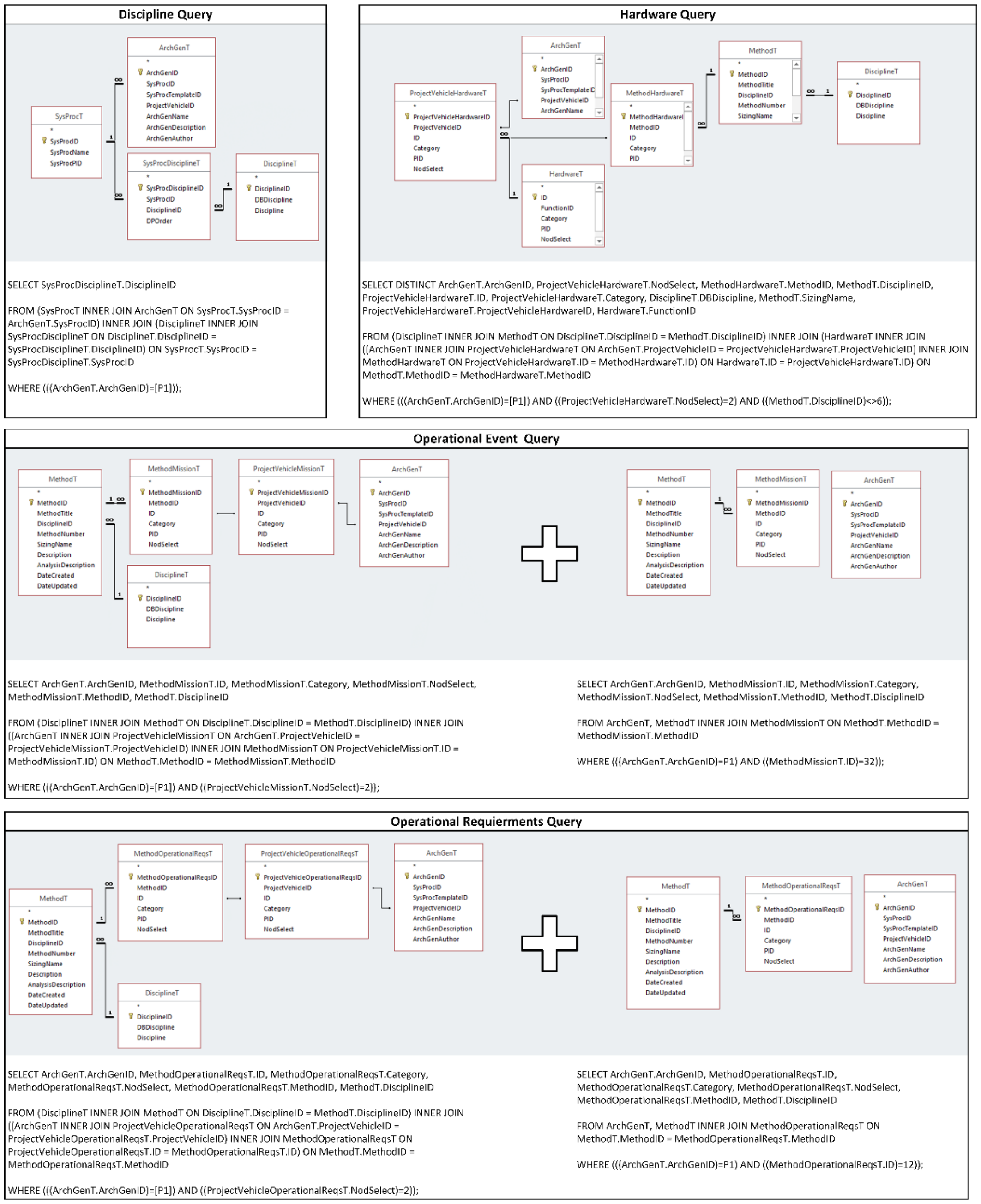
Selecting a Product using the dropdown menu updates the Hardware, Operational Event and Operational Requirements tree diagrams on the form. Clicking on the “Open” button opens the Product form for the selected entry. A drop drown menu is also used for the selection of an Analysis Process where the selection process is nearly identical to that of the Product. The difference comes in the ability to alter the Analysis Process to account for specific problem requirements of the CMDS being created. Specifically, once an

Analysis Process is selected the System and Disciplinary input and output variables can be added. This is the only adjustment that can be done to the Analysis Process and has been implemented in order to facilitate the reuse of Analysis Processes when only minor additions are needed to a saved entry. An example of this would be the addition of disciplinary look-up table variables in order to account for a known problem requirement. If it is required for the aerodynamics module to be a function of altitude, velocity and angle of attack but the selected Analysis Process is only a function of altitude and velocity, angle of attack can be added without the requirement of creating a new Analysis Process.

Once all required input have been selected and/or adjusted, clicking on the “Next” button performs several queries and creates data tables necessary for the next CMDS Composition step (Selecting). There are four queries that work in concert to assess the compatibility of disciplinary methods with the following categories: Discipline, Hardware, Operational Event, and Operational Requirement, see Figure 4-15. The queries work in a sequential nature, meaning that the results from the first query are used as inputs for the second and so on until the data set is output from the last query.

The Disciplines query returns all disciplinary methods the exist in the database that are classified under disciplines matching those found in the selected Analysis Process. This yields a matched listing of disciplinary methods, this set of methods is then used as input for the hardware query.

The hardware query takes each disciplinary method from the matched list and assesses whether they are applicable to any of the hardware components found in the selected Product. The methods that do not match are dropped from the matched data set and those that do match are continue to the Operational Events query.



Discipline Query

```
SELECT SysProcDisciplineT.DisciplineID

FROM (SysProcT INNER JOIN ArchGenT ON SysProcT.SysProcID =
ArchGenT.SysProcID) INNER JOIN (DisciplineT INNER JOIN
SysProcDisciplineT ON DisciplineT.DisciplineID =
SysProcDisciplineT.DisciplineID) ON SysProcT.SysProcID =
SysProcDisciplineT.SysProcID

WHERE (((ArchGenT.ArchGenID)=[P1]));
```

Hardware Query

```
SELECT DISTINCT ArchGenT.ArchGenID, ProjectVehicleHardwareT.NodSelect, MethodHardwareT.MethodID, MethodT.DisciplineID,
ProjectVehicleHardwareT.ID, ProjectVehicleHardwareT.Category, DisciplineT.DBDiscipline, MethodT.SizingName,
ProjectVehicleHardwareT.ProjectVehicleHardwareID, HardwareT.FunctionID

FROM (DisciplineT INNER JOIN MethodT ON DisciplineT.DisciplineID = MethodT.DisciplineID) INNER JOIN (HardwareT INNER JOIN
((ArchGenT INNER JOIN ProjectVehicleHardwareT ON ArchGenT.ProjectVehicleID = ProjectVehicleHardwareT.ProjectVehicleID) INNER JOIN
MethodHardwareT ON ProjectVehicleHardwareT.ID = MethodHardwareT.ID) ON HardwareT.ID = ProjectVehicleHardwareT.ID) ON
MethodT.MethodID = MethodHardwareT.MethodID

WHERE (((ArchGenT.ArchGenID)=[P1]) AND ((ProjectVehicleHardwareT.NodSelect)=2) AND ((MethodT.DisciplineID)<>6));
```

Operational Event Query

```
SELECT ArchGenT.ArchGenID, MethodMissionT.ID, MethodMissionT.Category, MethodMissionT.NodSelect,
MethodMissionT.MethodID, MethodT.DisciplineID

FROM (DisciplineT INNER JOIN MethodT ON DisciplineT.DisciplineID = MethodT.DisciplineID) INNER JOIN
((ArchGenT INNER JOIN ProjectVehicleMissionT ON ArchGenT.ProjectVehicleID =
ProjectVehicleMissionT.ProjectVehicleID) INNER JOIN MethodMissionT ON ProjectVehicleMissionT.ID =
MethodMissionT.ID) ON MethodT.MethodID = MethodMissionT.MethodID

WHERE (((ArchGenT.ArchGenID)=[P1]) AND ((ProjectVehicleMissionT.NodSelect)=2));
```

```
SELECT ArchGenT.ArchGenID, MethodMissionT.ID, MethodMissionT.Category,
MethodMissionT.NodSelect, MethodMissionT.MethodID, MethodT.DisciplineID

FROM ArchGenT, MethodT INNER JOIN MethodMissionT ON MethodT.MethodID =
MethodMissionT.MethodID

WHERE (((ArchGenT.ArchGenID)=P1) AND ((MethodMissionT.ID)=32));
```

Operational Requierments Query

```
SELECT ArchGenT.ArchGenID, MethodOperationalReqsT.ID, MethodOperationalReqsT.Category,
MethodOperationalReqsT.NodSelect, MethodOperationalReqsT.MethodID, MethodT.DisciplineID

FROM (DisciplineT INNER JOIN MethodT ON DisciplineT.DisciplineID = MethodT.DisciplineID) INNER JOIN
((ArchGenT INNER JOIN ProjectVehicleOperationalReqsT ON ArchGenT.ProjectVehicleID =
ProjectVehicleOperationalReqsT.ProjectVehicleID) INNER JOIN MethodOperationalReqsT ON
ProjectVehicleOperationalReqsT.ID = MethodOperationalReqsT.ID) ON MethodT.MethodID =
MethodOperationalReqsT.MethodID

WHERE (((ArchGenT.ArchGenID)=[P1]) AND ((ProjectVehicleOperationalReqsT.NodSelect)=2));
```

```
SELECT ArchGenT.ArchGenID, MethodOperationalReqsT.ID,
MethodOperationalReqsT.Category, MethodOperationalReqsT.NodSelect,
MethodOperationalReqsT.MethodID, MethodT.DisciplineID

FROM ArchGenT, MethodT INNER JOIN MethodOperationalReqsT ON
MethodT.MethodID = MethodOperationalReqsT.MethodID

WHERE (((ArchGenT.ArchGenID)=P1) AND ((MethodOperationalReqsT.ID)=12));
```

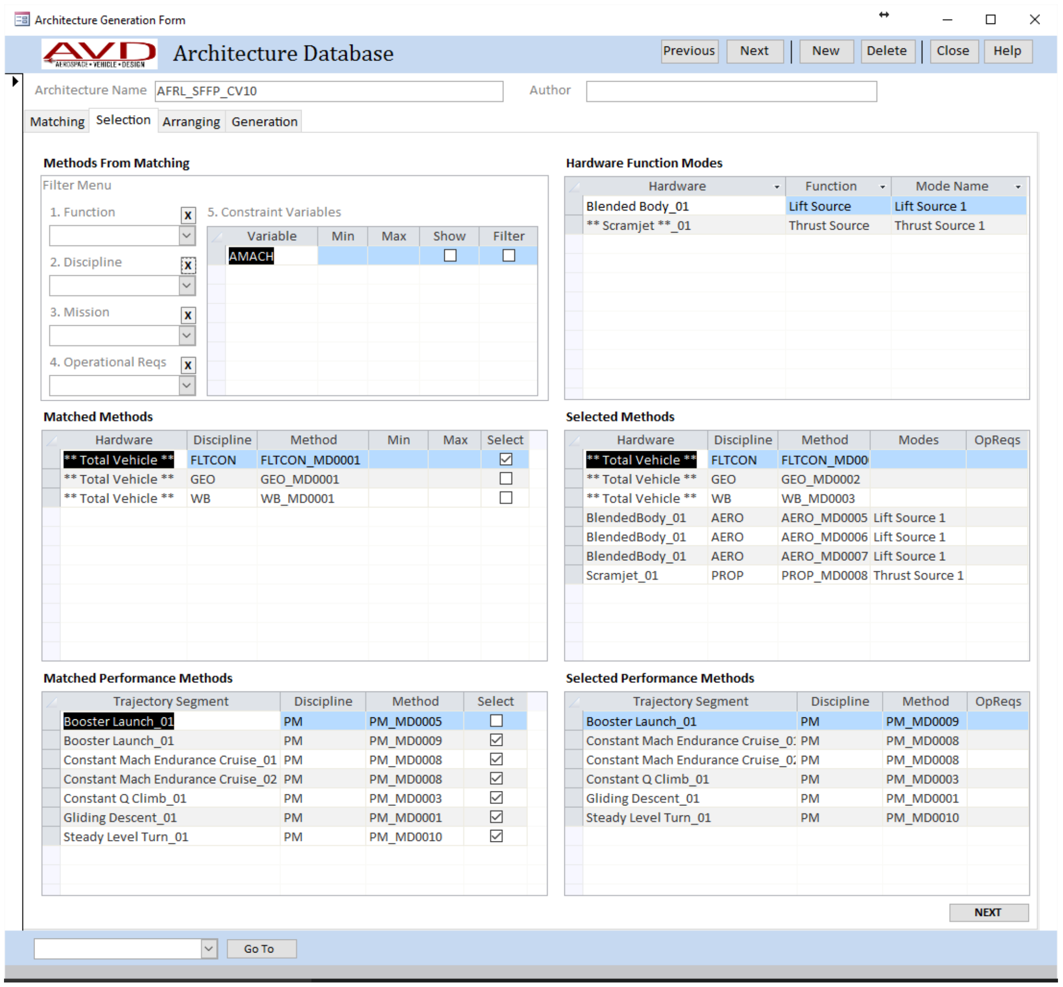
Figure 4-15 CMDS Composition (Matching) – SQL Queries

The Operational Events query takes each disciplinary method from the matched list and assesses whether they are applicable to any of the operational events building blocks found in the selected Product. The methods that do not match are dropped from the matched data set and those that do match are continue to the Operational Requirements query.

The Operational Requirements query takes each disciplinary method from the matched list and assesses whether they are applicable to any of the operational requirements building blocks found in the selected Product. The methods that do not match are dropped from the matched data set and those that do match are used as the master matched data set for use in the Selecting step in the CMDS composition process.

4.3.2 Selecting

The purpose of the Selecting step of the CMDS Composition process is to provide the user with syntactically valid disciplinary method options and provide the capability to select the combination of disciplinary methods that best fit the problem at hand. Although there has been an emphasis on automating much of the CMDS Composition process, it remains necessary for the user to be an integral part of the Selecting step. This is due wholly to the fact that the selection of disciplinary methods might be a function of problem objectives (time, cost and uncertainty), and/or user preference. As so, there will always be a large number of disciplinary methods combinations that are Syntactically valid to solve the problem, the choice between these methods must come from the application of Semantic criterion. It is the intent of the Selecting subform (Figure 4-16) to provide as much information as possible concerning each disciplinary method as well as the ramifications of selecting group of methods so that a CMDS can be tailored to a given problem. The rules and queries present in the CMDS Composition process guarantee Syntactic Composability, the implementation of user inputs for each step in the process provides a mechanism towards ensuring that building blocks that are not Semantically valid are not included. This does not ensure the Semantic Composability of the CMDS, but it removes any Semantically invalid building blocks that can be identified before CMDS Run-Time and subsequent output data mining.



Architecture Name AFRL_SFFP_CV10
Author

Methods From Matching

Variable	Min	Max	Show	Filter
AMACH			☐	☐

Hardware Function Modes

Hardware	Function	Mode Name
Blended Body_01	Lift Source	Lift Source 1
** Scramjet **_01	Thrust Source	Thrust Source 1

Matched Methods

Hardware	Discipline	Method	Min	Max	Select
** Total Vehicle **	FLTCON	FLTCON_MD0001			☑
** Total Vehicle **	GEO	GEO_MD0001			☐
** Total Vehicle **	WB	WB_MD0001			☐

Selected Methods

Hardware	Discipline	Method	Modes	OpReqs
** Total Vehicle **	FLTCON	FLTCON_MD00		
** Total Vehicle **	GEO	GEO_MD0002		
** Total Vehicle **	WB	WB_MD0003		
BlendedBody_01	AERO	AERO_MD0005	Lift Source 1	
BlendedBody_01	AERO	AERO_MD0006	Lift Source 1	
BlendedBody_01	AERO	AERO_MD0007	Lift Source 1	
Scramjet_01	PROP	PROP_MD0008	Thrust Source 1	

Matched Performance Methods

Trajectory Segment	Discipline	Method	Select
Booster Launch_01	PM	PM_MD0005	☐
Booster Launch_01	PM	PM_MD0009	☑
Constant Mach Endurance Cruise_01	PM	PM_MD0008	☑
Constant Mach Endurance Cruise_02	PM	PM_MD0008	☑
Constant Q Climb_01	PM	PM_MD0003	☑
Gliding Descent_01	PM	PM_MD0001	☑
Steady Level Turn_01	PM	PM_MD0010	☑

Selected Performance Methods

Trajectory Segment	Discipline	Method	OpReqs
Booster Launch_01	PM	PM_MD0009	
Constant Mach Endurance Cruise_0	PM	PM_MD0008	
Constant Mach Endurance Cruise_0	PM	PM_MD0008	
Constant Q Climb_01	PM	PM_MD0003	
Gliding Descent_01	PM	PM_MD0001	
Steady Level Turn_01	PM	PM_MD0010	

Figure 4-16 CMDS Composition Input Form - Selecting

The “Matched Methods” and “Matched performance Methods” section of the Selecting tab show the tabulated results of the Matching tab queries. The Top left section of the Selecting tab provides the capability to filter the Matched data based on the following categories: Hardware Function, Discipline, Operational Event, Operational Requirement, and Constraint Variables. Double clicking on the hardware or trajectory segment cells in the Matched Methods tables add the method for that row to the “Selected Methods” table.

The “Selected Method” tables show a listing of all selected disciplinary methods. Additionally, the “Modes” column in the “Selected Methods” table allows for the mapping of disciplinary methods, hardware and function modes.

Once all disciplinary methods have been selected for a given CMDS the “Next” button can be clicked. This action initiates a series of SQL queries meant to add or update data entries in eleven back end data table, see Figure 4-17. Up to this point all actions made in the Selecting tab have adjusted temporary tables, clicking on the “Next” button takes those actions and implements the permanent changes to the database file. There are two classes of tables associated with a CMDS: Data Tables and CMDS Mapping Tables. Data Tables store data meant to associate a specific class of information with a specific CMDS. CMDS Mapping Tables store data meant to connect two classes of data for use with a specific CMDS. The addition of new entries or updating of current entries in these eleven tables is the output of the Selecting step of the CMDS Composition process.

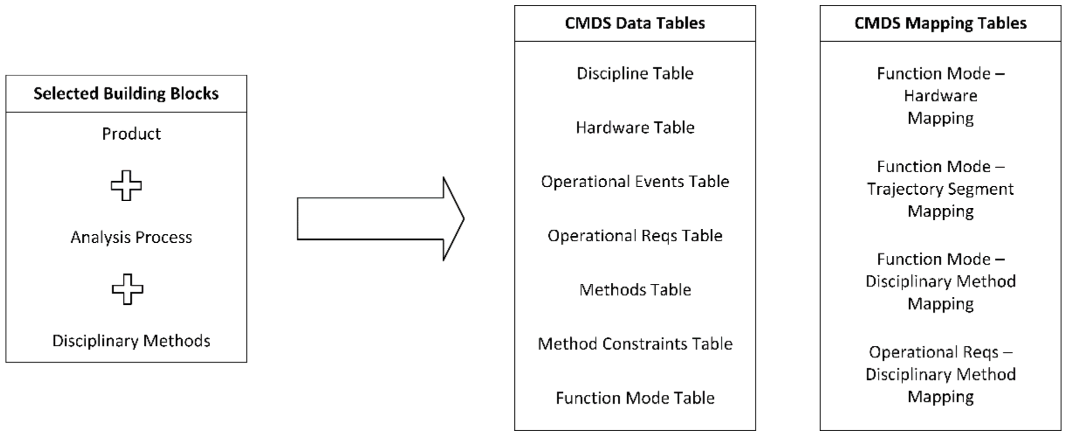
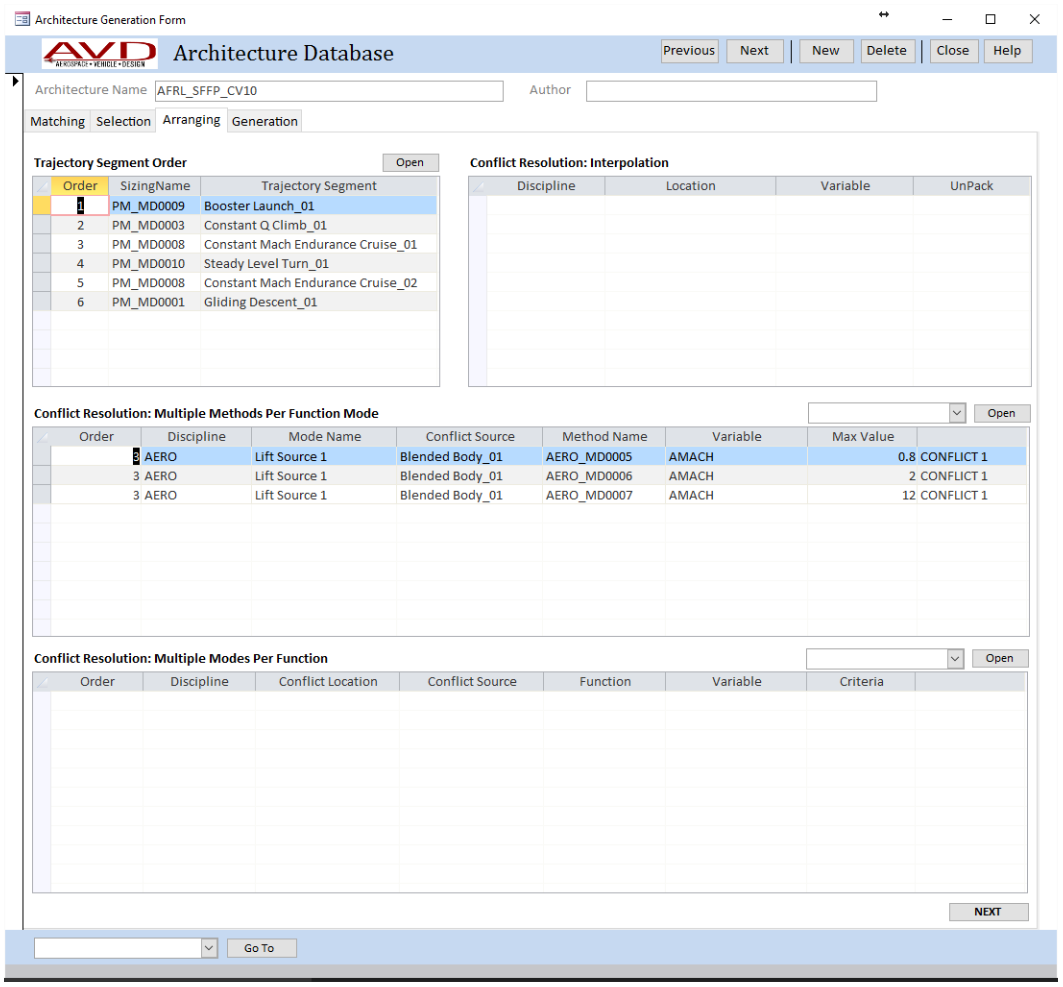


Figure 4-17 CMDS Composition (Selecting) – Generated Tables

4.3.3 Arranging

The Matching step provides the definition of the Product to be analyzed as well as the Analysis Process meant to guide the analysis of said Product. The Selecting step provides a total listing of disciplinary methods that will be used in the CMDS. The Arranging step of the CMDS Composition process ensures the syntactic composability of the output CMDS by taking those building blocks and creating all data interfaces needed to compose them into an analytical framework; the input form for the Arranging step can be seen in Figure 4-18.



Architecture Name: AFRL_SFFP_CV10

Author:

Trajectory Segment Order

Order	SizingName	Trajectory Segment
1	PM_MD0009	Booster Launch_01
2	PM_MD0003	Constant Q Climb_01
3	PM_MD0008	Constant Mach Endurance Cruise_01
4	PM_MD0010	Steady Level Turn_01
5	PM_MD0008	Constant Mach Endurance Cruise_02
6	PM_MD0001	Gliding Descent_01

Conflict Resolution: Interpolation

Discipline	Location	Variable	UnPack

Conflict Resolution: Multiple Methods Per Function Mode

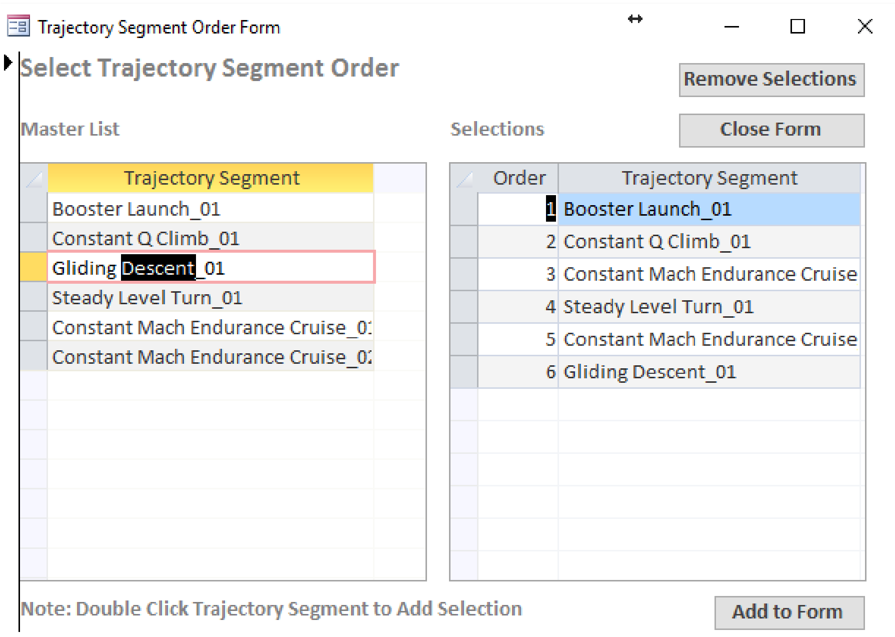
Order	Discipline	Mode Name	Conflict Source	Method Name	Variable	Max Value	
3	AERO	Lift Source 1	Blended Body_01	AERO_MD0005	AMACH	0.8	CONFLICT 1
3	AERO	Lift Source 1	Blended Body_01	AERO_MD0006	AMACH	2	CONFLICT 1
3	AERO	Lift Source 1	Blended Body_01	AERO_MD0007	AMACH	12	CONFLICT 1

Conflict Resolution: Multiple Modes Per Function

Order	Discipline	Conflict Location	Conflict Source	Function	Variable	Criteria

Figure 4-18 CMDS Composition Input Form - Arranging

The subform in the top left corner of Figure 4-18 shows a listing of Trajectory Segment attached to the Product selected in the Matching step. The trajectory segments are initially defined using the Product input form. This form only provides a mechanism for the selection of Trajectory Segments, it does not contain any information concerning the order these segments must be run along the flight profile. The Trajectory Segment run order information is created through the use of the Trajectory Segment Order Input form, see Figure 4-19.


Trajectory Segment Order Form

Select Trajectory Segment Order

Remove Selections

Master List

Selections

Close Form

Trajectory Segment
Booster Launch_01
Constant Q Climb_01
Gliding Descent_01
Steady Level Turn_01
Constant Mach Endurance Cruise_01
Constant Mach Endurance Cruise_02

Order	Trajectory Segment
1	Booster Launch_01
2	Constant Q Climb_01
3	Constant Mach Endurance Cruise
4	Steady Level Turn_01
5	Constant Mach Endurance Cruise
6	Gliding Descent_01

Note: Double Click Trajectory Segment to Add Selection

Add to Form

Figure 4-19 Trajectory Segment Order Input Form

This form is open by clicking the “Open” button next to the Trajectory Segment Order subform. The table on the left side shows all trajectory segments associated with the selected Product. Clicking on a Trajectory Segment adds it to the Selections table on the right hand side. As each Trajectory Segment is added to the Selections table the Run Order information is created. Once all Trajectory Segment and Run Order data is created, clicking on “Add to Form” saves all selections and return to the Arranging tab.

The remaining subforms and inputs fields in the Arranging tab deal with CMDS Conflict Resolution; where each Conflict represents an additional piece of input information required in order to create CMDS data interfaces, see Table 4-6 for a listing of conflicts.

Table 4-6 CMDS Composition - Conflict Resolution

Conflict Name	Reason for Conflict	Resolution
Interpolation		
Multiple Methods per Function Mode	For a single discipline, multiple disciplinary methods have been selected for the same function mode	Add variable range of applicability per disciplinary method
Multiple Function Modes per Function	For a single discipline, multiple function modes have been selected for the same function	Add objective function for function mode selection

The first conflict in Table 4-6 refers to the case where a lookup table output variable is required as an input at any point in the CMDS. An example of this would be the need for Propulsion performance data (Thrust, Isp, etc.) in a Performance Matching disciplinary method (Constant Q Climb). If the Propulsion discipline has been defined as a lookup table discipline in the Analysis Process, then any output data will be saved in lookup table form. Any disciplinary method requiring the use of these variable must then interpolate the output data to get performance data for a specific condition. This conflict helps to ensure that all of the input variables needed to interpolate the data are available for use in the method or function needed to interpolate performance data.

There are two cases that can occur for each required input variable: defined through inputs, or defined through in-line calculation. In the first case, the variables are input into the method or function through the variable inputs at the top of the analysis file. These variable can come from the input file or any analysis methods that has been run previously in the CMDS. The second case involves the definition of the input variables in the source code of the method or function itself. An example of this case can be seen in a Performance Matching method. A Constant Q Acceleration method starts at a given point (Altitude, and Velocity, and Time) and integrates forward along the trajectory. As the vehicle moves along the trajectory new values of Altitude, Velocity and Time are calculated in the method itself (In-Line). If the propulsion data is a function of both Altitude and

Velocity, then the interpolation function would need the new value of Altitude and Velocity at each step.

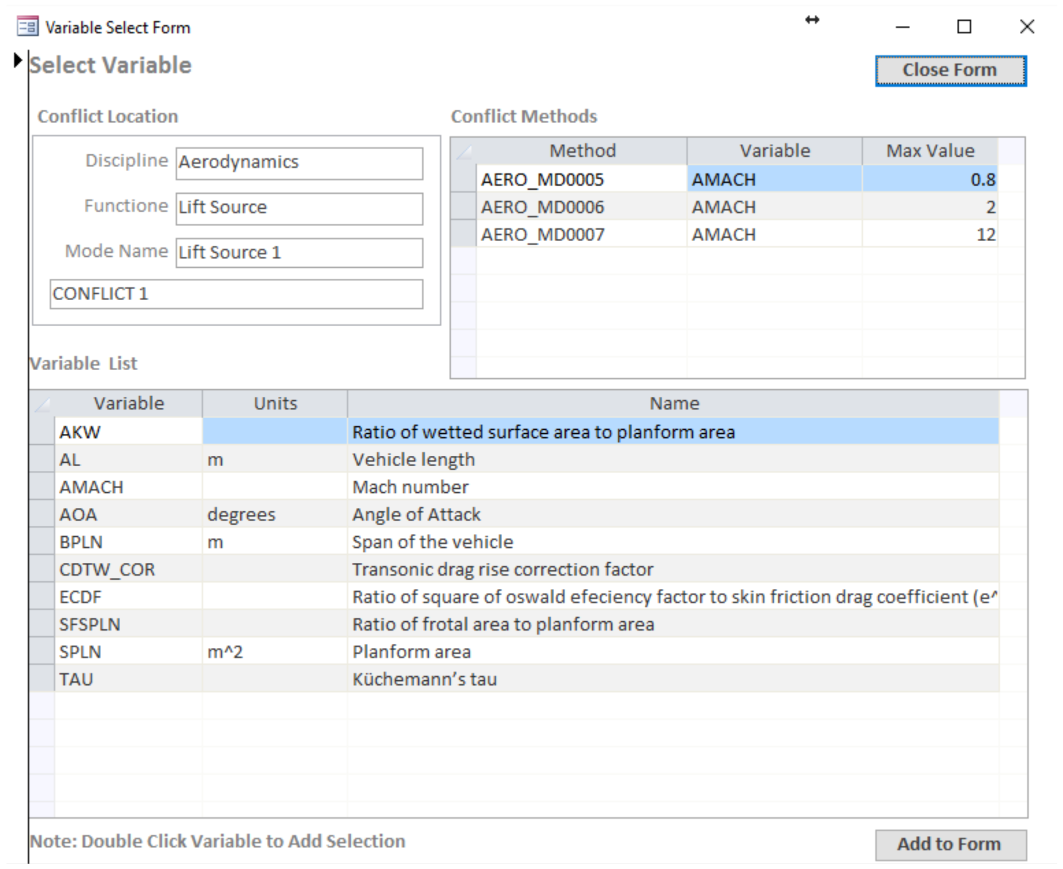
The Interpolation conflict table in the Arranging tab, provides a list of every instance that an interpolation variable is needed, and one or more of its input variable is not defined through in-line calculation in the method/function where the interpolation takes place and has not been defined in any previously run analysis methods in the CMDS. In this case, the interpolation input variable would have to come from the input file. This case is seen as a slight outlier to use constant input data when using interpolated performance data, and the listing of these instances serves as a visual cue for the user.

The second conflict in Table 4-6 refers to the case where multiple disciplinary methods have been selected for a given function mode. An example of this can be seen in Figure 4-18 where three Aerodynamic methods have been selected to model the Lift Source 1 Function Mode, see Table 4-7. This Conflict is resolved using the Multiple Methods per Mode form seen in Figure 4-20.

Table 4-7 Example Conflict Resolution - Multiple Methods per Mode

Applicability Variable Range	**Method to Run**
$0 \leq AMACH < 0.8$	AERO_MD0005
$0.8 \leq AMACH < 2.0$	AERO_MD0006
$2.0 \leq AMACH < 12$	AERO_MD0007

In order to create data connections for these methods, it is necessary to know when each method should be applied. This extra information comes in the form of an applicability variable. The applicability variable is an input variable that is common between all disciplinary methods found in a given conflict; the variable list table in Figure 4-20 shows all common input variables for aerodynamic methods AERO_MD0005, AERO_MD0006, and AERO_MD0007.



Variable Select Form

Select Variable

Close Form

Conflict Location

Discipline	Aerodynamics
Functione	Lift Source
Mode Name	Lift Source 1
	CONFLICT 1

Conflict Methods

Method	Variable	Max Value
AERO_MD0005	AMACH	0.8
AERO_MD0006	AMACH	2
AERO_MD0007	AMACH	12

Variable List

Variable	Units	Name
AKW		Ratio of wetted surface area to planform area
AL	m	Vehicle length
AMACH		Mach number
AOA	degrees	Angle of Attack
BPLN	m	Span of the vehicle
CDTW_COR		Transonic drag rise correction factor
ECDF		Ratio of square of oswald efeciency factor to skin friction drag coefficient (e^
SFSPLN		Ratio of frotal area to planform area
SPLN	m^2	Planform area
TAU		Küchemann's tau

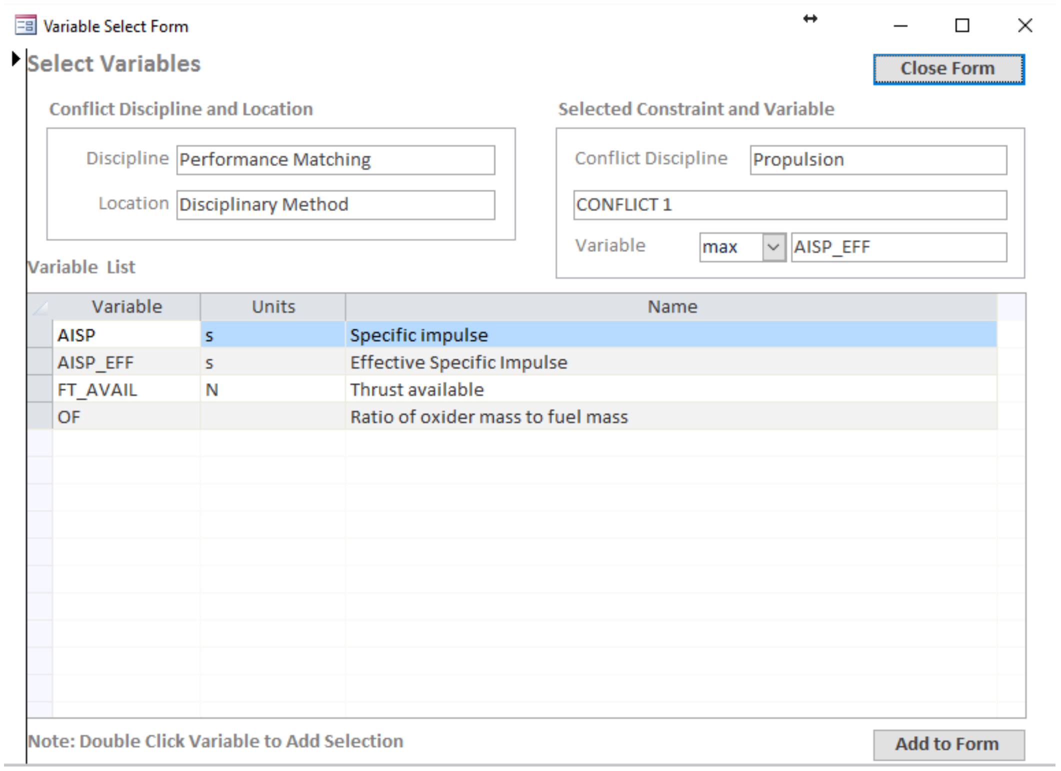
Note: Double Click Variable to Add Selection

Add to Form

Figure 4-20 Conflict Resolution Form - Multiple Methods per Mode

Clicking on a Variable in the Variable List Table adds it to the Conflict Methods Table. Once a Variable has been selected, it is necessary to attach a maximum value of the applicability variable for each disciplinary method. This creates a range of applicability and provides a guideline on when each of these methods will be used during CMDS run-time.

The third conflict in Table 4-6 refers to the case where multiple function modes exist for a single function; e.g. a vehicle that contains multiple propulsion function modes (Rocket, Scramjet, Rocket + Scramjet). In this case, if more than one of the propulsion function modes have been assigned to a specific disciplinary method (Rocket and Rocket + Scramjet are selected for the Constant Q Climb Trajectory Segment) then additional information is needed in order to select which function mode performance data will be used. This conflict is resolved through the use of the Multiple Modes per Function form (Figure 4-21).



Conflict Discipline and Location

Discipline	Performance Matching
Location	Disciplinary Method

Selected Constraint and Variable

Conflict Discipline	Propulsion
	CONFLICT 1
Variable	max AISP_EFF

Variable List

Variable	Units	Name
AISP	s	Specific impulse
AISP_EFF	s	Effective Specific Impulse
FT_AVAIL	N	Thrust available
OF		Ratio of oxider mass to fuel mass

Note: Double Click Variable to Add Selection

Figure 4-21 Conflict Resolution Form - Multiple Modes per Function

The variable list table shows all disciplinary output variables defined for the given discipline in the Analysis Process. Clicking on a variable in the Variable List adds that

Variable to the Selected Constraint Variable table. The selected Variable becomes the Constraint variable used in the selection objective function. Next it is necessary to select whether the objective function will be based on a maximum or minimum value of the constrain variable selected. The example in Figure 4-21 shows the selection of the Constraint Variable as AISP_EFF, and the objective function direction as max. In this case, the selection of which Thrust Source to use during the Constant Q Acceleration Trajectory Segment depends on the value of AISP_EFF for each mode at each point along the trajectory. With both of these selections made, clicking on “Add to Form” saves all inputs and updates data in the Arranging tab.

Once all required inputs have been made in the Arranging tab (Figure 4-18), clicking on the “Next” button initiates SQL statements creating six blueprint tables containing all interface data for the CMDS, see Table 4-8. Each of the temporary tables contain interface mapping information, creating a blueprint for CMDS Generation.

Table 4-8 CMDS Arranging – Blueprint Tables

Table Function	Temp Table Name	Description
Disciplinary Process - Output Variable Mapping	DPFileIO	Maps disciplinary methods where disciplinary output variables are created with disciplinary methods where disciplinary output variables are required as input
Disciplinary Process - Input Variable Mapping	DPInterpFileIO	Selected disciplinary methods where in-line interpolation is required
Disciplinary Process - Function Mode Mapping	DPFuncFileIO	Maps disciplinary output variables with function modes
Disciplinary Method - Input/Output Variable Mapping	MethodFileIO	Maps the input and output of all disciplinary methods
Disciplinary Method - Function Mode Mapping	MethodFuncFileIO	Maps disciplinary method output variables with function modes
Disciplinary Method - Conflict Mapping	MethodFileIN	Maps conflict resolutions with disciplinary outputs variables

In order to create the CMDS blueprint, the variables being mapped have been classified based on their information type, see Table 4-9.

Table 4-9 CMDS Variable Class Description

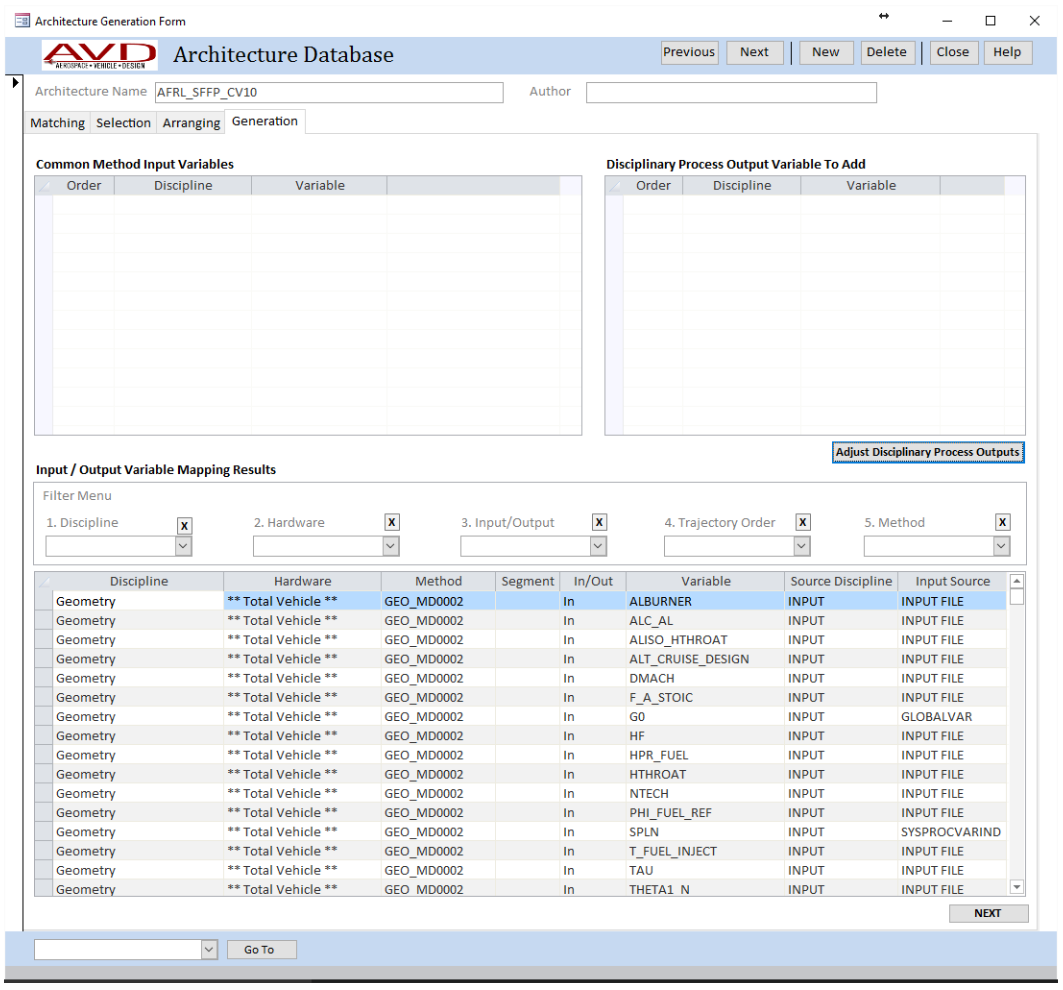
Variable Class	**Variable Structure Location**
Global Variable [GLOBALVAR]	Variable.SYSPROC.INPUT
Flight Condition [FLTCON]	Variable.DISCPROC.FLTCON.OUTPUT
System Process Independent Variable [SYSPROCVARIND]	Variable.SYSPROC.INPUT
System Process Dependent Variable [SYSPROCVAROUT]	Variable.SYSPROC.OUTPUT
Disciplinary Process Input Variable [DISCPROCVARIN]	Variable.DISCPROC.[Discipline].INPUT
Disciplinary Process Output Variable [DISCPROCOUT]	Variable.DISCPROC.[Discipline].OUTPUT
Disciplinary Method Input Variables [METHODIN]	Variable.HW.[Hardware].[Discipline].[Hardware]_[Method Name].INPUT
Disciplinary Method Output Variable [METHODOUT]	Variable.HW.[Hardware].[Discipline].[Hardware]_[Method Name].OUTPUT
Mission Variable [MISSION]	Variable.MISSION.OUTPUT
Trajectory Segment Input Variable [TRAJSEGIN]	Variable.TRAJSEG.[Trajectory Segment]_[Method Name].INPUT
Trajectory Segment Output Variable [TRAJSEG]	Variable.TRAJSEG.[Trajectory Segment]_[Method Name].OUTPUT
Performance Matching Start Variables [PM_START]	Variable.MISSION.INPUT.
Input File Variables [INPUTFILE]	Variable.HW.[Hardware].[Discipline].[Hardware]_[Method Name].INPUT
Look Up Table Variable [LUT_MAP]	Variable.DISCPROC.[Discipline].OUTPUT
Hardware Array [HARDWARE_ARRAY]	Variable.HW.[Hardware].[Discipline].[Hardware]_[Method Name].INPUT
Hardware Variable [HARDWARE]	Variable.HW.[Hardware].[Discipline].OUTPUT
Function Mode Variable [MODENAME]	Variable.FUNCMODE.[Mode Name].[Discipline].OUTPUT

There are fourteen variable classes defined in the Arranging step of the CMDS Composition process. Each class represents a specific type of information, and have its own set of parameters and data location information attached to it. Table 4-9 has a listing of both the Name of the Variable class as well as the Structure Location where these variables will exist in the generated CMDS. The syntax used for the structure location is written in MATLAB script notation, and utilizes the structure array data type.

4.3.4 Generation

The Generation step of the CMDS Composition process combines the Product, Analysis Process and Disciplinary Method selections from the Matching and Selecting steps, and the interface data created in the Arranging step in order to create a stand-alone CMDS. The CMDS is written for use in the MATLAB analysis environment, as so the output source code is comprised of ordered executable MATLAB scripts. Figure 4-22 shows the Generation tab on the CMDS Composition form. The ”Input/Output Variable Mapping Results” table provide a listing every variable mapping result for the CMDS.

The “Common Method Input Variables” table at the top of the Generation tab contains a listing of input variables that are created by a disciplinary method but are not classified as disciplinary process variables, and provides the capability to augment the Analysis Process by adding them as disciplinary process variables. Disciplinary process output variables are the main outputs of a discipline analysis and are the only variables created within the CMDS that can be used as inputs for a different discipline. For example, if vehicle lift and drag are defined as aerodynamic disciplinary process output variables, those variables can be used as inputs to any disciplinary analysis modules (e.g. Performance Matching) run after the aerodynamics module. The variables listed in the “Common Method Input Variables” are not define as disciplinary output variables, this results in the values for those input variables coming from the input file. To ensure that these variable input come from the method/discipline where they are created, as opposed to the input file, the user clicks on the variable in question and clicks on the "Adjust Disciplinary Process Outputs" button.



Architecture Name: AFRL_SFFP_CV10

Author:

Discipline	Hardware	Method	Segment	In/Out	Variable	Source Discipline	Input Source
Geometry	** Total Vehicle **	GEO_MD0002		In	ALBURNER	INPUT	INPUT FILE
Geometry	** Total Vehicle **	GEO_MD0002		In	ALC_AL	INPUT	INPUT FILE
Geometry	** Total Vehicle **	GEO_MD0002		In	ALISO_HTHROAT	INPUT	INPUT FILE
Geometry	** Total Vehicle **	GEO_MD0002		In	ALT_CRUISE_DESIGN	INPUT	INPUT FILE
Geometry	** Total Vehicle **	GEO_MD0002		In	DMACH	INPUT	INPUT FILE
Geometry	** Total Vehicle **	GEO_MD0002		In	F_A_STOIC	INPUT	INPUT FILE
Geometry	** Total Vehicle **	GEO_MD0002		In	G0	INPUT	GLOBALVAR
Geometry	** Total Vehicle **	GEO_MD0002		In	HF	INPUT	INPUT FILE
Geometry	** Total Vehicle **	GEO_MD0002		In	HPR_FUEL	INPUT	INPUT FILE
Geometry	** Total Vehicle **	GEO_MD0002		In	HTHROAT	INPUT	INPUT FILE
Geometry	** Total Vehicle **	GEO_MD0002		In	NTECH	INPUT	INPUT FILE
Geometry	** Total Vehicle **	GEO_MD0002		In	PHI_FUEL_REF	INPUT	INPUT FILE
Geometry	** Total Vehicle **	GEO_MD0002		In	SPLN	INPUT	SYSPROCVARIND
Geometry	** Total Vehicle **	GEO_MD0002		In	T_FUEL_INJECT	INPUT	INPUT FILE
Geometry	** Total Vehicle **	GEO_MD0002		In	TAU	INPUT	INPUT FILE
Geometry	** Total Vehicle **	GEO_MD0002		In	THETA1_N	INPUT	INPUT FILE

Figure 4-22 CMDS Composition Input Form - Generation

Clicking the "Next" button completes the final step in the CMDS Composition process and initializes a combination of SQL and VBA function in order to create a CMDS written in MATLAB script. The CMDS is a combination of the input selected at each step of the CMDS Composition process as well as the variable mappings that have been created as a function of those choices. The output CMDS is comprised of five file types:

Input File, Driver File, Convergence File, Disciplinary Process Files, Disciplinary Method Files and Utility Files, see Table 4-10.

Table 4-10 CMDS Output - MATLAB File Types

File Type	**Description**
Input	Provides mechanism to input values for system level, disciplinary process level and disciplinary method level input variables.
Driver	Serves as CMDS executable file. Controls the type of analysis to be run (Single Point or Multi-point), as well as the assignment of parallel processing workers
Convergence	Runs each discipline in the order specified by the Analysis Process. Catalogues and applies the Independent and Dependent variables to construct the System Objective Function.
Disciplinary Process	Control the running of Disciplinary Methods for a given discipline. Constructs Disciplinary Process Output variable and stores them as a function of hardware and function mode. There are three types of Disciplinary Processes: No Look-Up Table, Look-Up Table, and Performance Matching.
Disciplinary Method	Provides disciplinary analysis to create method and disciplinary outputs variables.

Chapter 5

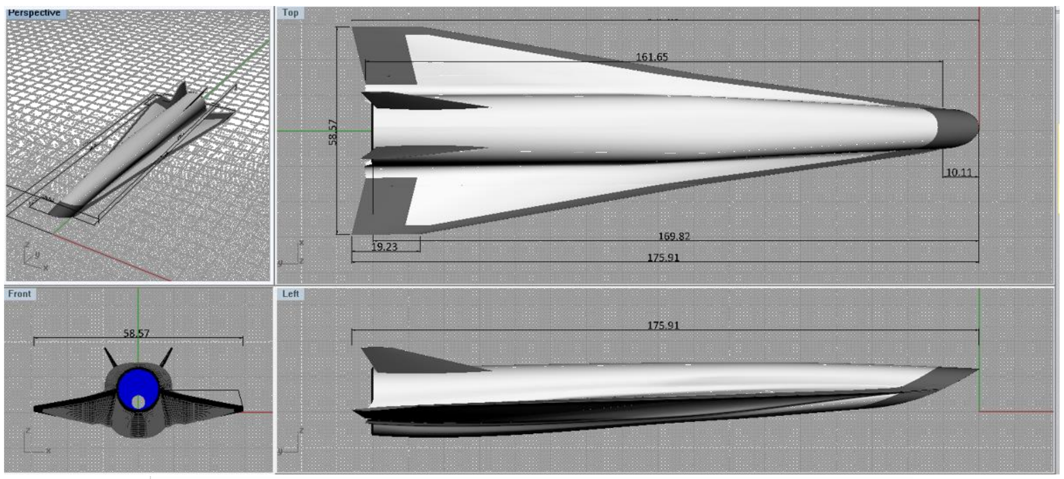
Case Studies

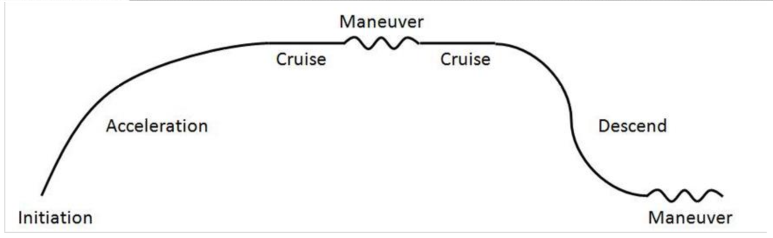
The Air Force Science and Technology Research Plan is recognizing that speed remains an Air Force priority for its warfighting capabilities. A cohesive plan is emerging that may enable operational high-speed weapons and aircraft platforms for a range of intelligence, surveillance, reconnaissance and other missions. This road-map pragmatically defines unmanned and possibly piloted operational systems to be operational between mid-2020 and 2030. Such planning needs to directly address the associated technological difficulties of the tasks and the realities of defense science and technology (S&T) spending in a time of austerity.

As a means to respond to this directive the Air Research Laboratory has seen the need to provide a setting to allow collaboration between aerospace hypersonic research partners working in government, industry or academia. Hypersonic research has for the most part been conducted while adhering to International Trade in Arms Regulations (ITAR) guidelines and industry proprietary technology considerations. To this end the AFRL has implemented the generic hypersonic vehicle (GHV) study. Liston (Ruttle, Stork, and Liston 2012) describes the impetus for and characteristics of the GHV study as follows:

> *“Due to proprietary or ITAR restrictions, AFRL cannot readily provide most data or designs to researchers who are not in the US Government or associated contractor community. It 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.*
>
> *The objective of this study was to create a family of generic hypersonic vehicles (GHV) completely in-house using design tools either owned by or licensed to AFRL. The GHV would have to be based upon the state of the art in hypersonic engine design so that it would be valuable for studies of operability, controllability, and aero-propulsion integration. It was agreed early on that the vehicle would need to have a blended wingbody configuration, 3D inlet and nozzle, an axisymmetric scramjet combustor,*

and a metallic structure with a thermal protection coating. The GHV would cruise at Mach 6 within a dynamic pressure range of 1000 to 2000 psf, and maneuver at a maximum loading factor of approximately 2G."





	Mach	Lift/Weight	Dynamic Pressure (psf)
Initiation (booster separation)	4 – 5	1	2500 – 1500
Acceleration and Climb	4 – 6	>1	2500 – 1500
Cruise	6	1	2000 – 1000
Maneuver (#1 or #2)	~6	~2	2000 – 1000
Cruise	6	1	2000 – 1000
Descend (powered)	6 – 4	<1	2000 – 3000
Descend (unpowered)	4 – 3	<1	2500 – 5000
Maneuver (unpowered)	~3	>1	~5000

Figure 5-1 Generic Hypersonic Vehicle Configuration and Mission Profile (Ruttle, Stork, and Liston 2012)

In an effort to add to the collective GHV knowledge base, the CMDS Composition process will be applied in order to produce a CMDS to model each specified combination of product, analysis process and disciplinary methods. Each CMDS will then be executed in order to create a solution space for each vehicle technology package. The case study will be separated into three tracks: Verification using GHV disciplinary data, Verification using selected disciplinary analysis methods, and Application of the CMDS Composition Process to assess the design solution space for the GHV hardware, see Figure 5-2.

The first validation study will be focused on matching the vehicle geometry and performance results reported by Ruttle et al (Ruttle, Stork, and Liston 2012). When available, the defined Product, Analysis Process, and Disciplinary Methods from the report will be used as input in the CMDS Composition process. Specifically, the aerodynamic, and propulsion disciplinary performance methods will be in the form of interpolated look-up tables, with the data coming directly from Ruttle (Ruttle, Stork, and Liston 2012). The intent of this study is twofold:

- The verification of the syntactic composability of the output CMDS

  Does the CMDS Composition process produce an analytical framework that can produce analytical results?

- The validation of the applicability of the output CMDS to solve the given problem being input

  When using the same input data (GHV aerodynamic and propulsion performance data), does the output analysis framework produce results that are consist with reference values for the given problem (GHV Reference data)?

The second validation study will also focus on matching the vehicle geometry and performance results reported by Ruttle et al (Ruttle, Stork, and Liston 2012). Although in this case, the estimation of aerodynamic performance will come in the form of engineering level analysis methods. The intent of this study is threefold:

- The verification of the syntactic composability of the output CMDS

Does the CMDS Composition process produce an analytical framework that can produce analytical results?

- The validation of the applicability of the output CMDS to solve the given problem being input

  When using the CMDS process to match, select and integrate disciplinary methods, does the output analysis framework produce results that are consist with reference values for the given problem (GHV Reference data)?
- Show the sensitivity of the output analysis framework to Disciplinary Method selection

  How does the output CMDS change when different disciplinary methods are chosen?

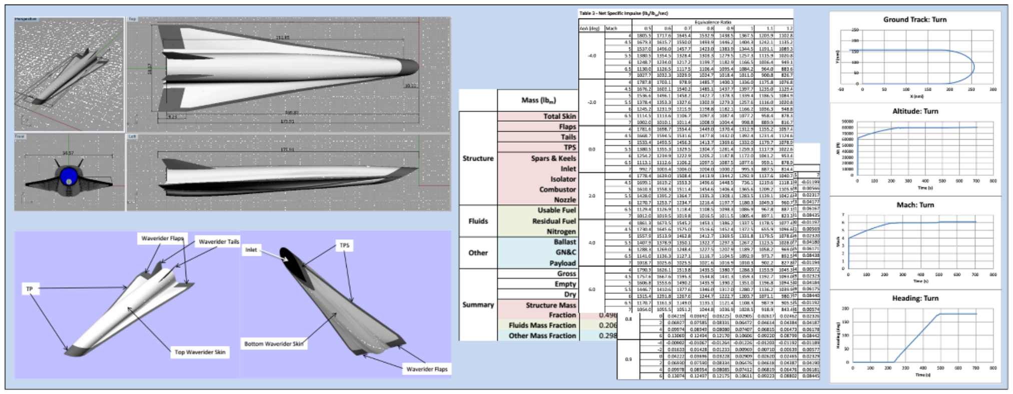
The Application study is an effort to highlight the versatility of the CMDS Composition process to answer a given problem. For this case study the GHV problem description will be altered in order to assess the design solution space of the GHV vehicle. The intent of this study is to:

- Show the sensitivity of the CMDS Execution capability to Disciplinary Method selection

  How does the selection of Disciplinary Methods effect CMDS performance parameter evaluation?

INPUT - Vehicle & Technology Information

- Vehicle / Component Specs ✓
- Vehicle Performance Data ✓
- Component Performance Data ✓
- Disciplinary Methods ✓
- Disciplinary Software ✓



AVD System Calibration

Create Sizing Tool to calibrate AVD System for given problem

- Vehicle hardware, design mission and operational requirements based on Vehicle specification
- Disciplinary methods based on reference component performance data

Vehicle Component Calibration

Create Analysis Tool to match Vehicle Component reference data

- Vehicle Component performance data used to create look-up tables
- Disciplinary Methods calibrated to match reference look-up tables

Vehicle Sizing Tool Creation

Create Sizing Tool for Vehicle Technology solution space visualization

- Vehicle hardware, design mission and operational requirements based on Vehicle specification
- Disciplinary Methods calibrated to match reference look-up tables

Figure 5-2 GHV Case Study Procedure Flow Chart

5.1 GHV Verification Study

The first case study is meant to show the validity of the CMDS Composition process through the composition of an analysis framework mean to re-create of performance results from the AFRL GHV reference study (Ruttle, Stork, and Liston 2012). This validation effort will follow a three step process: Building Block Creation, CMDS Composition, and CMDS Execution.

5.1.1 GHV Verification - Building Block Creation

The first step in the CMDS Composition process is the description of the problem being solved. As shown in chapter 3, there are three building block categories: Product, Analysis Process, and Disciplinary Methods. The following sections provide a description of the Product and Analysis Process building blocks created and/or selected for the GHV Verification CMDS. The description of Disciplinary Methods selected for the GHV Verification CMDS can be found in the Matching and Selecting sections of the CMDS Composition Process.

5.1.1.1 Product Description for GHV Verification

The GHV has been designed with the intent of matching a given propulsion system to a hypersonic vehicle configuration. This means that the first step in the design process has been the sizing of the engine, then a vehicle was made to fit around that sized engine. The propulsion system selected for the GHV is a scramjet, with the following attributes (see Figure 5-3):

- Inlet - Streamline traced inward turning inlet
- Isolator - Axisymmetric
- Combustor – Axisymmetric
- Nozzle – 3-D axisymmetric

The vehicle planform is designed around a given engine length and diameter, where the propulsion system (1-engine) is located along the centerline of the vehicle, see Figure 5-3. The configuration of the GHV is a wing-body with the following attributes, and hardware assumptions:

- Wing Planform – Cropped delta wing, underbody waverider shaping
- Control Surfaces – Split flaps, twin vertical tails
- Structure – Metallic structure
- Thermal Protection System – TPS coating along nose and leading edge, and inside engine

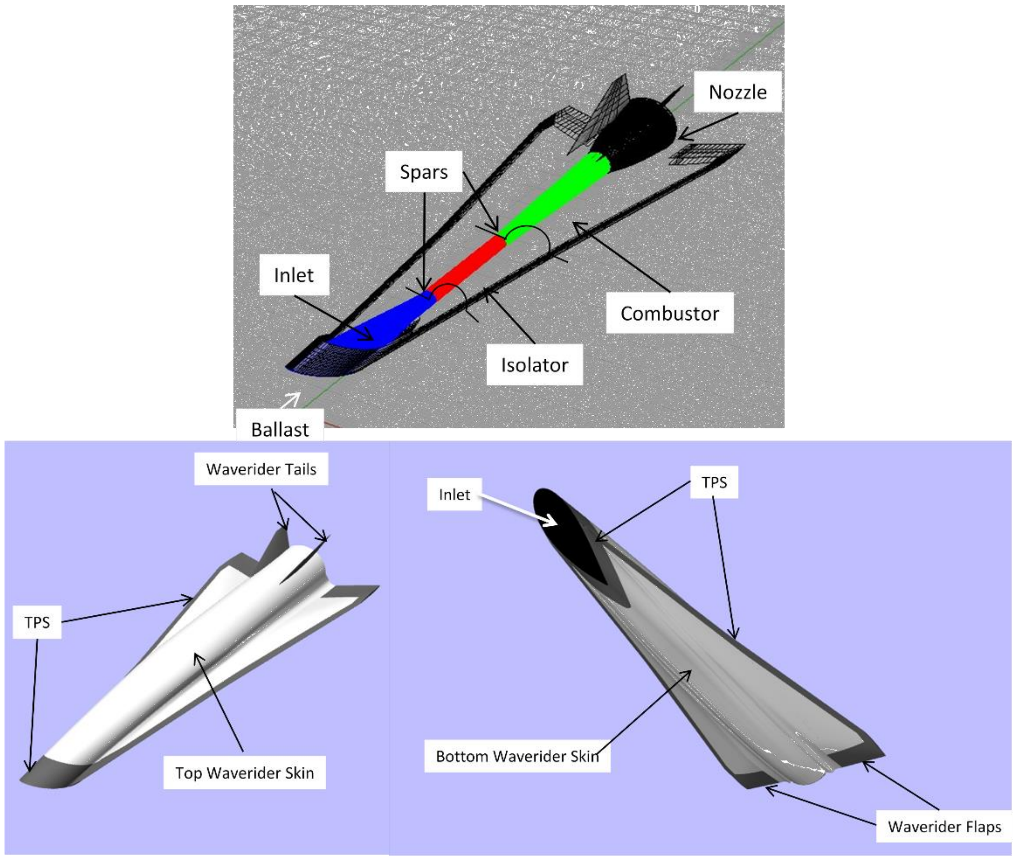


Figure 5-3 GHV Hardware Specification (Ruttle, Stork, and Liston 2012)

The flight profile for the GHV matches that from Figure 5-1. It is assumed that the vehicle is air-dropped, and subsequently boosted to the scramjet to Mach 4 at a dynamic pressure of 1500 lb/ft^2 [71,820 N/m^2]. The vehicle then climbs and accelerates along a constant dynamic pressure trajectory until it reaches the design Mach number of 6. Once at the design Mach number a 180° turn maneuver is executed at a g-loading of approximately 2 (Figure 5-4). Once a 180° heading is achieved the vehicle starts a constant Mach cruise segment.

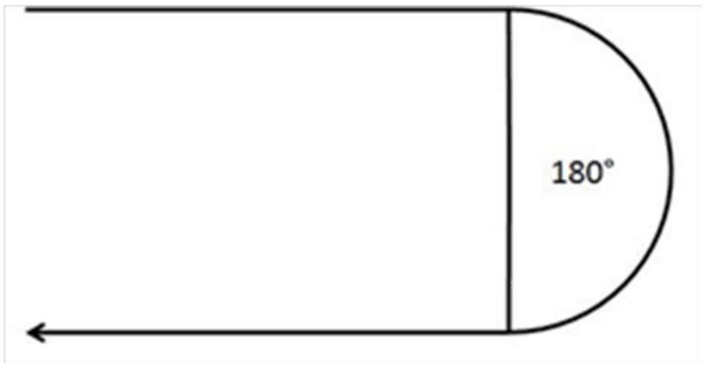
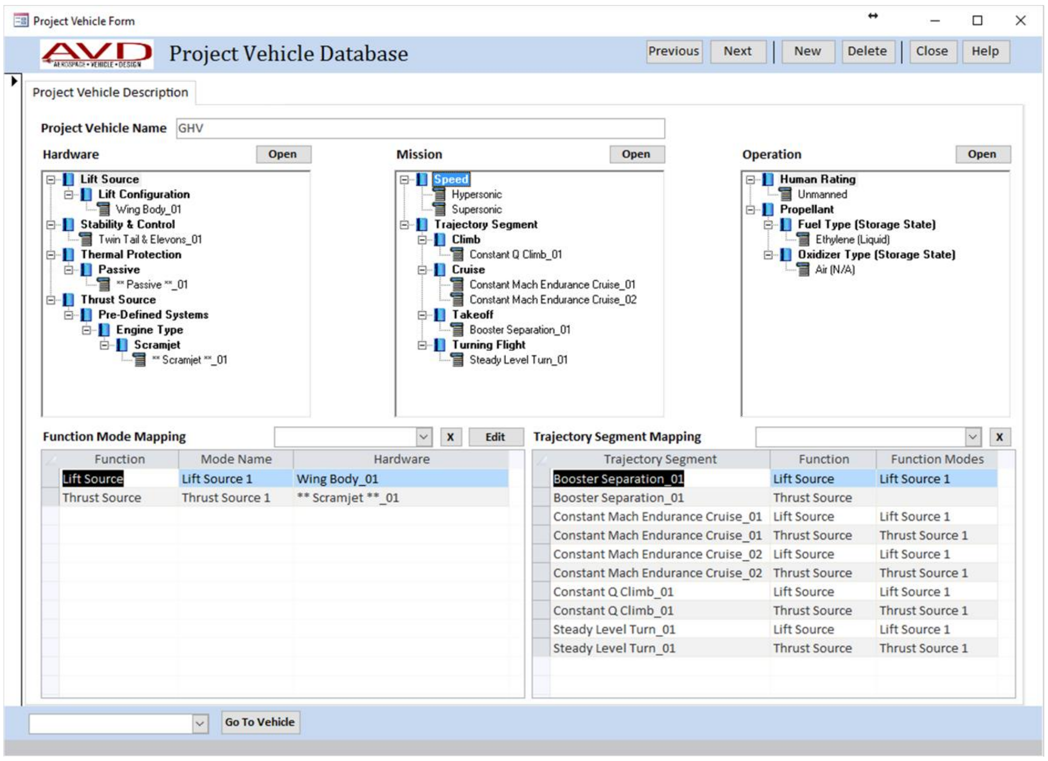


Figure 5-4 GHV Turn Maneuver - Ground Track (Ruttle, Stork, and Liston 2012)

The GHV design mission reference description do not specifically model the descent or landing portion of a nominal flight profile. The lack of a modelled landing or recovery segment (parachute) and the subsequent feedback of those design requirements towards vehicle and control effector sizing leads to a vehicle that has been sized to complete the climb, turn, and cruise segments only. This is consistent with the conceptual design of a vehicle that is not meant to land, and/or be recoverable.

The GHV hardware and operational events described are used as inputs into the Project Vehicle form in the AVD^{DBMS}. Figure 5-5 shows a snapshot of the GHV entry.



Project Vehicle Name GHV

Hardware
- Lift Source
  - Lift Configuration
    - Wing Body_01
- Stability & Control
  - Twin Tail & Elevons_01
- Thermal Protection
  - Passive
    - ** Passive **_01
- Thrust Source
  - Pre-Defined Systems
    - Engine Type
      - Scramjet
        - ** Scramjet **_01

Mission
- Speed
  - Hypersonic
  - Supersonic
- Trajectory Segment
  - Climb
    - Constant Q Climb_01
  - Cruise
    - Constant Mach Endurance Cruise_01
    - Constant Mach Endurance Cruise_02
  - Takeoff
    - Booster Separation_01
  - Turning Flight
    - Steady Level Turn_01

Operation
- Human Rating
  - Unmanned
- Propellant
  - Fuel Type (Storage State)
    - Ethylene (Liquid)
  - Oxidizer Type (Storage State)
    - Air (N/A)

Function Mode Mapping

Function	Mode Name	Hardware
Lift Source	Lift Source 1	Wing Body_01
Thrust Source	Thrust Source 1	** Scramjet **_01

Trajectory Segment Mapping

Trajectory Segment	Function	Function Modes
Booster Separation_01	Lift Source	Lift Source 1
Booster Separation_01	Thrust Source	
Constant Mach Endurance Cruise_01	Lift Source	Lift Source 1
Constant Mach Endurance Cruise_01	Thrust Source	Thrust Source 1
Constant Mach Endurance Cruise_02	Lift Source	Lift Source 1
Constant Mach Endurance Cruise_02	Thrust Source	Thrust Source 1
Constant Q Climb_01	Lift Source	Lift Source 1
Constant Q Climb_01	Thrust Source	Thrust Source 1
Steady Level Turn_01	Lift Source	Lift Source 1
Steady Level Turn_01	Thrust Source	Thrust Source 1

Figure 5-5 Product Specification for GHV

5.1.1.2 Analysis Process Description for GHV Validation

The Hypersonic Convergence sizing approach has been selected for the GHV Verification CMDS. Hypersonic Convergence has been used for transonic to hypersonic vehicle applications as developed at formerly McDonnell Aircraft Company between 1970 and 1990 (Czysz 2004). The first objective function for the Hypersonic Convergence process centers around the vehicle weight and balance budget. The results from the geometry, and performance matching modules are provided to assess the vehicle weight & volume available and required. For a given vehicle slenderness parameter ($\tau = \frac{V}{S_{pln}^{1.5}}$) , the planform area and wing loading are iterated through the total design process until weight & volume available equal weight & volume required. In order to do this the weight and volume of the vehicle are transformed into two equations that can be simultaneously solved for, see

Table 5-1.

Additionally, the analysis process begins with an estimate for TOGW. This estimate comes from and initial guess of the vehicle wing loading. The second objective function is a check to see if the initial guess for wing loading matches the wing loading output of the system.

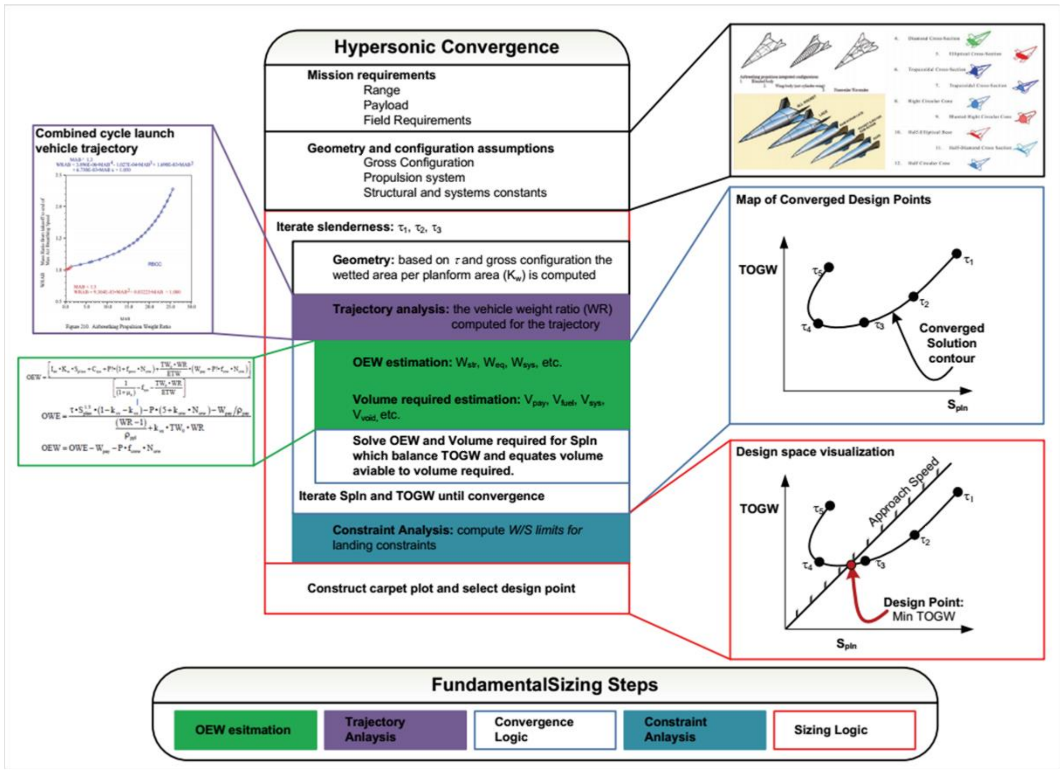

Hypersonic Convergence
Mission requirements
Range
Payload
Field Requirements
Geometry and configuration assumptions
Gross Configuration
Propulsion system
Structural and systems constants
Iterate slenderness: τ1, τ2, τ3
Geometry: based on τ and gross configuration the wetted area per planform area (Kw) is computed
Trajectory analysis: the vehicle weight ratio (WR) computed for the trajectory
OEW estimation: Wstr, Weq, Wsys, etc.
Volume required estimation: Vpay, Vfuel, Vsys, Vvoid, etc.
Solve OEW and Volume required for Spln which balance TOGW and equates volume aviable to volume required.
Iterate Spln and TOGW until convergence
Constraint Analysis: compute W/S limits for landing constraints
Construct carpet plot and select design point
Combined cycle launch vehicle trajectory
Map of Converged Design Points
TOGW
Spln
Converged Solution contour
Design space visualization
Approach Speed
Design Point: Min TOGW
FundamentalSizing Steps
OEW esitmation
Trajectory Anlaysis
Convergence Logic
Constraint Anlaysis
Sizing Logic

Figure 5-6 Hypersonic Convergence Analysis Process (Coleman 2010)

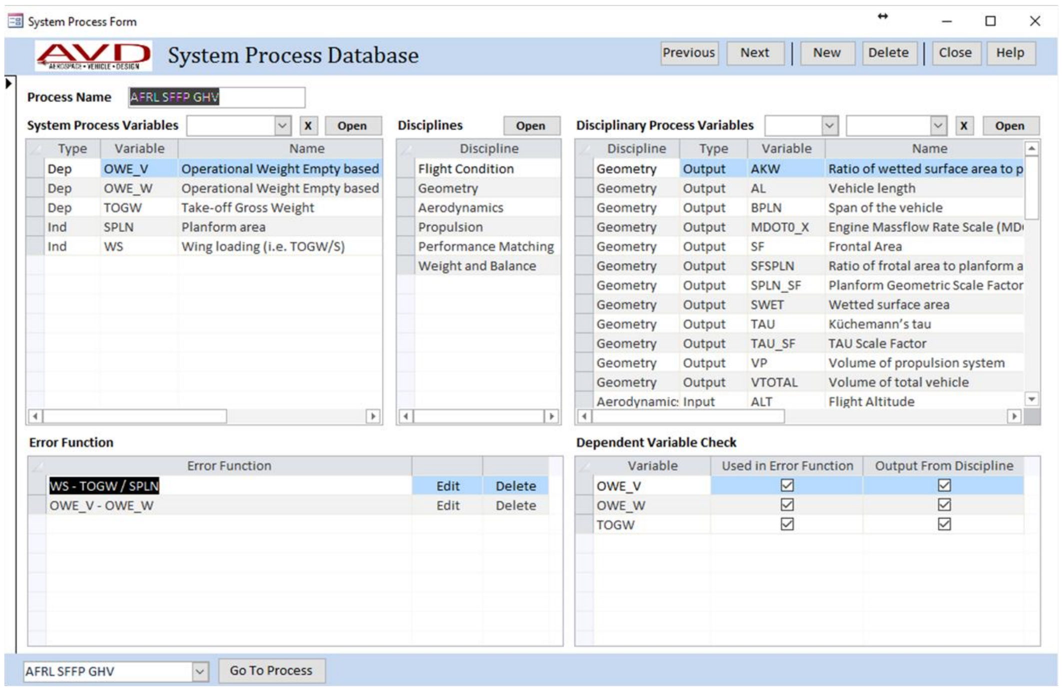
Table 5-1 Hypersonic Convergence Objective Functions (Czysz 2004)

Weight Budget	$\boldsymbol{OEW_W = \dfrac{W_{Fix} + W_{Eng} + W_{TPS}}{\dfrac{1}{1+\mu_a} - \dfrac{W_{str}}{OEW} - FWSYS}}$ **Note:** $\boldsymbol{OWE = OEW = W_{Payload}}$
Volume Budget	$OWE_V = \dfrac{V_{Total} - V_{Systems} - V_{Eng} - V_{Str} - V_{TPS} - V_{Void}}{\dfrac{WR-1}{\rho_{ppl} * g_0}}$
Wing Loading	$\dfrac{W}{S} = \dfrac{TOGW}{S_{pln}}$
Objective Functions	$OWE_V - OWE_W = 0$ $\left(\dfrac{W}{S}\right)_{Guess} - \dfrac{TOGW}{S_{pln}} = 0$

The analysis process also describes the disciplines included in the system analysis as well as the input and output variables associated with each of those disciplinary analyses. The disciplines included in the Hypersonic Convergence process are as follows:

- Flight Condition
- Geometry
- Aerodynamics
- Propulsion
- Performance Matching
- Weight and Balance

The Hypersonic Convergence Objective Functions, disciplines and their listed input and output variables are used as inputs into the Analysis Process form in the AVD^{DBMS}. Figure 5-7 shows a snapshot of the GHV entry.



Process Name: AFRL SFFP GHV

System Process Variables

Type	Variable	Name
Dep	OWE_V	Operational Weight Empty based
Dep	OWE_W	Operational Weight Empty based
Dep	TOGW	Take-off Gross Weight
Ind	SPLN	Planform area
Ind	WS	Wing loading (i.e. TOGW/S)

Disciplines

Discipline
Flight Condition
Geometry
Aerodynamics
Propulsion
Performance Matching
Weight and Balance

Disciplinary Process Variables

Discipline	Type	Variable	Name
Geometry	Output	AKW	Ratio of wetted surface area to p
Geometry	Output	AL	Vehicle length
Geometry	Output	BPLN	Span of the vehicle
Geometry	Output	MDOT0_X	Engine Massflow Rate Scale (MD
Geometry	Output	SF	Frontal Area
Geometry	Output	SFSPLN	Ratio of frotal area to planform a
Geometry	Output	SPLN_SF	Planform Geometric Scale Factor
Geometry	Output	SWET	Wetted surface area
Geometry	Output	TAU	Küchemann's tau
Geometry	Output	TAU_SF	TAU Scale Factor
Geometry	Output	VP	Volume of propulsion system
Geometry	Output	VTOTAL	Volume of total vehicle
Aerodynamic:	Input	ALT	Flight Altitude

Error Function

Error Function		
WS - TOGW / SPLN	Edit	Delete
OWE_V - OWE_W	Edit	Delete

Dependent Variable Check

Variable	Used in Error Function	Output From Discipline
OWE_V	☑	☑
OWE_W	☑	☑
TOGW	☑	☑

AFRL SFFP GHV

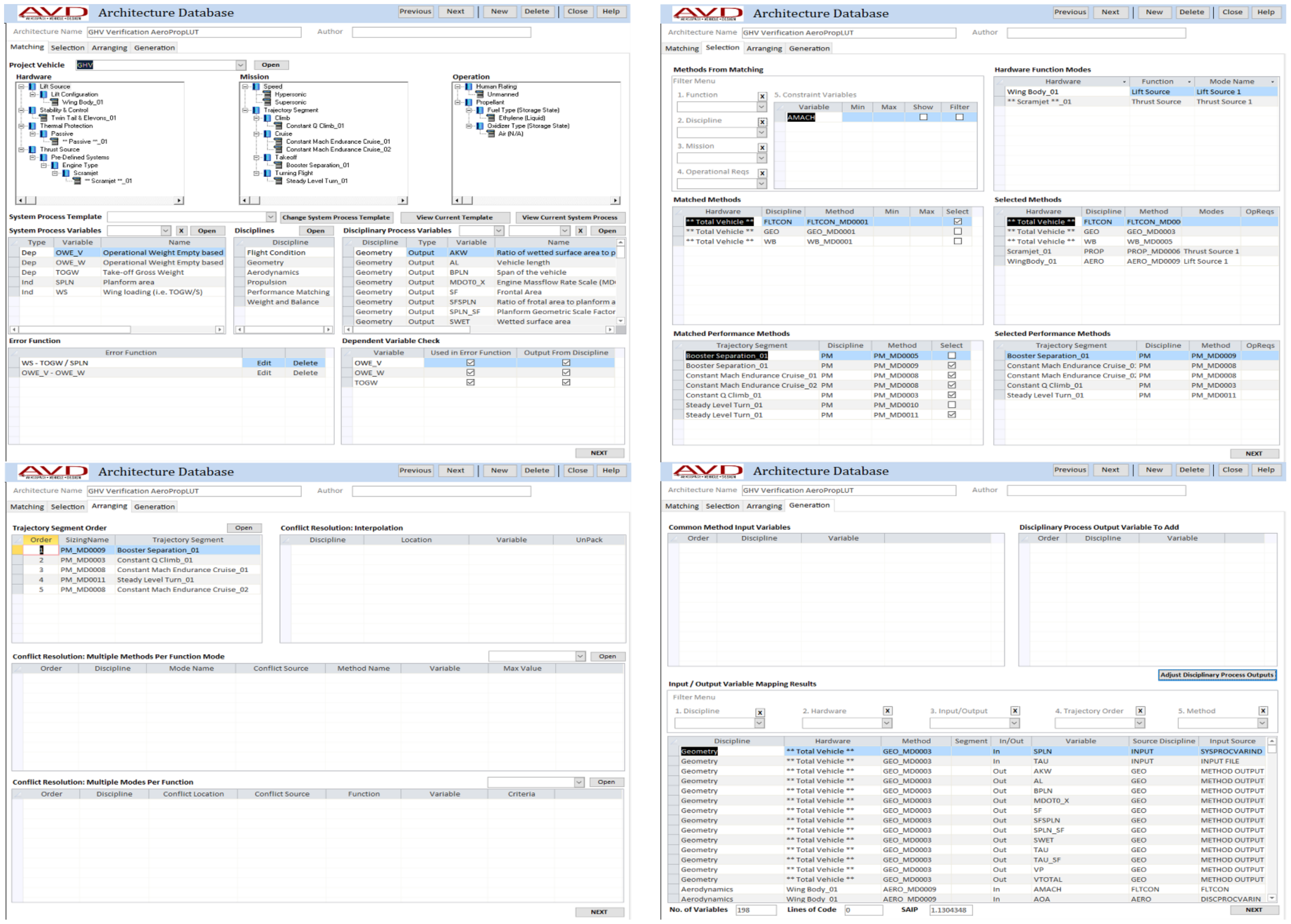
Figure 5-7 Analysis Process specification for GHV

5.1.2 GHV Verification - CMDS Composition Process

The following sections will walk through the Matching, Selecting, Arranging, and Generation steps of the CMDS Composition process. An overview of the inputs/outputs for each step can be seen in Figure 5-8.

5.1.2.1 Matching

The Matching step in the CMDS Composition process takes a given Product and Analysis Process definition and queries the AVD^{DBMS} returning all Disciplinary Methods that *Match* those specifications. Using the GHV Products and Hypersonic Convergence Analysis Process as inputs, the Matching step returns 32 disciplinary methods, see Table 5-1.



Architecture Name: GHV Verification AeroPropLUT
Author:

Matching

Project Vehicle: GHV

Hardware
- Lift Source
  - Lift Configuration
    - Wing Body_01
- Stability & Control
  - Twin Tail & Elevons_01
- Thermal Protection
  - Passive
    - ** Passive **_01
- Thrust Source
  - Pre-Defined Systems
    - Engine Type
      - Scramjet
        - ** Scramjet **_01

Mission
- Speed
  - Hypersonic
  - Supersonic
- Trajectory Segment
  - Climb
    - Constant Q Climb_01
  - Cruise
    - Constant Mach Endurance Cruise_01
    - Constant Mach Endurance Cruise_02
  - Takeoff
    - Booster Separation_01
  - Turning Flight
    - Steady Level Turn_01

Operation
- Human Rating
  - Unmanned
- Propellant
  - Fuel Type (Storage State)
    - Ethylene (Liquid)
  - Oxidizer Type (Storage State)
    - Air (N/A)

System Process Template

System Process Variables

Type	Variable	Name
Dep	OWE_V	Operational Weight Empty based
Dep	OWE_W	Operational Weight Empty based
Dep	TOGW	Take-off Gross Weight
Ind	SPLN	Planform area
Ind	WS	Wing loading (i.e. TOGW/S)

Disciplines

Discipline
Flight Condition
Geometry
Aerodynamics
Propulsion
Performance Matching
Weight and Balance

Disciplinary Process Variables

Discipline	Type	Variable	Name
Geometry	Output	AKW	Ratio of wetted surface area to p
Geometry	Output	AL	Vehicle length
Geometry	Output	BPLN	Span of the vehicle
Geometry	Output	MDOT0_X	Engine Massflow Rate Scale (MD
Geometry	Output	SF	Frontal Area
Geometry	Output	SFSPLN	Ratio of frotal area to planform a
Geometry	Output	SPLN_SF	Planform Geometric Scale Factor
Geometry	Output	SWET	Wetted surface area

Error Function

Error Function		
WS - TOGW / SPLN	Edit	Delete
OWE_V - OWE_W	Edit	Delete

Dependent Variable Check

Variable	Used in Error Function	Output From Discipline
OWE_V	☑	☑
OWE_W	☑	☑
TOGW	☑	☑

Selection

Methods From Matching

Filter Menu: 1. Function; 2. Discipline; 3. Mission; 4. Operational Reqs

5. Constraint Variables

Variable	Min	Max	Show	Filter
AMACH			☐	☐

Hardware Function Modes

Hardware	Function	Mode Name
Wing Body_01	Lift Source	Lift Source 1
** Scramjet **_01	Thrust Source	Thrust Source 1

Matched Methods

Hardware	Discipline	Method	Min	Max	Select
** Total Vehicle **	FLTCON	FLTCON_MD0001			☑
** Total Vehicle **	GEO	GEO_MD0001			☐
** Total Vehicle **	WB	WB_MD0001			☐

Selected Methods

Hardware	Discipline	Method	Modes	OpReqs
** Total Vehicle **	FLTCON	FLTCON_MD00		
** Total Vehicle **	GEO	GEO_MD0003		
** Total Vehicle **	WB	WB_MD0005		
Scramjet_01	PROP	PROP_MD0006	Thrust Source 1	
WingBody_01	AERO	AERO_MD0009	Lift Source 1	

Matched Performance Methods

Trajectory Segment	Discipline	Method	Select
Booster Separation_01	PM	PM_MD0005	☐
Booster Separation_01	PM	PM_MD0009	☑
Constant Mach Endurance Cruise_01	PM	PM_MD0008	☑
Constant Mach Endurance Cruise_02	PM	PM_MD0008	☑
Constant Q Climb_01	PM	PM_MD0003	☑
Steady Level Turn_01	PM	PM_MD0010	☐
Steady Level Turn_01	PM	PM_MD0011	☑

Selected Performance Methods

Trajectory Segment	Discipline	Method	OpReqs
Booster Separation_01	PM	PM_MD0009	
Constant Mach Endurance Cruise_0	PM	PM_MD0008	
Constant Mach Endurance Cruise_0	PM	PM_MD0008	
Constant Q Climb_01	PM	PM_MD0003	
Steady Level Turn_01	PM	PM_MD0011	

Arranging

Trajectory Segment Order

Order	SizingName	Trajectory Segment
1	PM_MD0009	Booster Separation_01
2	PM_MD0003	Constant Q Climb_01
3	PM_MD0008	Constant Mach Endurance Cruise_01
4	PM_MD0011	Steady Level Turn_01
5	PM_MD0008	Constant Mach Endurance Cruise_02

Conflict Resolution: Interpolation

Discipline	Location	Variable	UnPack

Conflict Resolution: Multiple Methods Per Function Mode

Order	Discipline	Mode Name	Conflict Source	Method Name	Variable	Max Value

Conflict Resolution: Multiple Modes Per Function

Order	Discipline	Conflict Location	Conflict Source	Function	Variable	Criteria

Generation

Common Method Input Variables

Order	Discipline	Variable

Disciplinary Process Output Variable To Add

Order	Discipline	Variable

Input / Output Variable Mapping Results

Filter Menu: 1. Discipline; 2. Hardware; 3. Input/Output; 4. Trajectory Order; 5. Method

Discipline	Hardware	Method	Segment	In/Out	Variable	Source Discipline	Input Source
Geometry	** Total Vehicle **	GEO_MD0003		In	SPLN	INPUT	SYSPROCVARIND
Geometry	** Total Vehicle **	GEO_MD0003		In	TAU	INPUT	INPUT FILE
Geometry	** Total Vehicle **	GEO_MD0003		Out	AKW	GEO	METHOD OUTPUT
Geometry	** Total Vehicle **	GEO_MD0003		Out	AL	GEO	METHOD OUTPUT
Geometry	** Total Vehicle **	GEO_MD0003		Out	BPLN	GEO	METHOD OUTPUT
Geometry	** Total Vehicle **	GEO_MD0003		Out	MDOT0_X	GEO	METHOD OUTPUT
Geometry	** Total Vehicle **	GEO_MD0003		Out	SF	GEO	METHOD OUTPUT
Geometry	** Total Vehicle **	GEO_MD0003		Out	SFSPLN	GEO	METHOD OUTPUT
Geometry	** Total Vehicle **	GEO_MD0003		Out	SPLN_SF	GEO	METHOD OUTPUT
Geometry	** Total Vehicle **	GEO_MD0003		Out	SWET	GEO	METHOD OUTPUT
Geometry	** Total Vehicle **	GEO_MD0003		Out	TAU	GEO	METHOD OUTPUT
Geometry	** Total Vehicle **	GEO_MD0003		Out	TAU_SF	GEO	METHOD OUTPUT
Geometry	** Total Vehicle **	GEO_MD0003		Out	VP	GEO	METHOD OUTPUT
Geometry	** Total Vehicle **	GEO_MD0003		Out	VTOTAL	GEO	METHOD OUTPUT
Aerodynamics	Wing Body_01	AERO_MD0009		In	AMACH	FLTCON	FLTCON
Aerodynamics	Wing Body_01	AERO_MD0009		In	AOA	AERO	DISCPROCVARIN

No. of Variables: 198 Lines of Code: 0 SAIP: 1.1304348

Figure 5-8 GHV Verification CMDS Composition Form Inputs

Table 5-2 GHV Verification CMDS - Matching Results

Hardware	Discipline	Method	Method Description
Total Vehicle	Flight Condition	FLTCON_MD0001	Atmospheric Model
Total Vehicle	Geometry	GEO_MD0001	Hypersonic Airbreather Geometry
Total Vehicle	Geometry	GEO_MD0002	Hypersonic Airbreather Geometry (AFRL SFFP CV10)
Total Vehicle	Geometry	GEO_MD0003	Hypersonic Airbreather Geometry (AFRL SFFP CV21)
Total Vehicle	Geometry	GEO_MD0004	Hypersonic Airbreather Geometry (AFRL SFFP CV10) ACAP
Total Vehicle	Performance Matching	PM_MD0003	Constant Q-Climb to an Altitude and Velocity at Small Flight Path Angles
Total Vehicle	Performance Matching	PM_MD0005	Fake Take off and Staged Method
Total Vehicle	Performance Matching	PM_MD0008	Constant Mach Range Cruise at Small Flight Path Angles
Total Vehicle	Performance Matching	PM_MD0008	Constant Mach Range Cruise at Small Flight Path Angles
Total Vehicle	Performance Matching	PM_MD0009	Launch Methods using WR
Total Vehicle	Performance Matching	PM_MD0010	Steady Level Turning Flight to Origin
Total Vehicle	Performance Matching	PM_MD0011	Steady Level Turning Flight by an Angle
Total Vehicle	Weight and Balance	WB_MD0001	Convergence Empty Weight Estimation Method
Total Vehicle	Weight and Balance	WB_MD0003	Convergence OWE Estimation for Scramjet w/ Landing skids
Total Vehicle	Weight and Balance	WB_MD0004	Convergence OWE Estimation for Scramjet w/ Parachute
Total Vehicle	Weight and Balance	WB_MD0005	Convergence OWE Estimation for Scramjet
Scramjet_01	Propulsion	PROP_MD0005	HAP Stream Thrust
Scramjet_01	Propulsion	PROP_MD0006	GHV Engine
Scramjet_01	Propulsion	PROP_MD0007	HAP Stream Thrust SERN CEA (C2H4 - Air)
Scramjet_01	Propulsion	PROP_MD0008	HAP Stream Thrust SERN CEA (C2H4 - Air) Look-Up Table
Scramjet_01	Propulsion	PROP_MD0008	HAP Stream Thrust SERN CEA (C2H4 - Air) Look-Up Table
Scramjet_01	Propulsion	PROP_MD0009	HAP Stream Thrust - GHV
WingBody_01	Aerodynamics	AERO_MD0005	MCAir Wing Body / Blended Body Subsonic Aerodynamics
WingBody_01	Aerodynamics	AERO_MD0005	MCAir Wing Body / Blended Body Subsonic Aerodynamics
WingBody_01	Aerodynamics	AERO_MD0006	MCAir Wing Body / Blended Body Transonic/Supersonic Aerodynamics
WingBody_01	Aerodynamics	AERO_MD0006	MCAir Wing Body / Blended Body Transonic/Supersonic Aerodynamics
WingBody_01	Aerodynamics	AERO_MD0007	MCAir Wing Body / Blended Body Supersonic/Hypersonic Aerodynamics
WingBody_01	Aerodynamics	AERO_MD0007	MCAir Wing Body / Blended Body Supersonic/Hypersonic Aerodynamics
WingBody_01	Aerodynamics	AERO_MD0008	HYFAC Wing-Body Aerodynamic Estimation
WingBody_01	Aerodynamics	AERO_MD0008	HYFAC Wing-Body Aerodynamic Estimation
WingBody_01	Aerodynamics	AERO_MD0009	GHV Aerodynamics Look-up Table
WingBody_01	Aerodynamics	AERO_MD0009	GHV Aerodynamics Look-up Table

5.1.2.2 Selecting

The Selecting step in the CMDS Composition process takes resulting list of Matched Disciplinary Methods and allows the designer to Select those to be integrated into the CMDS. This step of the CMDS Composition Process has not been automated and is intentionally meant to include the designer in the loop.

The GHV Verification case study is an attempt to prove that the CMDS Composition Process can compose an analysis framework from reference disciplinary data that can recreate the reference GHV results. Can the CMDS Composition Process use GHV disciplinary inputs to re-create the reference GHV multidisciplinary outputs? To this end, this CMDS will consist of mostly Disciplinary Methods made from reference disciplinary look-up table data. This lends to the selection of methods for Geometry (GEO_MD0003), Aerodynamics (AERO_MD0009), and Propulsion (PROP_MD0006). Each of these Disciplinary Methods has been created directly from reference GHV material.

There are however instances where disciplinary analysis tools and/or data was not available. The AVD^{DBMS} disciplinary methods library will be used to fill in the gaps for any disciplinary analysis that is not fully discernible in the GHV reference document (Ruttle, Stork, and Liston 2012). The trajectory analysis for the reference GHV vehicles has been done using an AFRL internal trajectory code. As this tool is not publicly available, trajectory segment methods derived from Miele (Miele 1962) and Vihn (Vinh 1981b) have been selected. Weight estimation for reference GHV vehicles has been done using AFWAT, an AFRL internal weight estimation spreadsheet. As this is not a publicly available tool, a weight estimation method from Czysz (Czysz 2004) has been selected and input parameters have been calibrated using reference GHV data.

A complete listing of disciplinary methods selected for the GHV Verification CMDS can be found in Table 5-3.

Table 5-3 Disciplinary Method Listing for GHV Verification CMDS

Discipline	Sizing Name	Method Title	Reference
Flight Condition	FLTCON_MD0001	Atmospheric Model	(MINZNER et al. 1959)
Geometry	GEO_MD0003	Hypersonic Airbreather Geometry (GHV)	(Ruttle, Stork, and Liston 2012)
Aerodynamics	AERO_MD0009	GHV Aerodynamics Look-up Table	(Ruttle, Stork, and Liston 2012)
Propulsion	PROP_MD0006	GHV Propulsion Look-up Table	(Ruttle, Stork, and Liston 2012)
Performance Matching	PM_MD0003	Constant Q-Climb to an Altitude and Velocity at Small Flight Path Angles	(Miele 1962)
	PM_MD0008	Constant Mach Range Cruise at Small Flight Path Angles	(Miele 1962)
	PM_MD0009	Launch Methods using WR	(Miele 1962)
	PM_MD0011	Steady Level Turn	(Vinh 1981b)
Weight & Balance	WB_MD0005	Convergence OWE Estimation for Scramjet (GHV)	(Czysz 2004; Ruttle, Stork, and Liston 2012)

5.1.2.3 Arranging

The Arranging step in the CMDS Composition process takes the list of Selected Disciplinary Methods along with the Product and Analysis Process definitions and creates a blueprint for the multidisciplinary integration of the CMDS. It is at this point in the CMDS Composition process that all CMDS data relationships are created.

The resulting CMDS blueprint (DSM) can be seen in Figure 5-9. The DSM is a diagonal matrix with two parts, where each part is separated into three sections: Selected Methods (Red), Objective Function (Green), Disciplinary Output Variables (Blue). Each entry in the DSM has a color and an arrow. The color denotes the field of interest, the arrow points towards the output of said field.

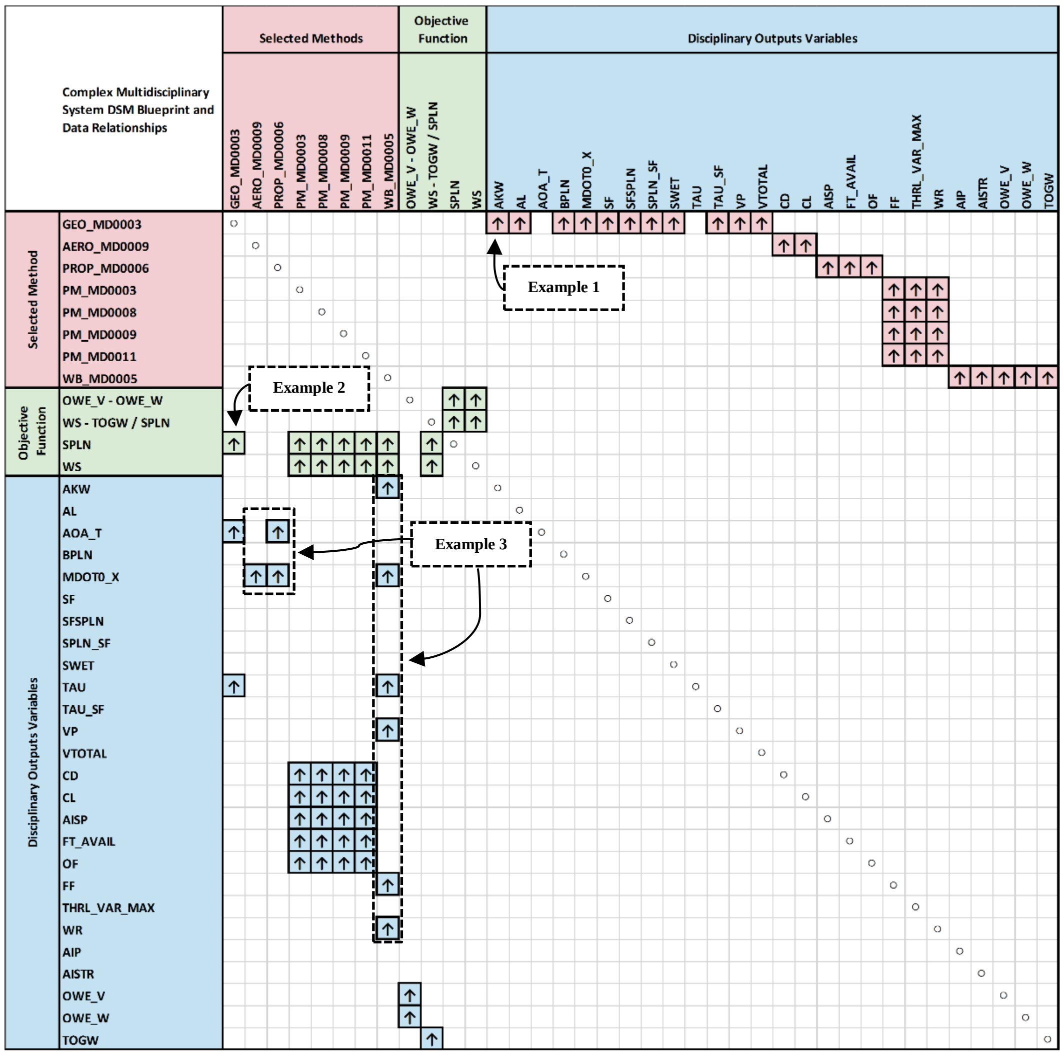


Figure 5-9 DSM for GHV Verification CMDS

For example in Figure 5-9, the top entry in the first column (Example 1) is green corresponding to the SPLN Objective Function Variable and has an arrow pointing upward at GEO_MD0003. This means that the SPLN Objective Function Variable is an input for the GEO_MD0003 Disciplinary Method. If we look at the first entry in the top row (Example 2), we have a red entry corresponding to the GEO_MD0003 Disciplinary Method, with an arrow pointing towards the AKW Disciplinary Output Variable. This means that AKW is an output of the GEO_MD0003 Disciplinary Method.

The green entries in Figure 5-9 denote entries dealing with the objective function of the CMDS. There are two types of green entries in the GHV Verification DSM: Objective Function, Objective Function Input Variables. Referring back to chapter 3, the objective form is of the form $y = f(x)$, where y refers to the objective function itself, and x refers to the objective function input variables. Figure 5-9 shows 2 objective functions and 2 objective function input variables; these values correspond to the choice of the Hypersonic Convergence Analysis Process in the Matching step. A measure of the effect of the objective function on the CMDS can be made by observing the numbers of green entries associated with the objective function input variables. Of a maximum possible 8 Disciplinary Method, SPLN is an input into 6, and WS in an input into 5. This gives the impression that the SPLN and WS variable have a high level of integration into the CMDS.

As mentioned in Chapter 3, Disciplinary Outputs Variables serve as the integration mechanism for the CMDS as they serve to transfer information from one discipline to another. Figure 5-9 shows 34 instances of a Disciplinary Output Variable being used as input into another disciplinary method; blue entries with arrows pointing up toward individual disciplinary methods. Viewing each Disciplinary Method and counting the number of Variable input gives an idea of the level of multidisciplinary integration of said method. In addition to counting the number of blue arrows, finding the source of the blue arrows gives a more complete understanding of the integration level of a given method.

As an example (Example 3), viewing Figure 5-9 we see that the AERO_MD0009 entry has 1 blue arrows, WB_MD0005 entry has 6 blue arrows, and PROP_MD0005 has 2 blue arrows. By viewing each of the blue entries (Disciplinary Output Variable) and finding the corresponding Disciplinary Method where the Variable originates (red entry), we can find the number of disciplines where the inputs are coming from. Again viewing Figure 5-9 we see that all of entries for AERO_ MD0009 and PROP_MD0008 originate from the

Geometry disciplinary analysis. This is in contrast to WB_MD0005, whose blue entries originate from the Geometry, and Performance Matching disciplinary analysis. So WB_MD0005 has more blue entries and those entries are from a wider range of disciplinary analyses, leading to the conclusion that WB_MD0005 has a higher level of integration than AERO_ MD0009 or PROP_MD0008.

The power of this visualization is that it gives a holistic representation of the integration level CMDS and a guide for exactly how information is being distributed throughout the CMDS. As the Arranging step of the CMDS Composition process in automated, the ability of the user to test different Disciplinary Method Selection scenarios and have instant feedback of the effect of that selection on the disciplinary integration level of the CMDS is possible.

5.1.2.4 Generation

The Generation step in the CMDS Composition process uses the CMDS integration Blueprint from the Arranging step to procedurally recall information from AVD^{DBMS} in order to create a custom tailored Analysis Framework. The Generation step creates source code file types: System Process Files, Disciplinary Process File, and Disciplinary Method Files, see Table 5-4.

The output GHV Verification CMDS is comprised of 3 System Process Files, 5 Disciplinary Process Files and 10 Disciplinary Method files; all files are written in MATLAB script notation. The 18 m-files total 5,948 lines of code and contain 196 unique variables.

Table 5-4 File Listing for the GHV Validation CMDS

File Type	File Name	Description
System	Driver	Runs GHV Verification CMDS
System	CONV_GHVVerificationAeroPropLUT	Controls all disciplinary process function calls

System	GHVVerificationAeroPropLUT	User input mechanism for required system and disciplinary method parameter inputs
Disciplinary Process	AERO_DP_GHVVerificationAeroPropLUT	Controls all aerodynamic disciplinary analysis function calls
Disciplinary Process	GEO_DP_GHVVerificationAeroPropLUT	Controls all geometry disciplinary analysis function calls
Disciplinary Process	PM_DP_GHVVerificationAeroPropLUT	Controls all performance matching disciplinary analysis function calls
Disciplinary Process	PROP_DP_GHVVerificationAeroPropLUT	Controls all propulsion disciplinary analysis function calls
Disciplinary Process	WB_DP_GHVVerificationAeroPropLUT	Controls all weight & balance disciplinary analysis function calls
Disciplinary Method	FLTCON_MD0001	Atmospheric Model
Disciplinary Method	TotalVehicle_GEO_MD0003	Hypersonic Airbreather Geometry (GHV)
Disciplinary Method	WingBody_01_AERO_MD0009	GHV Aerodynamics Look-up Table
Disciplinary Method	Scramjet_01_PROP_MD0006	GHV Propulsion Look-up Table
Disciplinary Method	ConstantQClimb_01_PM_MD0003	Constant Q-Climb to an Altitude and Velocity at Small Flight Path Angles
Disciplinary Method	ConstantMachEnduranceCruise_01_PM_MD0008	Constant Mach Range Cruise at Small Flight Path Angles
Disciplinary Method	ConstantMachEnduranceCruise_02_PM_MD0008	Constant Mach Range Cruise at Small Flight Path Angles
Disciplinary Method	BoosterSeparation_01_PM_MD0009	Launch Methods using WR
Disciplinary Method	SteadyLevelTurn_01_PM_MD0011	Steady Level Turn
Disciplinary Method	TotalVehicle_WB_MD0005	Convergence OWE Estimation for Scramjet (GHV)

5.1.3 GHV Verification - CMDS Execution

In order to assess the syntactic and semantic composability of the AVD^{DBMS}, an assessment of capability of the GHV Verification CMDS to recreate reference performance results has been undertaken. A snapshot of the input file for the GHV Verification CMDS can be found in Appendix C. Input values not explicitly stated in the reference GHV material have been assumed using nominal values. Figure 5-10 provides the comparative results for a selection of vehicle performance parameters.

GHV 5X - Aerodynamics Look-Up Table

Parameter	*Units*	*GHV 5X*	*AVD*	*% Diff*
TAU		0.066	0.066	0.0%
SPLN	m^2	19.5	19.5	0.1%
WS	N/m^2	1870	1846	-1.3%
AL	m	9.99	10.01	0.2%
BPLN	m	3.33	3.34	0.2%
MDOT0_X		5	5.00	0.1%
L/D_{max}		4.57	4.28	-6.3%
FF		0.3835	0.389	1.4%
Weight	*Units*	*GHV 5X*	*AVD*	*% Diff*
TOGW	N	36456	36020	-1.2%
OWE	N	22476	22012	-2.1%
WFUEL	N	13980	14008	0.2%
WMARGIN	N	2225	2145	-3.6%
WP	N	5890	5894	0.1%
WSTR	N	10065	9858	-2.1%
WSYS	N	2793	2639	-5.5%
WTPS	N	1503	1476	-1.8%
Volume	*Units*	*GHV 5X*	*AVD*	*% Diff*
VTOTAL	m^3	5.64	5.648	0.1%
VFUEL	m^3	3.403	3.410	0.2%
VP	m^3	0.280	0.280	0.1%
VSYS	m^3	0.398	0.370	-7.0%
VVOID	m^3	0.314	0.285	-9.2%
VSTR	m^3	1.170	1.227	4.8%
VTPS	m^3	0.0767	0.0760	-0.9%

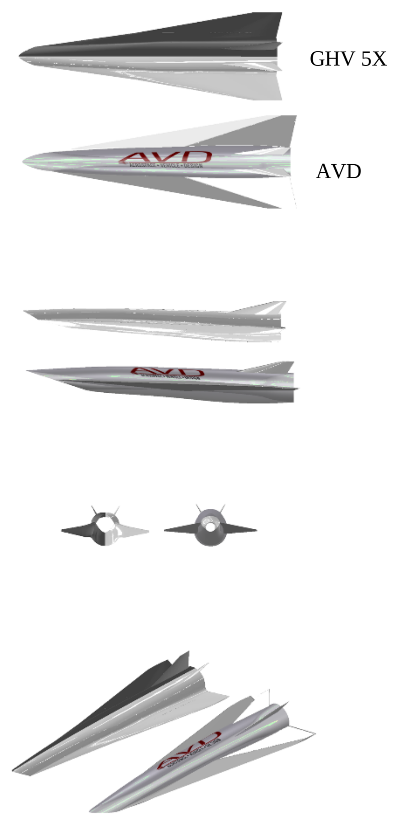


Figure 5-10 GHV Verification CMDS Execution Results

One of the goals of the research has been to ensure syntactic composability of AVD^{DBMS}. To this end, the first assessment to be made is the whether the GHV Verification CMDS is a full stand-alone analysis framework. Syntactic Composability refers to whether a system has the correct data relationships and connections in order for it to run. AVD^{DBMS}

creates these data links each time it creates a CMDS. Each set of data relationships are created in the Arranging step of the CMDS Composition Process and implemented in the Generation step. The resulting source code for the GHV Verification CMDS is able to run through and create output data with only modification of the CMDS input file.

The next question to be answered is whether the composed CMDS is semantically valid to model the GHV problem. During the CMDS Composition process the Matching and Selection steps had direct user input. These inputs are meant to guide the user toward selecting Disciplinary Methods that are semantically valid toward the problem being solved. In the case of the GHV Verification CMDS, the Disciplinary Methods (Geometry, Aerodynamics, and Propulsion) selected have been derived from reference look-up table data when available. The remaining disciplinary analysis methods (Performance Matching, and Weight and Balance) have been selected as part of the CMDS Composition Process.

Figure 5-10 shows general agreement between the reference GHV data and the GHV Verification CMDS results. It should be noted that the Objective Function associated with the GHV Verification CMDS is a function of Planform Area, and Wing Loading. Initial guesses for these parameters are input, and the CMDS moves these values to drive the Objective Functions to 0. The resulting values for planform are and wing loading show a percent difference of 0.1% and -1.3% respectively. Meaning the analysis converged to this point, as it tried to match weight and volume required versus weight and volume available. A more complete listing of result data can be found in Appendix C.

5.2 GHV Adaptation

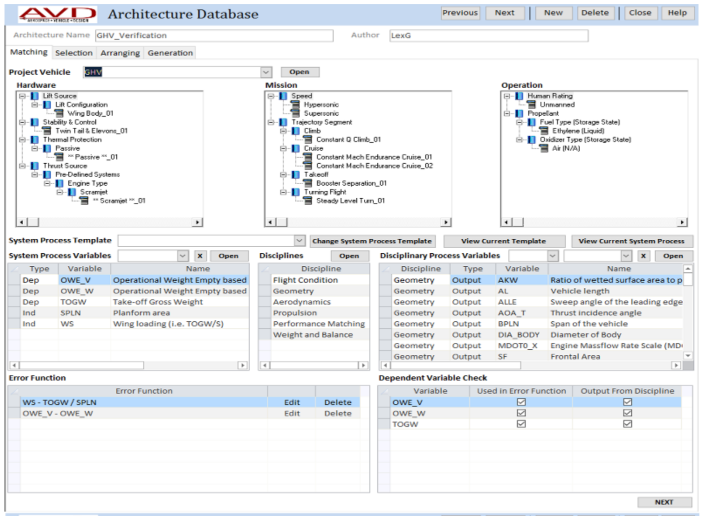
The second case study is meant to show the capability of the CMDS Composition to compose a CMDS meant to re-create the performance results from the AFRL GHV reference study (Ruttle, Stork, and Liston 2012) using an empirical aerodynamics estimation method. This validation effort will follow the same three step process described by the first case study: Building Block Creation, CMDS Composition, and CMDS Execution. As certain steps have not changed from the first case study, certain aspects will not be repeated here but rather referenced to the earlier discussion.

5.2.1 GHV Adaptation - Building Block Creation

The Product (GHV) and Analysis Process (Hypersonic Convergence) Building Blocks for the GHV Adaptation case study matches those of the GHV Validation study. The goal of this case study has been to show the effect that Disciplinary Method selection has on the integration level of the resulting CMDS. As so, in order to directly compare the results of this case study with those of the GHV Validation study, the inputs into the system remain constant.

5.2.2 GHV Adaptation - CMDS Composition Process

The following sections will walk through the Matching, Selecting, Arranging, and Generation steps of the CMDS Composition process. An overview of the inputs/outputs for each step can be seen in Figure 5-11.



Architecture Name: GHV_Verification Author: LexG

Project Vehicle: GHV

Hardware
- Lift Source
  - Lift Configuration
    - Wing Body_01
- Stability & Control
  - Twin Tail & Elevons_01
- Thermal Protection
  - Passive
    - ** Passive **_01
- Thrust Source
  - Pre-Defined Systems
    - Engine Type
      - Scramjet
        - ** Scramjet **_01

Mission
- Speed
  - Hypersonic
  - Supersonic
- Trajectory Segment
  - Climb
    - Constant Q Climb_01
  - Cruise
    - Constant Mach Endurance Cruise_01
    - Constant Mach Endurance Cruise_02
  - Takeoff
    - Booster Separation_01
  - Turning Flight
    - Steady Level Turn_01

Operation
- Human Rating
  - Unmanned
- Propellant
  - Fuel Type (Storage State)
    - Ethylene (Liquid)
  - Oxidizer Type (Storage State)
    - Air (N/A)

System Process Template

System Process Variables

Type	Variable	Name
Dep	OWE_V	Operational Weight Empty based
Dep	OWE_W	Operational Weight Empty based
Dep	TOGW	Take-off Gross Weight
Ind	SPLN	Planform area
Ind	WS	Wing loading (i.e. TOGW/S)

Disciplines

Discipline
Flight Condition
Geometry
Aerodynamics
Propulsion
Performance Matching
Weight and Balance

Disciplinary Process Variables

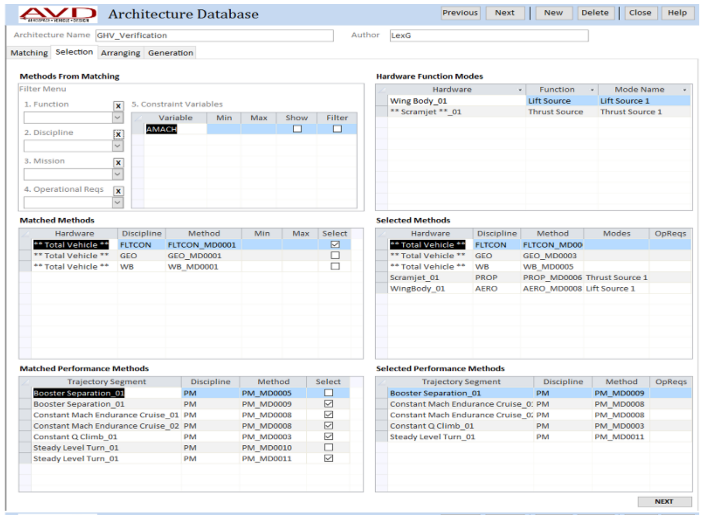
Discipline	Type	Variable	Name
Geometry	Output	AKW	Ratio of wetted surface area to p
Geometry	Output	AL	Vehicle length
Geometry	Output	ALLE	Sweep angle of the leading edge
Geometry	Output	AOA_T	Thrust incidence angle
Geometry	Output	BPLN	Span of the vehicle
Geometry	Output	DIA_BODY	Diameter of Body
Geometry	Output	MDOT0_X	Engine Massflow Rate Scale (MD
Geometry	Output	SF	Frontal Area

Error Function

Error Function		
WS - TOGW / SPLN	Edit	Delete
OWE_V - OWE_W	Edit	Delete

Dependent Variable Check

Variable	Used in Error Function	Output From Discipline
OWE_V	✓	✓
OWE_W	✓	✓
TOGW	✓	✓



Architecture Name: GHV_Verification Author: LexG

Methods From Matching

5. Constraint Variables

Variable	Min	Max	Show	Filter
AMACH			☐	☐

Hardware Function Modes

Hardware	Function	Mode Name
Wing Body_01	Lift Source	Lift Source 1
** Scramjet **_01	Thrust Source	Thrust Source 1

Matched Methods

Hardware	Discipline	Method	Min	Max	Select
** Total Vehicle **	FLTCON	FLTCON_MD0001			✓
** Total Vehicle **	GEO	GEO_MD0001			☐
** Total Vehicle **	WB	WB_MD0001			☐

Selected Methods

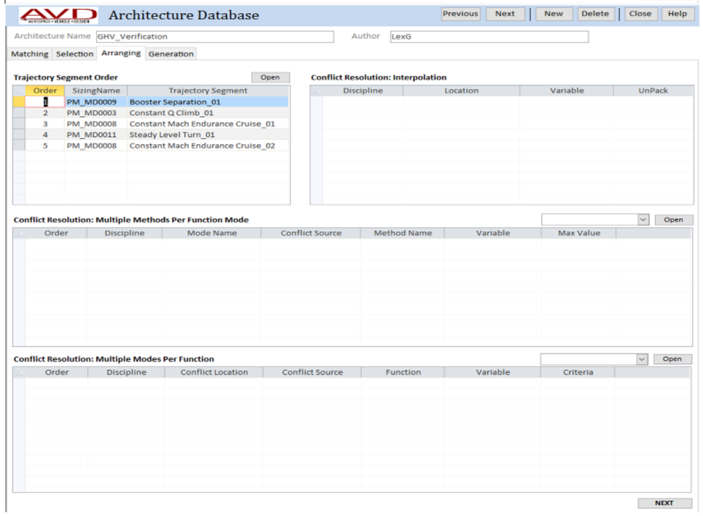
Hardware	Discipline	Method	Modes	OpReqs
** Total Vehicle **	FLTCON	FLTCON_MD00		
** Total Vehicle **	GEO	GEO_MD0003		
** Total Vehicle **	WB	WB_MD0005		
Scramjet_01	PROP	PROP_MD0006	Thrust Source 1	
WingBody_01	AERO	AERO_MD0008	Lift Source 1	

Matched Performance Methods

Trajectory Segment	Discipline	Method	Select
Booster Separation_01	PM	PM_MD0005	☐
Booster Separation_01	PM	PM_MD0009	✓
Constant Mach Endurance Cruise_01	PM	PM_MD0008	✓
Constant Mach Endurance Cruise_02	PM	PM_MD0008	✓
Constant Q Climb_01	PM	PM_MD0003	✓
Steady Level Turn_01	PM	PM_MD0010	☐
Steady Level Turn_01	PM	PM_MD0011	✓

Selected Performance Methods

Trajectory Segment	Discipline	Method	OpReqs
Booster Separation_01	PM	PM_MD0009	
Constant Mach Endurance Cruise_0	PM	PM_MD0008	
Constant Mach Endurance Cruise_0	PM	PM_MD0008	
Constant Q Climb_01	PM	PM_MD0003	
Steady Level Turn_01	PM	PM_MD0011	



Architecture Name: GHV_Verification Author: LexG

Trajectory Segment Order

Order	SizingName	Trajectory Segment
1	PM_MD0009	Booster Separation_01
2	PM_MD0003	Constant Q Climb_01
3	PM_MD0008	Constant Mach Endurance Cruise_01
4	PM_MD0011	Steady Level Turn_01
5	PM_MD0008	Constant Mach Endurance Cruise_02

Conflict Resolution: Interpolation

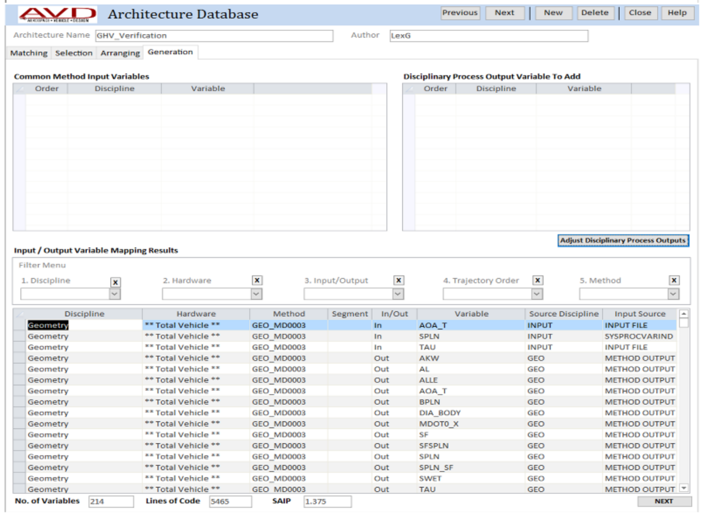
Discipline	Location	Variable	UnPack

Conflict Resolution: Multiple Methods Per Function Mode

Order	Discipline	Mode Name	Conflict Source	Method Name	Variable	Max Value

Conflict Resolution: Multiple Modes Per Function

Order	Discipline	Conflict Location	Conflict Source	Function	Variable	Criteria



Architecture Name: GHV_Verification Author: LexG

Common Method Input Variables

Order	Discipline	Variable

Disciplinary Process Output Variable To Add

Order	Discipline	Variable

Input / Output Variable Mapping Results

Discipline	Hardware	Method	Segment	In/Out	Variable	Source Discipline	Input Source
Geometry	** Total Vehicle **	GEO_MD0003		In	AOA_T	INPUT	INPUT FILE
Geometry	** Total Vehicle **	GEO_MD0003		In	SPLN	INPUT	SYSPROCVARIND
Geometry	** Total Vehicle **	GEO_MD0003		In	TAU	INPUT	INPUT FILE
Geometry	** Total Vehicle **	GEO_MD0003		Out	AKW	GEO	METHOD OUTPUT
Geometry	** Total Vehicle **	GEO_MD0003		Out	AL	GEO	METHOD OUTPUT
Geometry	** Total Vehicle **	GEO_MD0003		Out	ALLE	GEO	METHOD OUTPUT
Geometry	** Total Vehicle **	GEO_MD0003		Out	AOA_T	GEO	METHOD OUTPUT
Geometry	** Total Vehicle **	GEO_MD0003		Out	BPLN	GEO	METHOD OUTPUT
Geometry	** Total Vehicle **	GEO_MD0003		Out	DIA_BODY	GEO	METHOD OUTPUT
Geometry	** Total Vehicle **	GEO_MD0003		Out	MDOT0_X	GEO	METHOD OUTPUT
Geometry	** Total Vehicle **	GEO_MD0003		Out	SF	GEO	METHOD OUTPUT
Geometry	** Total Vehicle **	GEO_MD0003		Out	SFSPLN	GEO	METHOD OUTPUT
Geometry	** Total Vehicle **	GEO_MD0003		Out	SPLN	GEO	METHOD OUTPUT
Geometry	** Total Vehicle **	GEO_MD0003		Out	SPLN_SF	GEO	METHOD OUTPUT
Geometry	** Total Vehicle **	GEO_MD0003		Out	SWET	GEO	METHOD OUTPUT
Geometry	** Total Vehicle **	GEO_MD0003		Out	TAU	GEO	METHOD OUTPUT

No. of Variables 214 Lines of Code 5465 SAIP 1.375

Figure 5-11 GHV Adaptation CMDS Composition Form Inputs

5.2.2.1 Matching

The Matched listing of Disciplinary Methods for the GHV Adaptation CMDS is exactly the same as that from the previous case study, see Table 5-2. This is a result of the same Product (GHV) and Analysis Process (Hypersonic Convergence) being used as inputs into the CMDS Composition Process.

5.2.2.2 Selecting

A listing of selected methods for the GHV Adaptation CMDS can be found in Table 5-5. The Aerodynamics method, AERO_MD0008, has been selected in place of the GHV aerodynamics look-up table method AERO_MD00009. The other methods selected match those selected for the GHV Validation case study.

Table 5-5 Disciplinary Method Listing for GHV Adaptation Study

Discipline	Sizing Name	Method Title	Reference
Flight Condition	FLTCON_MD0001	Atmospheric Model	(MINZNER et al. 1959)
Geometry	GEO_MD0003	Hypersonic Airbreather Geometry (GHV)	(Ruttle, Stork, and Liston 2012)
Aerodynamics	**AERO_MD0008**	**Hypersonic Convergence Aerodynamic Estimation Method**	**(Czysz 2004; Sforza 2016)**
Propulsion	PROP_MD0006	GHV Propulsion Look-up Table	(Ruttle, Stork, and Liston 2012)
Performance Matching	PM_MD0003	Constant Q-Climb to an Altitude and Velocity at Small Flight Path Angles	(Miele 1962)
	PM_MD0008	Constant Mach Range Cruise at Small Flight Path Angles	(Miele 1962)
	PM_MD0009	Launch Methods using WR	(Miele 1962)
	PM_MD0011	Steady Level Turn	(Vinh 1981b)
Weight & Balance	WB_MD0005	Convergence OWE Estimation for Scramjet (GHV)	(Czysz 2004; Ruttle, Stork, and Liston 2012)

The aerodynamics estimation method, AERO_MD0008, is a combination of estimation techniques from Czysz (Czysz 2004) and Sforza (Sforza 2016). The aerodynamic coefficients (C_N, C_A, C_L, and C_D) are a function of the lift curve slope $\left(C_{L_\alpha}\right)$, the lift induced drag factor (L'), and the parasite drag coefficient $\left(C_{D_0}\right)$.

$$C_L = \left(\alpha - \alpha_{C_L=0}\right) C_{L_\alpha}$$

$$C_D = C_{D_0} + L' C_L^2$$

$$C_N = C_L \cos\alpha + C_D \sin\alpha$$

$$C_A = -C_L \sin\alpha + C_D \cos\alpha$$

For more information concerning the implementation of the method please refer to Appendix B – Methods Library.

5.2.2.3 Arranging

The resulting CMDS blueprint (DSM) can be seen in Figure 5-12. The DSM is a diagonal matrix with two parts, where each part is separated into three sections: Selected Methods (Red), Objective Function (Green), Disciplinary Output Variables (Blue). Each entry in the DSM has a color and an arrow. The color denotes the field of interest, the arrow points towards the output of said field.

The previous discussion of the CMDS DSM focused on the assessment of the integration level of the Objective Function and Disciplinary Methods based on the number and variation of disciplinary output variable interaction. The impetus of the previous discussion has been to compare entries within the same CMDS against each other to gain an understanding of the integration landscape of that specific CMDS. An additional aspect of the visual DSM representation is the ability to compare CMDS blueprints against one another in order to judge the effect of decision made in the composition of one CMDS versus another.

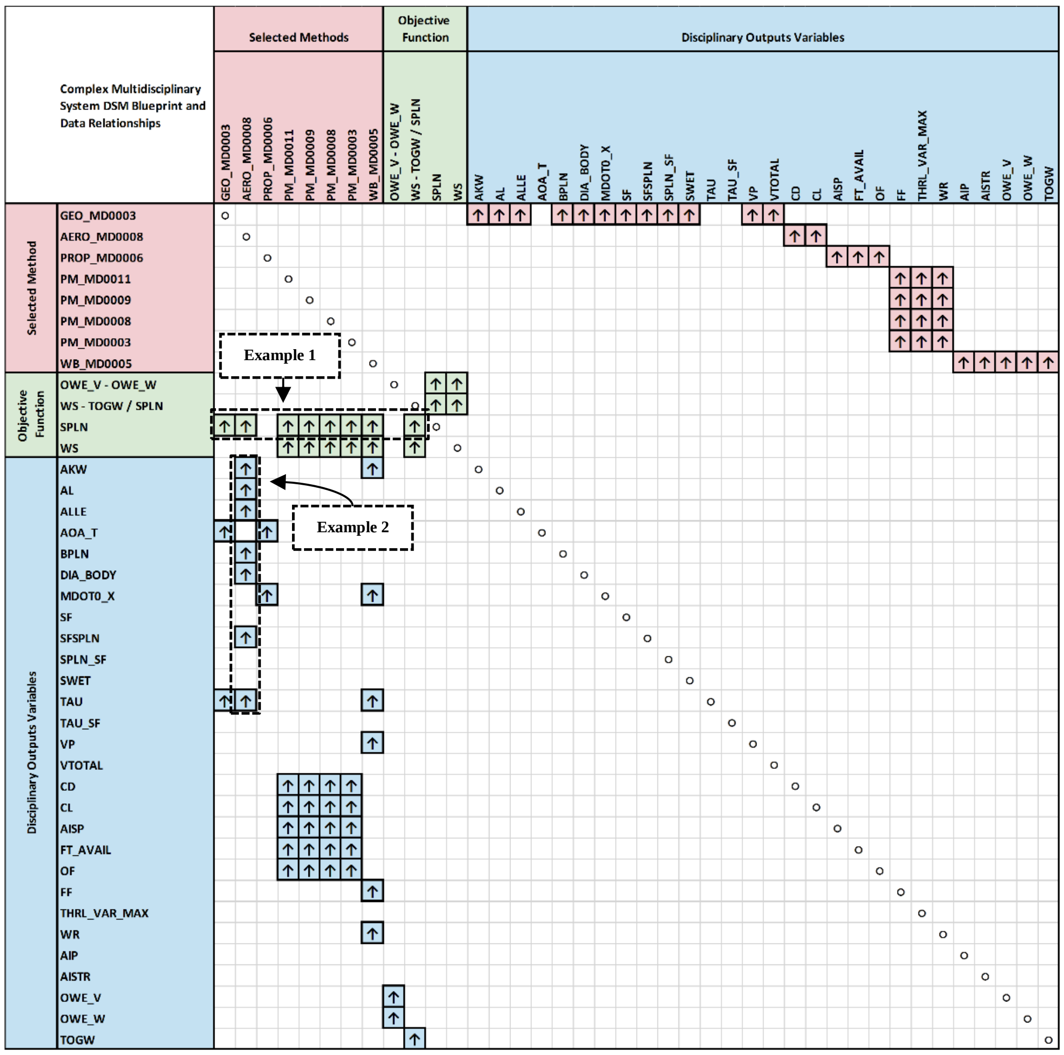


Figure 5-12 DSM for GHV Adaptation CMDS

As discussed previously, the green entries in Figure 5-12 denote entries dealing with the objective function of the CMDS. There are two types of green entries in the GHV Verification DSM: Objective Function, Objective Function Input Variables. The previous CMDS (GHV Validation Figure 5-9) showed that of a maximum possible 8 Disciplinary Methods, SPLN is an input into 6, and WS in an input into 5. Viewing Figure 5-12 (Example 1), we see that SPLN is an input into 7 Disciplinary Methods and WS in an input into 5.

This means that our choice of Aerodynamic Method has directly affected the integration level of the Objective Function.

When viewing the integration level of aerodynamics in the GHV Adaptation CMDS, Figure 5-12 (Example 2) shows that AERO_MD0008 has 7 blue entries. This is in contrast to the one blue entry found associated with the aerodynamics method from the GHV Validation CMDS (AERO_MD0009, Figure 5-9). As with the previous case, all of entries for AERO_MD0008 originate from the Geometry disciplinary analysis. This leads to the conclusion that the aerodynamics analysis in the GHV Adaptation CMDS is more integrated than that of the GHV Verification CMDS.

5.2.2.4 Generation

The output GHV Adaptation CMDS is comprised of 3 System Process Files, 5 Disciplinary Process Files and 10 Disciplinary Method files; all files are written in MATLAB script notation. The 18 m-files total 5,465 lines of code and contain 214 unique variables.

Table 5-6 File Listing for the GHV Validation CMDS

File Type	File Name	Description
System	Driver	Runs GHV Verification CMDS
System	CONV_GHVVerificationAeroPropLUT	Controls all disciplinary process function calls
System	GHVVerificationAeroPropLUT	User input mechanism for required system and disciplinary method parameter inputs
Disciplinary Process	AERO_DP_GHVVerificationAeroPropLUT	Controls all aerodynamic disciplinary analysis function calls
Disciplinary Process	GEO_DP_GHVVerificationAeroPropLUT	Controls all geometry disciplinary analysis function calls
Disciplinary Process	PM_DP_GHVVerificationAeroPropLUT	Controls all performance matching disciplinary analysis function calls
Disciplinary Process	PROP_DP_GHVVerificationAeroPropLUT	Controls all propulsion disciplinary analysis function calls
Disciplinary Process	WB_DP_GHVVerificationAeroPropLUT	Controls all weight & balance disciplinary analysis function calls

Disciplinary Method	FLTCON_MD0001	Atmospheric Model
Disciplinary Method	TotalVehicle_GEO_MD0003	Hypersonic Airbreather Geometry (GHV)
Disciplinary Method	WingBody_01_AERO_MD0008	Hypersonic Convergence Aerodynamic Estimation Method
Disciplinary Method	Scramjet_01_PROP_MD0006	GHV Propulsion Look-up Table
Disciplinary Method	ConstantQClimb_01_PM_MD0003	Constant Q-Climb to an Altitude and Velocity at Small Flight Path Angles
Disciplinary Method	ConstantMachEnduranceCruise_01_PM_MD0008	Constant Mach Range Cruise at Small Flight Path Angles
Disciplinary Method	ConstantMachEnduranceCruise_02_PM_MD0008	Constant Mach Range Cruise at Small Flight Path Angles
Disciplinary Method	BoosterSeparation_01_PM_MD0009	Launch Methods using WR
Disciplinary Method	SteadyLevelTurn_01_PM_MD0011	Steady Level Turn
Disciplinary Method	TotalVehicle_WB_MD0005	Convergence OWE Estimation for Scramjet (GHV)

5.2.3 GHV Validation - CMDS Execution

In order to assess the syntactic and semantic composability of the AVD^{DBMS}, an assessment of capability of the GHV Adaptation CMDS to recreate reference performance results has been undertaken. A snapshot of the input file for the GHV Adaptation CMDS can be found in Appendix D. Input values not explicitly stated in the reference GHV material have been assumed using nominal values. Figure 5-13 provides the comparative results for a selection of vehicle performance parameters.

GHV 5X - Aerodynamics Estimation Method

Parameter	*Units*	*GHV 5X*	*AVD*	*% Diff*
TAU		0.066	0.066	0.0%
SPLN	m^2	19.5	19.5	0.0%
WS	N/m^2	1870	1845	-1.3%
AL	m	9.99	10.01	0.2%
BPLN	m	3.33	3.34	0.2%
MDOT0_X		5	5.00	0.0%
L/D_{max}		4.57	4.69	2.6%
FF		0.3835	0.389	1.4%
Weight	*Units*	*GHV 5X*	*AVD*	*% Diff*
TOGW	N	36456	35989	-1.3%
OWE	N	22476	22000	-2.1%
WFUEL	N	13980	13989	0.1%
WMARGIN	N	2225	2143	-3.7%
WP	N	5890	5891	0.0%
WSTR	N	10065	9851	-2.1%
WSYS	N	2793	2638	-5.6%
WTPS	N	1503	1474	-1.9%
Volume	*Units*	*GHV 5X*	*AVD*	*% Diff*
VTOTAL	m^3	5.64	5.644	0.0%
VFUEL	m^3	3.403	3.405	0.1%
VP	m^3	0.280	0.280	0.0%
VSYS	m^3	0.398	0.370	-7.1%
VVOID	m^3	0.314	0.285	-9.3%
VSTR	m^3	1.170	1.227	4.9%
VTPS	m^3	0.0767	0.0759	-1.0%

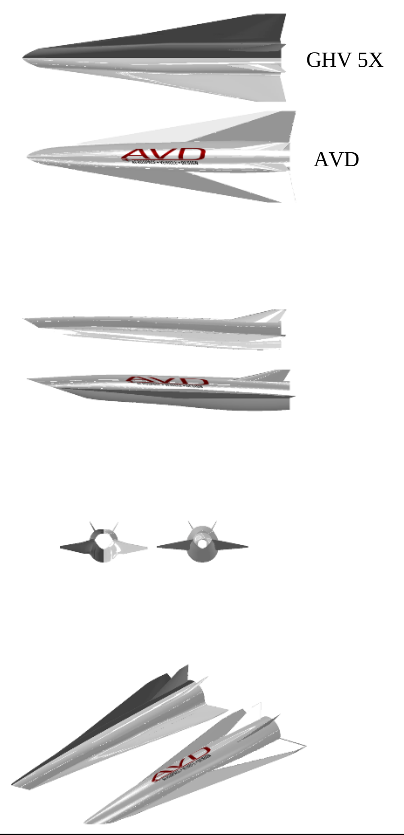


Figure 5-13 GHV Adaptation CMDS Execution Results

As with the previous case study, the first assessment to be made is the whether the GHV Adaptation CMDS is a full stand-alone analysis framework to ensure that AVD^{DBMS} is providing Syntactically Composable results. The resulting source code for the GHV Verification CMDS is able to run through and create output data with only modification of the CMDS input file.

The next question to be answered is whether the composed CMDS is semantically valid to model the GHV problem. In the case of the GHV Adaptation CMDS, the aerodynamic estimation method has been chosen from the listing resulting from the Matching step of the CMDS Composition Process. The intent with this deviation from the previous case has been to show the effect of method selection on the output CMDS. This has been described in the comparison of case study DSMs. In each case study both Aerodynamic methods (AERO_MD0008, and AERO_MD0009) were included in the Matched Disciplinary Method listing, see Table 5-1. In other words, the AVD^{DBMS} Matching step found both of these Disciplinary methods to be applicable to model the GHV problem. So although the choice of a different aerodynamic method has changed the integration level and overall source of the output CMDS, it should still show possess the capability to model the GHV to an acceptable level of accuracy. As Figure 5-13 shows, the GHV Adaptation CMDS is in general agreement with the GHV reference data. A more complete listing of result data can be found in Appendix D.

One main differences between each of the aerodynamics method, is the number of variables each respective method is a function of, and the subsequent level of integration of those variables. An important aspect of conceptual design is the ability to run trade studies and create solution spaces of design relevant input parameters. A solution space is a dashboard visualization of vehicle metrics in order aid in decision making. It is constructed by plotting resulting metrics of individually converged vehicle sized to a fixed mission requirements and varying vehicle parameters.

The aerodynamic method (AERO_MD0009) from the GHV Verification CMDS is a function of the engine mass flow rate, and other flight condition specific parameters. The engine mass flow scale is an output of Geometry Disciplinary Analysis and is a strong function of planform area. This means that AERO_MD0009 is only effected by changes in the vehicle planform area. If a trade study were to be conducted by changing any other geometry parameter, the aerodynamic results would not be effected.

The aerodynamic method (AERO_MD0008) from the GHV Adaptation CMDS is a function of multiple geometry parameters. This means that trade studies can be conducted by varying any of those parameters. As an example, AERO_MD0008 is a function of the K$\ddot{u}$chemann slenderness parameter (τ), where $\tau = \frac{V_{Total}}{S_{pln}^{1.5}}$. τ is a dimensionless parameter measuring the ratio of the vehicle total volume to planform area. This ratio gives an idea of the relative stoutness of the vehicle; for a given planform area vehicles with a low value of τ are more planform dominated, whereas larger values of tau describe vehicles that are more stout, see Figure 5-14. Referring back to the GHV Adaptation DSM, we can see that the aerodynamics, and weight and balance disciplinary analyses are a both a function of τ. This leads to the notion that the GHV Adaptation CMDS results will be a strong function of τ.

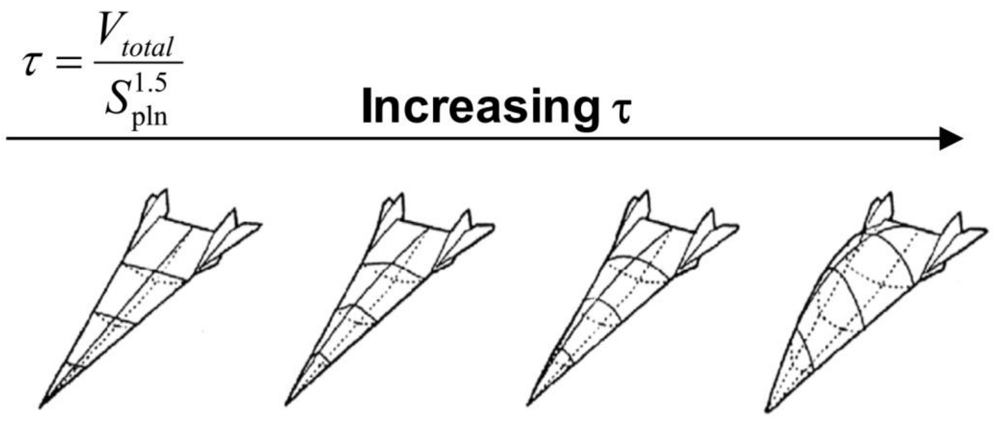
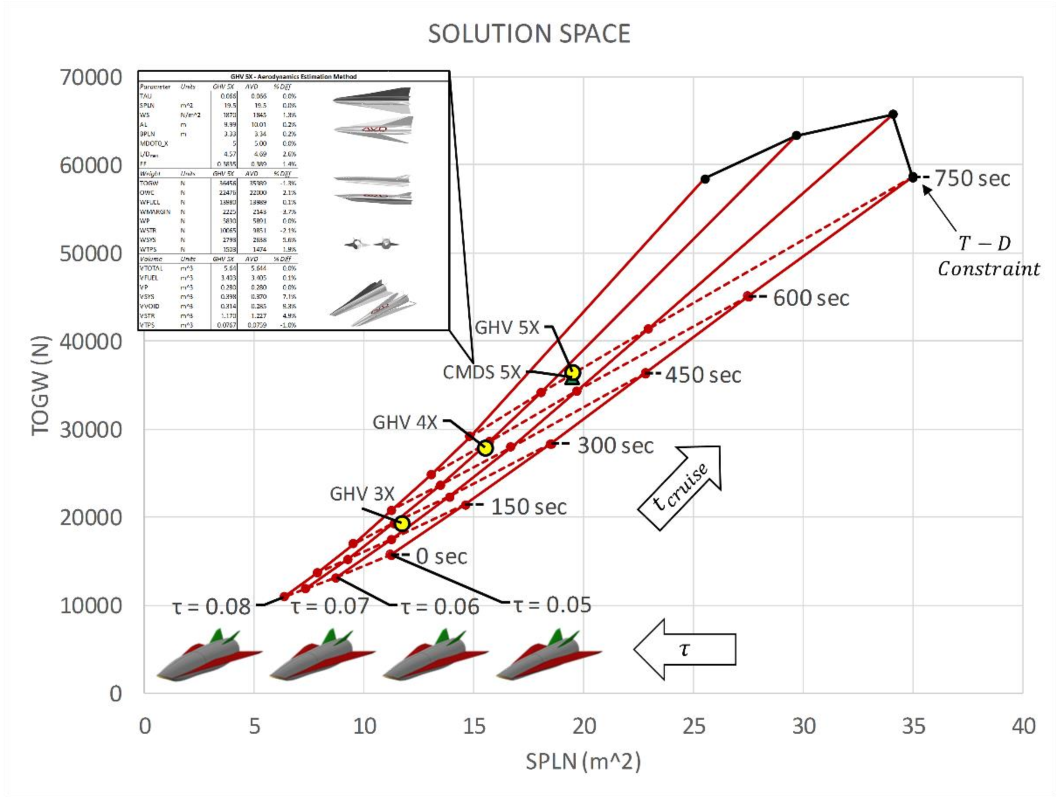


Figure 5-14 Explanation of Kuchemann slenderness parameter

Additionally, the previous CMDS Execution results have been focused on matching reference GHV data. This has led to the requirement that any inputs that effect the design mission of the vehicle be set to match those from the reference GHV data. One such parameter is the cruise endurance time after completing the 180° heading turn. This parameter directly effects the amount of time that the vehicle is at the design Mach number. An increase in the endurance cruise time would change the fuel fraction of the vehicle as it would require more fuel to complete the mission. This would then alter the convergence point where weight and volume required equals weight and volume available. Selecting this parameter for the solution space yields a solution space whose traded parameters are a function of both vehicle geometry and design mission.

Figure 5-15 is the solution space result showing the effect of varying both tau and endurance cruise time on the vehicle planform area and TOGW. The pop-up on the top left of the solution space shows results for single point on the solution space. This is meant to emphasize the fact that each point on the solution space is a vehicle that has been converged to meet the input mission requirements; each point a closed solution in terms of weight and volume required.



GHV 5X - Aerodynamics Estimation Method

Parameter	Units	GHV 5X	AVD	% Diff
TAU		0.066	0.066	0.0%
SPLN	m^2	19.5	19.5	0.0%
WS	N/m^2	1870	1845	1.3%
AL	m	9.99	10.01	0.2%
BPLN	m	3.33	3.34	0.2%
MDOT0_X		5	5.00	0.0%
L/D_{max}		4.57	4.69	2.0%
FF		0.3835	0.389	1.4%
Weight	*Units*	*GHV 5X*	*AVD*	*% Diff*
TOGW	N	36456	35980	-1.3%
OWE	N	22476	22000	2.1%
WFUEL	N	13980	13989	0.1%
WMARGIN	N	2225	2143	3.7%
WP	N	5890	5891	0.0%
WSTR	N	10065	9851	-2.1%
WSYS	N	2793	2638	5.6%
WTPS	N	1503	1474	1.9%
Volume	*Units*	*GHV 5X*	*AVD*	*% Diff*
VTOTAL	m^3	5.64	5.644	0.0%
VFUEL	m^3	3.403	3.405	0.1%
VP	m^3	0.280	0.280	0.0%
VSYS	m^3	0.398	0.370	7.1%
VVOID	m^3	0.314	0.285	9.3%
VSTR	m^3	1.170	1.227	4.9%
VTPS	m^3	0.0767	0.0759	-1.0%

Figure 5-15 GHV Solution Space – Planform Area vs. Gross Weight

The red lines on the solution space are the results of the design trade; solid lines represent line of constant τ whereas dashed line represent lines of constant endurance cruise time. A visual representation of the effect of τ on the GHV outer mold line can be seen along the bottom of the solution space.

The black solid line represents the “thrust minus drag” constraint line. For each value of tau, any increase in endurance cruise time creates a thrust requirement greater than the maximum capability of the vehicle. This constraint serves to cap our available solution space and provides a maximum for design mission capability in terms of endurance cruise time.

The yellow points represent reference GHV data points. A view of the position of the yellow points shows that they all follow a line of constant tau. The reported tau values for the vehicles listed ranges from $\tau = 0.065: 0.0675$. This means that the reference GHV vehicle has been scaled so that an increase in planform area does not yield a change in the outer mold line of the vehicle. The solution space achievable using the aerodynamic method from the GHV Verification CMDS would be limited to point along that line of constant τ. The green triangle slightly offset from the GHV 5X yellow reference point, is the result detailed in Figure 5-13.

5.3 Summary

The CMDS Composition Process (Chapter 3) and its software implementation AVD^{DBMS} (Chapter 4) present the capability to create tailor-made analysis frameworks. The GHV case studies provided an environment to test the ability of AVD^{DBMS} to both create the data relationships and write out a source code for a CMDS that is executable (Syntactically Composable), but also gives the user the ability to make choices throughout the CMDS Composition Process that aid to ensure Semantic Composability.

In the first case study a CMDS has been created using Disciplinary Methods based on tabulated reference data. The goal of this case study has been to show that given the same or similar system inputs, the composed CMDS can recreate similar vehicle performance results. The second case study has been an effort to show that selecting other Disciplinary Method options from the Matching step of the CMDS Composition Process can yield similarly agreeable performance results data, although the structure and integration level of the composed CMDS may differ.

Additionally, observation of the solution space created from the GHV Adaptation CMDS has shown a link between Disciplinary Method selection and the capability of the composed CMDS. The solution space trade study occurs in the CMDS Execution phase, after the CMDS has been composed. The availability of parameters to use in a trade study is a direct result of the integration effects of those trade study variables on the CMDS. As so, choices made in the Selection step of the CMDS Composition Process have an immense impact on what can be executed and observed once the CMDS has been composed. Once a CMDS has been created if it becomes apparent that different method should be selected to account for an unforeseen circumstance, it is necessary to create a new CMDS with the new method selected. The key benefit of the CMDS Composition process is the ability to quickly adjust to create new analysis frameworks as information about the given problem becomes available.

Chapter 6

Conclusions and Summary of Contributions

Resulting from a review of past and present aircraft synthesis codes, the breadth of the current research endeavor has been focused on the creation of a system that had 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. It was hypothesized that such a system would be required to have the following attributes:

- 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

With these specifications in mind, a methodology (CMDS Composition Process), and subsequent software implementation (AVD^{DBMS}) have been successfully created through the inclusion of techniques found in the fields of Systems Engineering, and Modelling & Simulation.

The functional analysis stage of the Systems Engineering Process has been applied to logically decompose a system into its constituent parts. This breakdown centers on the ability to take a given set of input requirements and define the hardware and function needed to fulfill those requirements. Functional analysis answers the question of “what” is needed to meet requirements, it does not attempt to answer the question of "how". Applying functional analysis to aircraft synthesis systems allows for system decomposition into three top-level building blocks: Product, Analysis Process, Disciplinary Methods. Using these building blocks, as well as their constituent subcategories (see Chapter 3), provides a mechanism to consistently and systematically decomposed aircraft synthesis systems.

In order to create analysis frameworks from the those building block, the field of Simulation Composability has been utilized. Simulation Composability is a Modelling and Simulations (M&S) concept describing the "capability to select and assemble simulation components in various combinations into valid simulation systems to satisfy specific user requirements" (Petty and Weisel 2003). The power of this type of system comes into the ability to re-use components previously built for other applications. The components are stored in a repository, where the choice of components and the order which they run are based on user need. There are two main types of composability:

- Syntactic Composability - Requires that the composable components be constructed so that their implementation details, such as parameter passing mechanisms, external data accesses, and timing assumptions are compatible for all of the different configurations that might be composed. The question in engineering (syntactic) composability is whether the components can be connected
- Semantic Composability - Addresses whether the models that make up the composed simulation system can be meaningfully composed, i.e., if their combined computation is semantically valid

A review of syntactically composable systems has highlighted several different mechanisms and techniques to ensure composability. A combination of these characteristics has been applied to create a syntactically composable framework for the automatic generation of a user defined CMDS, namely the CMDS Composition Process.

The CMDS Composition Process is meant to systematically evolve a user's problem description into an analysis framework meant to solve said problem. A brief description of each step is as follows:

> ***Matching****: The Matching phase queries and returns all disciplinary methods that are applicable to the problem requirements, namely the product and analytics process. The resulting list of disciplinary methods contains all of the attribute information for each method; see earlier discussion of disciplinary method building blocks.*
>
> ***Selecting****: The Selecting phase reviews all disciplinary methods returned from the Matching phase, and selects those that will be integrated into the CMDS. This step in the process is highly user-inclusive and is not meant to be done in an automated fashion. The engineer creating the CMDS selects the methods he/she feels best represent the problem they are trying to solve. That being said, the selection of disciplinary methods can be aided through the visualization of method specific information and the cross referencing of that information to the problem input requirements.*
>
> ***Arranging****: The Arranging phase assesses the combination of Product, Analysis Process and Selected Disciplinary Methods, and creates an integration blueprint for the DBMS. The integration blueprint is comprised of a Run Order for the selected Disciplinary Methods, and a listing of all variables input into and created by the DBMS.*
>
> ***Generation****: The Generation phase creates an analysis architecture based on the analysis blueprint created in the arranging phase. Up to this point every phase in the CMDS Generation process has been wholly contained in the DBMS setting. The Generation phase differs in this respect as its output is meant to be a self-contained executable, where the execution setting is not in the purview of the CMDS. There are two main components of the CMDS Generation phase: Input Parameter Listing and Analysis Architecture.*

As a mean of assessing the validity of the CMDS Composition process a prototype system (AVD^{DBMS}) has been developed. AVD^{DBMS} is comprised of three distinct layers: The Graphical User Interface (GUI), the database layer, and the analysis layer. AVD^{DBMS} has been applied to model the Generic Hypersonic Vehicle, an open source originating at the Air Force Research Laboratory. AVD^{DBMS} has been applied in three different ways in order to assess its validity: Verification using GHV disciplinary data, Validation using selected disciplinary analysis methods, and Application of the CMDS Composition Process to assess the design solution space for the GHV hardware.

In the first case study a CMDS has been created using Disciplinary Methods based on tabulated reference data. The second case study has been an effort to show that selecting other Disciplinary Method options from the Matching step of the CMDS Composition Process can yield similarly agreeable performance results data, although the structure and integration level of the composed CMDS may differ. Both case studies we shown to be both syntactically valid as well as semantically valid to model the GHV problem. Additionally, observation of the solution space created from the GHV Adaptation CMDS has shown a link between Disciplinary Method selection and the capability of the composed CMDS. The key benefit of the CMDS Composition process is the ability to quickly adjust to create new analysis frameworks as information about the given problem becomes available.

6.1 Summary of Contribution

- A generic methodology for the syntactic composition of aircraft synthesis systems
- A visual representation technique to assess the integration level of an aircraft synthesis system in terms of disciplinary analysis input variable requirements
- A mechanism to numerically evaluate the integration level of an aircraft synthesis system in terms of disciplinary analysis input variable requirements

6.2 Future Work

Several aspects of both the increase in capability of the CMDS Composition Process as well as its application are presented.

6.2.1 Architecture Creation and Evaluation

The current scope of AVD^{DBMS} follows a system of systems approach where the CMDS is the top-level. Each CMDS is based on a given Product, Analysis Process and a range of selected Disciplinary Methods. The application of this setting to the problem of system architecture design would entail the need to model several combinations of Product and Analysis Process. In order to do this using the current setting would require the user to keep track of each individual CMDS and manually record inputs and output results for each. Research into the topic of system architecture composability would include the generation of a CMDS for each constituent part of the Architecture as well as the resolution of results data from each CMDS to create holistic architecture results. An example of this would be the modelling of space launch systems, where the assessment would include the launch vehicle, ground systems and any in-space elements as well.

6.2.2 Method Selection

The Selecting step in the CMDS Composition Process has been intentionally designed to have a user in the loop. It is the authors opinion that having this step automated would reduce the user's ability to apply outside constraints and influences into their selection of disciplinary methods. That being said, the opportunity does arise to create a system which attempts to provide the user with as much information as possible in terms of applicable Disciplinary Methods. The inclusion of a methodology or system to rank/recommend one matched Disciplinary Method over another would be a boon for the CMDS Composition Process. This type of setting would also open the door into the inclusion of AI systems to provide analysis and assessments as to which Disciplinary Methods would be most appropriate for the given problem. The final selection would still be made by the user, but the amount of information he/she has to make that decision would increase by order of magnitude.

Appendix A

Listing of Aircraft Synthesis Systems

Table A-1 Aircraft Synthesis Systems (Chudoba 2001; Huang 2006; Coleman 2010)

Acronym	Full Name	Developer	Primary Application	Years
AAA	Advanced Airplane Analysis	DARcorporation	Aircraft	1991-
ACAD	Advanced Computer Aided Design	General Dynamics, Fort Worth	Aircraft	1993
ACAS	Advanced Counter Air Systems	US Army Aviation Systems Command	Air fighter	1987
ACDC	Aircraft Configuration Design Code	Boeing Defense and Space Group	Helicopter	1988-
ACDS	Parametric Preliminary Design System for Aircraft and Spacecraft Configuration	Northwestern Polytechnical University	Aircraft and AeroSpace Vehicle	1991-
ACES	Aircraft Configuration Expert System	Aeritalia	Aircraft	1989-
ACSYNT	AirCraft SYNThesis	NASA	Aircraft	1987-
ADAM	(-)	McDonnell Douglas	Aircraft	
ADAS	Aircraft Design and Analysis System	Delft University of Technology	Aircraft	1988-
ADROIT	Aircraft Design by Regulation Of Independent Tasks	Cranfield University	Aircraft	
ADST	Adaptable Design Synthesis Tool	General Dynamics/Fort Worth Division	Aircraft	1990
AGARD				1994
AIDA	Artificial Intelligence Supported Design of Aircraft	Delft University of Technology	Aircraft	1999
AircraftDesign	(-)	University of Osaka Prefecture	Aircraft	1990
APFEL	(-)	IABG	Aircraft	1979
Aprog	Auslegungs Programm	Dornier Luftfahrt	Aircraft	
ASAP	Aircraft Synthesis and Analysis Program	Vought Aeronautics Company	Fighter Aircraft	1974
ASCENT	(-)	Lockheed Martin Skunk Works	AeroSpace Vehicle	1993
ASSET	Advanced Systems Synthesis and Evaluation Technique	Lockheed California Company	Aircraft	Before 1993
Altman	Design Methodology for Low Speed High Altitude UAV's	Cranfield University	Unmanned Aerial Vehicles	Paper 1998
AVID	Aerospace Vehicle Interactive Design	N.C. State University, NASA LaRC	Aircraft and AeroSpace Vehicle	1992
AVSYN	?	Ryan Teledyne	?	1974
BEAM	(-)	Boeing	?	NA
CAAD	Computer-Aided Aircraft Design	SkyTech	High-Altitude Composite Aircraft	NA
CAAD	Computer-Aided Aircraft Design	Lockheed-Georgia Company	Aircraft	1968
CACTUS	(-)	Israel Aircraft Industries	Aircraft	NA
CADE	Conceptual Aircraft Design Environment	McDonnel Douglas Corporation	Fighter Aircraft (F-15)	1974
CAP	Configuration Analysis Program	North American Rockwell (B-1 Division)	Aircraft	1974
CAPDA	Computer Aided Preliminary Design of Aircraft	Technical University Berlin	Transonic Transport Aircraft	1984-
CAPS	Computer Aided Project Studies	BAC Military Aircraft Devision	Military Aircraft	1968
CASP	Combat Aircraft Synthesis Program	Northrop Corporation	Combat Aircraft	1980
CASDAT	Conceptual Aerospace Systems Design and Analysis Toolkit	Georgia Institute of Technology	Conceptual Aerospace Systems	late 1995
CASTOR	Commuter Aircraft Synthesis and Trajectory Optimization Routine	Loughborough University	Transonic Transport Aircraft	1986
CDS	Configuration Development System	Rockwell International	Aircraft and AeroSpace Vehicle	1976
CISE	(-)	Grumman Aerospace Corporation	AeroSpace Vehicle	1994
COMBAT	(-)	Cranfield University	Combat Aircraft	

CONSIZ	CONfiguration SIZing	NASA Langley Research Center	AeroSpace Vehicle	1993
CPDS	Computerized Preliminary Design System	The Boeing Company	Transonic Transport Aircraft	1972
Crispin	Aircraft sizing methodology	Loftin	Aircraft sizing methodology	1980
DesignSheet	(-)	Rockwell international	Aircraft and AeroSpace Vehicle	1992
DRAPO	Définition et Réalisation d'Avions Par Ordinateur	Avions Marcel Dassault/Bréguet Aviation	Aircraft	1968
DSP	Decision Support Problem	University of Houston	Aircraft	1987
EASIE	Environment for Application Software Integration and Execution	NASA Langley Research Center	Aircraft and AeroSpace Vehicle	1992
EADS				
ESCAPE	(-)	BAC (Commercial Aircraft Devision)	Aircraft	1995
ESP	Engineer's Scratch Pad	Lockheed Advanced Development Co.	Aircraft	1992
Expert Executive	(-)	The Boeing Company	?	
FASTER	Flexible Aircraft Scaling To Requirements	Florian Schieck		
FASTPASS	Flexible Analysis for Synthesis, Trajectory, and Performance for Advanced Space Systems	Lockheed Martin Astronautics	AeroSpace Vehicle	1996
FLOPS	FLight OPtimization System	NASA Langley Research Center	?	1980s-
FPDB & AS	Future Projects Data Banks & Application Systems	Airbus Industrie	Transonic Transport Aircraft	1995
FPDS	Future Projects Design System	Hawker Siddeley Aviation Ltd	Aircraft	1970
FRICTION	Skin friction and form drag code			1990
FVE	Flugzeug VorEntwurf	Stemme GmbH & Co. KG	GA Aircraft	1996
GASP	General Aviation Synthesis Program	NASA Ames Research Center	GA Aircraft	1978
GPAD	Graphics Program For Aircraft Design	Lockheed-Georgia Company	Aircraft	1975
HACDM	Hypersonic Aircraft Conceptual Design Methodology	Turin Polytechnic	Hypersonic aircraft	1994
HADO	Hypersonic Aircraft Design Optimization	Astrox	?	1987-
HASA	Hypersonic Aerospace Sizing Analysis	NASA Lewis Research Center	AeroSpace Vehicle	1985, 1990
HAVDAC	Hypersonic Astrox Vehicle Design and Analysis Code	Astrox		1987-
HCDV	Hypersonic Conceptual Vehicle Design	NASA Ames Research Center	Hypersonic Vehicles	
HESCOMP	HElicopter Sizing and Performance COMputer Program	Boeing Vertol Company	Helicopter	1973
HiSAIR/Pathfinder	High Speed Airframe Integration Research	Lockheed Engineering and Sciences Co.	Supersonic Commercial Transport Aircraft	1992
Holist	?	?	Hypersonic Vehicles with Airbreathing Propulsion	1992
ICAD	Interactive Computerized Aircraft Design	USAF-ASD	?	1974
ICADS	Interactive Computerized Aircraft Design System	Delft University of Technology	Aircraft	1996
IDAS	Integrated Design and Analysis System	Rockwell International Corporation	Fighter Aircraft	1986
IDEAS	Integrated DEsign Analysis System	Grumman Aerospace Corporation	Aircraft	1967
IKADE	Intelligent Knowledge Assisted Design Environment	Cranfield University	Aircraft	1992
IMAGE	Intelligent Multi-Disciplinary Aircraft Generation Environment	Georgia Tech	Supersonic Commercial Transport Aircraft	1998
IPAD	Integrated Programs for Aerospace-Vehicle Design	NASA Langley Research Center	AeroSpace Vehicle	1972-1980
IPPD	Integrated Product and Process Design	Georgia Tech	Aircraft, weapon system	1995

JET-UAV CONCEPTUAL DEISGN CODE		Northwestern Polytechnical University, China	Medium range JET-UAV	2000
LAGRANGE			Optimization	1993
LIDRAG	Span efficiency			1990
LOVELL				1970-1980
MAVRIS	an analysis-based environment	Georgia Institue of Technology		2000
MELLER		Daimler-Benz Aerospace Airbus	Civil aviation industry	1998
MacAirplane	(-)	Notre Dame University	Aircraft	1987
MIDAS	Multi-Disciplinary Integrated Design Analysis & Sizing	DaimlerChrysler Military	Aircraft	1996
MIDAS	Multi-Disciplinary Integration of Deutsche Airbus Specialists	DaimlerChrysler Aerospace Airbus	Supersonic Commercial Transport Aircraft	1996
MVA	Multi-Variate Analysis	RAE (BAC)	Aircraft	1991
MVO	MultiVariate Optimisation	RAE Farnborough	Aircraft	1973
NEURAL NETWORK FORMULATION	Optimization method for Aircrat Design	Georgia Institute of Technology	Aircraft	1998
ODIN	Optimal Design INtegration System	NASA Langley Research Center	AeroSpace Vehicle	1974
ONERA	Preliminary Design of Civil Transport Aircraft	Office National d'Etudes et de Recherches Aérospatiales	Subsonic Transport Aircraft	1989
OPDOT	Optimal Preliminary Design Of Transports	NASA Langley Research Center	Transonic Transport Aircraft	1970-1980
PACELAB	knowledge based software solutions	PACE	Aircraft	2000
Paper Airplane	(-)	MIT	Aircraft	
PASS	Program for Aircraft Synthesis Studies	Stanford University	Aircraft	1988
PATHFINDER		Lockheed Engineering and Sciences Co.	Supersonic Commercial Transport Aircraft	1992
PIANO	Project Interactive ANalysis and Optimisation	Lissys Limited	Transonic Transport Aircraft	1980-
POP	Parametrisches Optimierungs-Programm	Daimler-Benz Aerospace Airbus	Transonic Transport Aircraft	2000
PrADO	Preliminary Aircraft Design and Optimisation	Technical University Braunschweig	Aircraft and AeroSpace Vehicle	1986-
PreSST	Preliminary SuperSonic Transport Synthesis and Optimisation	DRA UK	Supersonic Commercial Transport Aircraft	
PROFET	(-)	IABG	Missile	1979
RAE	Artificial Intelligence Supported Design of Aircraft	Royal Aircraft Establishment, Farnborough	Aircraft conceptual design	Early1970's.
RAM		NASA	geometric modeling tool	1991
RCD	Rapid Conceptual Design	Lockheed Martin Skunk Works	AeroSpace Vehicle	
RDS	(-)	Conceptual Research Corporation	Aircraft	1992
RECIPE	(-)	?	?	1999
RSM	Response Surface Methodology			1998
Rubber Airplane	(-)	MIT	Aircraft	1960s-1970s
Schnieder				
Siegers	Numerical Synthesis Methodology for Combat Aircraft	Cranfield University	combat aircraft	Late 1970s
Spreadsheet Program	Spreadsheet Analysis Program	Loughborough University	Aircraft Design Studies	1995
SENSxx	(-)	DaimlerChrysler Aerospace Airbus	Transonic Transport Aircraft	
SIDE	System Integrated Design Environment	Astrox	?	1987-

SLAM	Simulated Langauge for Alternative Modeling	?	?	
Slate Architect	(-)	SDRC (Eds)	?	
SSP	System Synthesis Program	University of Maryland	Helicopter	
SSSP	Space Shuttle Synthesis Program	General Dynamics Corporation	AeroSpace Vehicle	
SYNAC	SYNthesis of AirCraft	General Dynamics	Aircraft	1967
TASOP	Transport Aircraft Synthesis and Optimisation Program	BAe (Commercial Aircraft) LTD	Transonic Transport Aircraft	
TIES	Technology Identification, Evaluation, and Selection	Georgia Institute of Technology		1998
TRANSYN	TRANsport SYNthesis	NASA Ames Research Center	Transonic Transport Aircraft	1963-(25years)
TRANSYS	TRANsportation SYStem	DLR (Aerospace Research)	AeroSpace Vehicle	1986-
TsAGI	Dialog System for Preliminary Design	TsAGI	Transonic Transport Aircraft	1975
VASCOMPII	V/STOL Aircraft Sizing and Performance Computer Program	Boeing Vertol CO.	V/STOL aircraft	1980
VDEP	Vehicle Design Evaluation Program	NASA Langley Research Center	Transonic Transport Aircraft	
VDI				
Vehicles	(-)	Aerospace Corporation	Space Systems	1988
VizCraft	(-)	Virginia Tech	Supersonic Commercial Transport Aircraft	1999
Voit-Nitschmann				
WIPAR	Waverider Interactive Parameter Adjustment Routine	DLR Braunschweig	AeroSpace Vehicle (Waverider)	
X-Pert	(-)	Delft University of Technology	Aircraft	Paper 1992

Appendix B

Methods Library Source Code

B.1 Aerodynamics

B.1.1 *AERO_MD0008*

```
%%%%%%%%% Pre-Allocate Outputs %%%%%%%%%

ALDMAX=repmat(NaN,size(AMACH));

ALIND=repmat(NaN,size(AMACH));

CD0=repmat(NaN,size(AMACH));

CLA=repmat(NaN,size(AMACH));

DCD_TDRAG = repmat(NaN,size(AMACH));

BETA = repmat(NaN,size(AMACH));

CL = repmat(NaN,size(AMACH));

CD = repmat(NaN,size(AMACH));

CA = repmat(NaN,size(AMACH));

CN = repmat(NaN,size(AMACH));

%%%%%% Regression Data %%%%%%%%%

AMACH_MAP=[2.0,                                   6.0,                                   12.0];

TAU_MAP=[0.01118,0.041569219,0.051822958,0.064,0.076367532,0.088772738,0.102486384,0.117575508,         ...

0.132574507,0.147369057,0.164316767,0.181019336,0.198252364,0.216,0.234247732, ...

0.252982213,0.272191109,0.29086856];

ALDMAX_MAP=[8.83,6.85,6.39,5.99,5.64,5.29,4.98,4.68,4.38,4.11,3.85,3.59,3.34,3.10,2.86,2.61,2.37,2.15;

8.50,6.32,5.90,5.53,5.19,4.87,4.59,4.31,4.07,3.80,3.56,3.35,3.12,2.89,2.68,2.46,2.26,2.06;

5.67,4.68,4.39,4.14,3.90,3.68,3.49,3.30,3.11,2.93,2.78,2.63,2.48,2.33,2.19,2.05,1.91,1.79];

D_B_MAP = [ 0.1, 0.15,0.25,0.5];

BetaCotLam_MAP = [0,0.132,0.25,0.368,0.487,0.633,0.727,1,1.5,2,2.5,3,3.5,4,4.5,5,5.5,6];

BetaCLA_MAP = [0,0.681,1.36,1.99,2.51,2.97,3.3,3.59,3.96,4.16,4.28,4.34,4.42,4.46,4.49,4.53,4.56,4.58;

0,0.681,1.36,1.99,2.51,2.97,3.3,3.64,4.04,4.29,4.48,4.61,4.75,4.84,4.96,5.02,5.09,5.16;

0,0.681,1.36,1.99,2.51,2.97,3.3,3.71,4.28,4.64,4.95,5.21,5.47,5.72,6.00,6.27,6.53,6.8;

0,0.681,1.36,1.99,2.51,2.97,3.3,3.97,4.99,5.67,6.21,6.71,7.25,7.73,8.31,8.88,9.46,10.03 ];

% SFORZA L/D Max Estimation

QBAR = Variable.DISCPROC.FLTCON.OUTPUT.QBAR;

T = Variable.DISCPROC.FLTCON.OUTPUT.T;

P = Variable.DISCPROC.FLTCON.OUTPUT.P;

RHO = Variable.DISCPROC.FLTCON.OUTPUT.RHO;
```

```
AMU = Variable.DISCPROC.FLTCON.OUTPUT.AMU;

RM = Variable.DISCPROC.FLTCON.OUTPUT.RM;

AKB = SFSPLN;

[KCF] = KCF_SFORZA(AKW, AKB, AL, TW_LIMIT, AMACH, QBAR, T, P, RHO, AMU, RM);

ALDMAX = (2/3).*KCF.^(-1/3);

ALDMAX = ALDMAX.*ALD_KFACT;

% CLA Calculations

D_B = DIA_BODY./BPLN;

BETA(AMACH<=1) = sqrt(1-AMACH(AMACH<=1).^2);

BETA(AMACH>1) = sqrt(AMACH(AMACH>1).^2-1);

CLA(AMACH <= 1) = CLAS;

CLA(AMACH >= 1.2) = pi./180.*interp2(BetaCotLam_MAP, D_B_MAP, BetaCLA_MAP, BETA(AMACH >=
1.2).*cotd(ALLE), repmat(D_B,size(BETA(AMACH >= 1.2))),'spline')./ BETA(AMACH >= 1.2);  % 1/degrees fig 4-18

BETA2 = sqrt(1.2^2-1);

CLA2 = pi./180*interp2(BetaCotLam_MAP, D_B_MAP, BetaCLA_MAP, BETA2*cotd(ALLE), D_B) ./ BETA2 ;

CLA(AMACH > 1.0 & AMACH < 1.2)= (CLA2(AMACH > 1.0 & AMACH < 1.2) - CLAS) ./ (1.2 - 1.0).*(CLA(AMACH > 1.0
& AMACH < 1.2) - 1) + CLAS;  %Linear interpolation M1 to M1.2

% ALIND CALC

LESP = zeros(size(AMACH));

AR = BPLN.^2./SPLN;

ALINDS = 1./(pi.*AR.*E_OS) + ALIND_ADD; % pg 4-30

INDEX = AMACH > 1 & BETA < 1./cotd(ALLE);

INDEX1 = AMACH > 1 & BETA > 1./cotd(ALLE);

LESP(INDEX)  = sqrt(1 - (BETA(INDEX).*cotd(ALLE)).^2); % pg 4-33 Leading Edge Suction Parameter

LESP(INDEX1) = 0;

ALIND(AMACH <= 1) = ALINDS;

ALIND(INDEX) = 1./(CLA(INDEX).*180./pi) - LESP(INDEX).* (1./(CLAS.*180./pi) - ALINDS); % pg 4-32

ALIND(INDEX1) = 1./(CLA(INDEX1).*180./pi) - LESP(INDEX1).* (1./(CLAS.*180./pi) - ALINDS); % pg 4-32

% CD0 Calc

SF=SPLN.*SFSPLN;

if (SF./(AL.^2) < 0.015)

    DCDT_MAX=(1.3862.*(SF./AL.^2)+0.067).*SFSPLN.*CDTW_COR; % fig 4-24

else
```

```
    DCDT_MAX=(0.9536*(SF./AL.^2).^3-1.916.*(SF./AL.^2).^2+1.3651.*(SF./AL.^2)+0.1119).*SFSPLN.*CDTW_COR;
%fig 4-25
end
DCD_TDRAG (AMACH <= 0.8 | AMACH > 1.2) = 0;
DCD_TDRAG (AMACH > 0.8 & AMACH <= 1.2) = (DCDT_MAX(AMACH > 0.8 & AMACH <= 1.2) - 0)./(1.2 -0.8).*
(AMACH(AMACH > 0.8 & AMACH <= 1.2) - 0.8);
DCD_TDRAG (AMACH > 1.2 & AMACH < 2.0) = (0 - DCDT_MAX(AMACH > 1.2 & AMACH < 2.0))./(2 - 1.2).*
(AMACH(AMACH > 1.2 & AMACH < 2.0) - 1.2) + DCDT_MAX(AMACH > 1.2 & AMACH < 2.0);
CD0 = 1.0./(4.0.*(ALIND).*ALDMAX.^2) + DCD_TDRAG;
CL = CLA.*(AOA-AOA_CL0);
CD = CD0 + ALIND.*CL.^2;
ALD = CL./CD;
CN = CL.*cosd(AOA) + CD.*sind(AOA);
CA = -CL.*sind(AOA) + CD.*cosd(AOA);
%% SubFunction
function [KCF] = KCF_SFORZA(AKW, AKB, AL, TW_LIMIT, AMACH, QBAR, T, P, RHO, AMU, RM)
RE_L = RM.*AMACH.*AL;
% Base Drag
AMACH_BD=[0.0, 0.8, 1.0, 3.0, 10.0];
CPB_A=[-0.11, -0.11, -0.20, -0.10, -0.014];
CPB=interp1(AMACH_BD,CPB_A,AMACH,'spline','extrap');
PBASE=P+QBAR.*CPB./P;
CDBASE=AKB.*(PBASE-P)./P;
% Sklin Friction Drag
T_SL = 288.2;        % T(K)
CP_SL = 1005;        % CP(J/(kg*K)
MU_SL = 1.46*1e-6.*(T_SL.^(3./2)./(T_SL+111));   % T(K), MU(N*sec/m^2)
k_SL = 1.99*1e-3.*(T_SL.^(3./2)./(T_SL+112));    % T(K), k(J/(sec*m*K))
Pr_SL = (MU_SL*CP_SL)/k_SL;
[CP, GAMMA, R, H] = Air_CEA(T, P);
FPRE_INPUT.T = T;
FPRE_INPUT.P = P;
FPRE_INPUT.RHO = RHO;
FPRE_INPUT.MU = AMU;

FPRE_INPUT.Pr_SL = Pr_SL;

FPRE_INPUT.CP_SL = CP_SL;

FPRE_INPUT.H = H;

FPRE_INPUT.AMACH = AMACH;

FPRE_INPUT.RE_L = RE_L;

FPRE_INPUT.TW_LIMIT = repmat(TW_LIMIT,size(T));

warning('off', 'NAG:warning')

T_STAR_GUESS = repmat(TW_LIMIT,size(T));

[T_STAR FPRE_INPUT] = runfsolve(@(T_STAR) FPRE_TURB(T_STAR,FPRE_INPUT),T_STAR_GUESS, 1e-1);

warning('off', 'NAG:warning')

CF = FPRE_INPUT.CF;

KCF = CF.*AKW + CDBASE.*AKB;

end

function [H_STAR_ERROR, FPRE_INPUT] = FPRE_TURB(T_STAR, FPRE_INPUT)

T = FPRE_INPUT.T;

P = FPRE_INPUT.P;

RHO = FPRE_INPUT.RHO;

MU = FPRE_INPUT.MU;

Pr_SL = FPRE_INPUT.Pr_SL;

CP_SL = FPRE_INPUT.CP_SL;

H = FPRE_INPUT.H;

AMACH = FPRE_INPUT.AMACH;

RE_L = FPRE_INPUT.RE_L;

TW_LIMIT = FPRE_INPUT.TW_LIMIT;

[CP_STAR, GAMMA_STAR, R_STAR, H_STAR] = Air_CEA(T_STAR, P);

RHO_STAR = P./(R_STAR.*T_STAR);

MU_STAR = 1.46*1e-6.*(T_STAR.^(3./2)./(T_STAR+111));   % T(K), MU(N*sec/m^2)

C_STAR = (RHO_STAR.*MU_STAR)./(RHO.*MU);

Pr_STAR = Pr_SL.*(CP_STAR./CP_SL);

r = (Pr_STAR).^(1/3);

TAW = T.*(1+r.*((GAMMA_STAR-1)./2).*AMACH.^2);

TW = TAW;
```

```
TW(TW > TW_LIMIT) = TW_LIMIT(TW > TW_LIMIT);

[CPW, GAMMAW, RW, HW] = Air_CEA(TW, P);

CF = (0.0266./RE_L.^0.139).*(C_STAR).^0.861.*(MU_STAR./MU).^-0.722;

H_STAR_NEW = H.*(0.5.*(1+HW./H)+(0.16.*r).*((GAMMA_STAR-1)./2).*AMACH.^2);

H_STAR_ERROR = (H_STAR-H_STAR_NEW);

FPRE_INPUT.CF = CF;

FPRE_INPUT.H_STAR_ERROR = H_STAR_ERROR;

% [T_STAR, TW, H_STAR, HW, H_STAR_ERROR]

end

function [CP, GAMMA, R, H] = Air_CEA(T, P)

CP = repmat(NaN,size(T));

GAMMA = repmat(NaN,size(T));

R = repmat(NaN,size(T));

H = repmat(NaN,size(T));

%%%%%% Regression Data %%%%%%

T_S = [200 500 1000 2000 3000];               % Kelvin

P_S = [0.0101325 1.0132 3.0397 5.0663 7.5994];    % BAR

[CP_S_MAP] = ...

              [1.0024        1.0295    1.1421    1.517     5.2532;

               1.0024        1.0295    1.141     1.3352    2.726;

               1.0024        1.0295    1.141     1.3267    2.2223;

               1.0024        1.0295    1.141     1.324     2.0569;

               1.0024        1.0295    1.141     1.3226    1.9668];  % kJ/(kg*K)

 [GAMMA_S_MAP] = ...

              [1.4013        1.3866    1.3357    1.2462    1.1474;

               1.4013        1.3866    1.3361    1.2754    1.1747;

               1.4013        1.3866    1.3361    1.277     1.1923;

               1.4013        1.3866    1.3361    1.2775    1.2002;

               1.4013        1.3866    1.3361    1.2777    1.2051];

[H_S_MAP] = ...

              [-102.8        200.47    743.65    1999.83   5818.37;

               -102.8        200.47    743.55    1976.02   3766.73;

               -102.8        200.47    743.55    1974.91   3601.53;
```

```
                -102.8          200.47     743.55     1974.57   3549.12;
                -102.8          200.47     743.55     1974.39   3520.94];    % kJ/kg

%%%%%% Create Interpolation Grid %%%%%%
[P_S_MAP, T_S_MAP] = ndgrid (P_S, T_S);
%%%%%% Create Interpolation Vectors %%%%%%
T_S_V = reshape(T_S_MAP,[numel(T_S_MAP),1]);
P_S_V = reshape(P_S_MAP,[numel(P_S_MAP),1]);
CP_S_V = reshape(CP_S_MAP,[numel(CP_S_MAP),1]);
GAMMA_S_V = reshape(GAMMA_S_MAP,[numel(GAMMA_S_MAP),1]);
H_S_V = reshape(H_S_MAP,[numel(H_S_MAP),1]);
%****************************************************
%** Convert P from Pa to BAR
%****************************************************
P = P.* 1e-5;
%****************************************************
%** Set Interpolation Boundaries
%****************************************************
INDEX = (T >= 200 & T <= 3000 & P >= 0.0101325 & P <= 7.5994);
%****************************************************
%** INTERPOLATE Specific Heat at Constant Pressure, CP
%****************************************************
 % Interpolate CP
[CP(INDEX), ifail] = runinterp([P_S_V, T_S_V], CP_S_V, [P(INDEX), T(INDEX)]);
% Convert CP From KJ/(kg*K) to J/(kg*K)
CP = CP.*1e3;
%****************************************************
%** INTERPOLATE Ratio of Specific Heat, GAMMA (CP/CV)
%****************************************************
[GAMMA(INDEX), ifail] = runinterp([P_S_V, T_S_V], GAMMA_S_V, [P(INDEX), T(INDEX)]);

%****************************************************
%** SOLVE for R
```

```
%****************************************************

R = CP.*(1-1./GAMMA);

%****************************************************

%** INTERPOLATE Enthalpy (kJ/kg)

%****************************************************

[H(INDEX), ifail] = runinterp([P_S_V, T_S_V], H_S_V, [P(INDEX), T(INDEX)]);

% Convert H From KJ/(kg) to J/(kg)

H = H.*1e3;

End
```

B.1.2 *AERO_MD0009*

```
%% Pre-Allocate Outputs
CN=repmat(NaN,size(AMACH));
CA=repmat(NaN,size(AMACH));
CL=repmat(NaN,size(AMACH));
CD=repmat(NaN,size(AMACH));
%% Set Up Input Interpolation Arrays
NP_AMACH = max(size(AMACH));
NP_AOA = max(size(AOA));
NP_MDOT0_X = max(size(MDOT0_X));
if prod([NP_AMACH, NP_AOA, NP_MDOT0_X]) > 1
if NP_AMACH == 1
AMACH = AMACH.*ones(max([NP_AMACH, NP_AOA, NP_MDOT0_X]));
end
if NP_AOA == 1
AOA = AOA.*ones(max([NP_AMACH, NP_AOA, NP_MDOT0_X]));
end
if NP_MDOT0_X == 1
MDOT0_X = MDOT0_X.*ones(max([NP_AMACH, NP_AOA, NP_MDOT0_X]));
end
end
%%%%%% Regression Data %%%%%%
AMACH_S = ...
[4.0 4.5 5.0 5.5 6.0 6.5 7.0];
MDOT0_X_S = ...
[1 2 3 4 5];
AOA_S = ...
[-4.0 -2.0 0.0 2.0 4.0 6.0 20];
[CN_S_MAP] = CN_S();
[CA_S_MAP] = CA_S();
%%%%%% Create Interpolation Grid %%%%%%
[AOA_S_MAP, AMACH_S_MAP, MDOT0_X_S_MAP] = ndgrid (AOA_S, AMACH_S, MDOT0_X_S);
AOA_S_VECT = reshape(AOA_S_MAP,numel(AOA_S_MAP),1);
AMACH_S_VECT = reshape(AMACH_S_MAP,numel(AMACH_S_MAP),1);
MDOT0_X_S_VECT = reshape(MDOT0_X_S_MAP,numel(MDOT0_X_S_MAP),1);
X_VECT = [AOA_S_VECT, AMACH_S_VECT, MDOT0_X_S_VECT];
CN_VECT = reshape(CN_S_MAP,numel(CN_S_MAP),1);
CA_VECT = reshape(CA_S_MAP,numel(CA_S_MAP),1);
%%%%%% Create Index %%%%%%
INDEX = (AMACH >= 3.9 & AMACH <= 7.0);
%****************************************************
%** INTERPOLATE Normal Force Coefficent
%****************************************************
CN(INDEX) = runinterp(X_VECT, CN_VECT, [AOA(INDEX), AMACH(INDEX), MDOT0_X(INDEX)]);

%****************************************************
```

```
%** INTERPOLATE Axial Force Coefficient
%****************************************************
CA(INDEX) = runinterp(X_VECT, CA_VECT, [AOA(INDEX), AMACH(INDEX), MDOT0_X(INDEX)]);
%****************************************************
%** Solve for Lift and Drag Coefficient
%****************************************************
CL = CN.*cosd(AOA) - CA.*sind(AOA);
CD = CN.*sind(AOA) + CA.*cosd(AOA);
%% SubFunction
function [CN_S_MAP] = CN_S()
CN_S_MAP(:,:,1) = ...
[-0.00901 -0.01065 -0.01262 -0.01224 -0.01201 -0.0119 -0.01187;
0.01634    0.0143     0.01235    0.00971    0.00713    0.00641    0.00579    ;
0.04224    0.03698    0.0323     0.02911    0.02623    0.02468    0.02331    ;
0.06932    0.07591    0.08337    0.06479    0.0462     0.0439     0.04193    ;
0.09981    0.08954    0.08088    0.07415    0.06822    0.06479    0.06184    ;
0.13075    0.12499    0.12179    0.10614    0.09226    0.08804    0.08447    ;
0.34733    0.37314    0.40816    0.33007    0.26054    0.25079    0.24288 ];
CN_S_MAP(:,:,2) = ...
[          -0.00896 -0.01074 -0.01283 -0.01244        -0.01221 -0.01211 -0.01208;
0.01727    0.01512    0.01307    0.01031    0.00761    0.00688    0.00621    ;
0.04405    0.03857    0.03368    0.03035    0.02734    0.02575    0.02434    ;
0.07207    0.079      0.08654    0.06744    0.0481     0.04572    0.04371    ;
0.10368    0.09304    0.08404    0.07708    0.0709     0.06747    0.0645     ;
0.13556    0.13003    0.12707    0.11074    0.09518    0.09096    0.08739    ;
0.35872    0.38896    0.42828    0.34636    0.26514    0.25539    0.24762 ];
CN_S_MAP(:,:,3) = ...
[          -0.00933 -0.01102 -0.01305 -0.01265        -0.01239 -0.01225 -0.01223;
0.01687    0.01473    0.01276    0.01003    0.00737    0.00667    0.00602    ;
0.0436     0.03818    0.03333    0.03003    0.02707    0.02549    0.02409    ;
0.07158    0.0784     0.08584    0.06692    0.04775    0.04541    0.04344    ;
0.10314    0.09254    0.08362    0.07669    0.07055    0.06712    0.06419    ;
0.13496    0.12942    0.12647    0.11022    0.09475    0.09058    0.08704    ;
0.3577     0.38758    0.42642    0.34493    0.26415    0.2548     0.24699 ];
CN_S_MAP(:,:,4) = ...
[          -0.00847 -0.01046 -0.01275 -0.01239        -0.01217 -0.0121 -0.01208;
0.01815    0.01589    0.01374    0.01078    0.00789    0.00715    0.00644    ;
0.04535    0.03963    0.0345     0.03104    0.0279     0.02625    0.02481    ;
0.07383    0.08163    0.08941    0.06957    0.04898    0.0466     0.04459    ;
0.10613    0.09522    0.08591    0.07873    0.07234    0.06886    0.06593    ;
0.13861    0.13354    0.13075    0.11382    0.09705    0.09289    0.08944    ;
0.36597    0.40178    0.44463    0.35945    0.27002    0.2611     0.25401 ];
CN_S_MAP(:,:,5) = ...
[          -0.00947 -0.01121 -0.01325 -0.01284        -0.01257 -0.01244 -0.01239;
0.01701    0.01486    0.01286    0.01011    0.00741    0.00673    0.00607    ;
0.04407    0.03855    0.03367    0.03034    0.02734    0.02574    0.02436    ;
0.07237    0.07932    0.08673    0.06769    0.0483     0.04597    0.04396    ;
0.10434    0.09368    0.08462    0.07765    0.07147    0.06806    0.06516    ;
0.13662    0.13089    0.12818    0.11185    0.09606    0.09194    0.08847    ;
0.36258    0.39136    0.4331     0.35125    0.26819    0.2591     0.25164 ];
end
function [CA_S_MAP] = CA_S()
CA_S_MAP(:,:,1) = ...
[          0.00881    0.0083     0.00811    0.00814    0.00821    0.00835    0.00855    ;
0.00989    0.00943    0.00921    0.00908    0.00908    0.00912    0.00923    ;
0.01118    0.01064    0.01035    0.0102     0.01016    0.01018    0.01027    ;
0.01276    0.01292    0.01479    0.01246    0.01154    0.01154    0.01165    ;
0.01463    0.01391    0.01349    0.01327    0.01326    0.01326    0.0134     ;
0.01708    0.01576    0.01558    0.01506    0.01578    0.01572    0.01584    ;
0.03423    0.02871    0.03021    0.02759    0.03342    0.03294    0.03292 ];
CA_S_MAP(:,:,2) = ...
[          0.00805    0.00754    0.00731    0.00729    0.00733    0.0074     0.00754    ;
0.00927    0.00877    0.00845    0.00825    0.00818    0.00816    0.00822    ;
0.01061    0.01001    0.00961    0.00941    0.00927    0.00924    0.00927    ;
0.01213    0.013      0.01534    0.01239    0.01064    0.0106     0.01066    ;
0.01389	0.01319	0.01269	0.01242	0.01224	0.01221	0.0123	;
0.01577	0.01545	0.01601	0.01474	0.01406	0.01404	0.01417	;
0.02893	0.03127	0.03925	0.03098	0.0268	0.02685	0.02726 ];
CA_S_MAP(:,:,3) = ...
[	0.00765	0.00713	0.00687	0.0068	0.00681	0.00686	0.00696	;
0.00885	0.00834	0.00799	0.00776	0.00765	0.00763	0.00762	;
0.01017	0.00957	0.00913	0.0089	0.00875	0.00869	0.00866	;
0.01167	0.01251	0.01479	0.01184	0.01011	0.01004	0.01004	;
0.0134	0.01269	0.01219	0.01186	0.01167	0.01163	0.01165	;
0.01526	0.01493	0.01547	0.01417	0.01346	0.01342	0.01346	;
0.02828	0.03061	0.03843	0.03034	0.02599	0.02595	0.02613 ];
CA_S_MAP(:,:,4) = ...
[	0.00754	0.00694	0.00662	0.00651	0.0065	0.00656	0.00663	;
0.00879	0.00826	0.00785	0.00753	0.00737	0.00732	0.00732	;
0.01016	0.0095	0.00898	0.00871	0.00851	0.00844	0.00841	;
0.01167	0.01336	0.01623	0.01248	0.0099	0.00988	0.00988	;
0.01351	0.01279	0.01222	0.01184	0.01156	0.01155	0.01159	;
0.0153	0.01562	0.01657	0.01475	0.0133	0.01336	0.01347	;
0.02783	0.03543	0.04702	0.03512	0.02548	0.02603	0.02663 ];
CA_S_MAP(:,:,5) = ...
[	0.00723	0.00671	0.00643	0.00633	0.00631	0.00634	0.00641	;
0.00847	0.00795	0.00757	0.00729	0.00715	0.00708	0.00706	;
0.00982	0.00919	0.00871	0.00845	0.00826	0.00817	0.00813	;
0.01131	0.01243	0.01483	0.01167	0.00962	0.00956	0.00957	;
0.01309	0.01239	0.01184	0.01148	0.01123	0.0112	0.01123	;
0.01488	0.01482	0.01559	0.01407	0.01296	0.01297	0.01305	;
0.02741	0.03183	0.04184	0.0322	0.02507	0.02536	0.02579 ];
end
```

B.2 Propulsion

B.2.1 *PROP_MD0006*

```
%% Pre-Allocate Outputs
AISP=repmat(NaN,size(THRL_VAR));
FT_AVAIL=repmat(NaN,size(THRL_VAR));
OF=repmat(NaN,size(THRL_VAR));
CFN=repmat(NaN,size(THRL_VAR));
%% Set Up Input Interpolation Arrays
PHI_FUEL = THRL_VAR.*PHI_FUEL_REF;
AOA_PROP = AOA-repmat(AOA_T,size(AOA));
%%%%%% Regression Data %%%%%%
AMACH_S = ...
[4.0 4.5 5.0 5.5 6.0 6.5 7.0];
PHI_S = ...
[0.0 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 10];
AOA_S = ...
[-4.0 -2.0 0.0 2.0 4.0 6.0 20];
[CFN_S_MAP] = CFN_S();
[AISP_S_MAP] = AISP_S();
%%%%%% Create Interpolation Grid %%%%%%
[AMACH_S_MAP, PHI_S_MAP, AOA_S_MAP] = ndgrid (AMACH_S, PHI_S, AOA_S);
AMACH_VECT = reshape(AMACH_S_MAP,[numel(AMACH_S_MAP),1]);
PHI_S_VECT = reshape(PHI_S_MAP,[numel(PHI_S_MAP),1]);
AOA_S_VECT = reshape(AOA_S_MAP,[numel(AOA_S_MAP),1]);
CFN_S_VECT = reshape(CFN_S_MAP,[numel(CFN_S_MAP),1]);
AISP_S_VECT = reshape(AISP_S_MAP,[numel(AISP_S_MAP),1]);
x = [AMACH_VECT, PHI_S_VECT, AOA_S_VECT];
%****************************************************
%** INTERPOLATE RAM/SCRAMJET THRUST
%****************************************************
A10_REF = 0.1799; % FROM GHV CONCEPTUAL DESIGN COMPLETE REPORT
INDEX = (AMACH >= 4.0 & AMACH <= 7.0 & PHI_FUEL <= 1.2);
```

```
px = [AMACH(INDEX), PHI_FUEL(INDEX), AOA_PROP(INDEX)];
[CFN(INDEX), ifail] = runinterp(x, CFN_S_VECT, px);
FT_AVAIL(INDEX) = CFN(INDEX).*QBAR(INDEX).*A10_REF.*MDOT0_X;
INDEX = (AMACH >= 4.0 & AMACH <= 7.0 & PHI_FUEL > 1.2);
CFN_EXTRAP = CFN;
PHI_FUEL_EXTRAP = PHI_FUEL;
PHI_FUEL_EXTRAP(INDEX) = 1.2;
px = [AMACH(INDEX), PHI_FUEL_EXTRAP(INDEX), AOA_PROP(INDEX)];
[CFN_EXTRAP(INDEX), ifail] = runinterp(x, CFN_S_VECT, px);
CFN(INDEX) = 0.2.*(PHI_FUEL(INDEX)-1.2) + CFN_EXTRAP(INDEX);
FT_AVAIL(INDEX) = CFN(INDEX).*QBAR(INDEX).*A10_REF.*MDOT0_X;
%*****************************************************
%** INTERPOLATE ISP
%*****************************************************
INDEX = (AMACH >= 4.0 & AMACH <= 7.0 & PHI_FUEL <= 1.2);
px = [AMACH(INDEX), PHI_FUEL(INDEX), AOA_PROP(INDEX)];
[AISP(INDEX), ifail] = runinterp(x, AISP_S_VECT, px);
INDEX = (AMACH >= 4.0 & AMACH <= 7.0 & PHI_FUEL > 1.2);
PHI_FUEL_EXTRAP = PHI_FUEL;
PHI_FUEL_EXTRAP(INDEX) = 1.2;
px = [AMACH(INDEX), PHI_FUEL_EXTRAP(INDEX), AOA_PROP(INDEX)];
[AISP(INDEX), ifail] = runinterp(x, AISP_S_VECT, px);
OF(~isnan(AISP)) = 0;
%% SubFunction
function [CFN_S_MAP] = CFN_S()
CFN_S_MAP(:,:,1) = ...
[0.000   0.217    0.2477   0.2768   0.2981   0.3145   0.332    0.3226   0.3224   1.907    ;
0.000    0.1859   0.2146   0.2402   0.2646   0.2882   0.311    0.3036   0.3027   2.363    ;
0.000    0.1582   0.1847   0.21     0.2342   0.2563   0.2767   0.2705   0.2699   2.1127   ;
0.000    0.1337   0.1575   0.1801   0.2019   0.223    0.2435   0.2385   0.2381   2.0885   ;
0.000    0.1147   0.136    0.1565   0.1762   0.1955   0.2142   0.21     0.2098   1.8972   ;
0.000    0.0991   0.1186   0.1372   0.1553   0.1729   0.1901   0.1864   0.1863   1.7381   ;
0.000    0.0865   0.1042   0.1213   0.1379   0.1541   0.17     0.1671   0.1673   1.601    ];
CFN_S_MAP(:,:,2) = ...
[0.000   0.2371   0.2711   0.3012   0.1722   0.3379   0.358    0.3477   0.3474   2.167    ;
0.000    0.2067   0.2372   0.2659   0.2931   0.3193   0.345    0.3363   0.3356   2.658    ;
0.000    0.1778   0.2077   0.2362   0.2633   0.287    0.3099   0.3029   0.3022   2.3709   ;
0.000    0.1509   0.1777   0.2034   0.2281   0.2519   0.2751   0.2694   0.2688   2.3631   ;
0.000    0.1298   0.1541   0.1774   0.1998   0.2216   0.2429   0.2381   0.2379   2.1599   ;
0.000    0.1104   0.1323   0.1534   0.1738   0.1937   0.2132   0.2093   0.2092   1.9682   ;
0.000    0.0947   0.1146   0.1338   0.1525   0.1708   0.1887   0.1855   0.1857   1.7997   ];
CFN_S_MAP(:,:,3) = ...
[0.000   0.2588   0.2959   0.3198   0.3405   0.3621   0.3852   0.374    0.3735   2.4642   ;
0.000    0.2269   0.2602   0.2917   0.3217   0.3507   0.3788   0.3696   0.3683   2.9078   ;
0.000    0.197    0.2303   0.2619   0.2905   0.3167   0.3422   0.3345   0.3337   2.6372   ;
0.000    0.1684   0.1984   0.227    0.2546   0.2813   0.3071   0.3008   0.3002   2.6291   ;
0.000    0.1462   0.1735   0.1996   0.2248   0.2492   0.2731   0.2678   0.2675   2.4241   ;
0.000    0.123    0.1475   0.1711   0.1939   0.2162   0.238    0.2337   0.2336   2.2      ;
0.000    0.1045   0.1267   0.1482   0.169    0.1894   0.2094   0.2059   0.2061   2.0094   ];
CFN_S_MAP(:,:,4) = ...
[0.000   0.2805   0.3071   0.3369   0.3606   0.3855   0.4118   0.3999   0.399    2.7788   ;
0.000    0.2528   0.2892   0.3237   0.3565   0.3882   0.4187   0.4053   0.4054   3.1637   ;
0.000    0.2278   0.2645   0.2993   0.3292   0.3581   0.3864   0.3776   0.3768   2.9334   ;
0.000    0.1927   0.2259   0.2577   0.2882   0.3175   0.3461   0.339    0.3385   2.9201   ;
0.000    0.1645   0.1947   0.2237   0.2518   0.2789   0.3054   0.2996   0.2993   2.6904   ;
0.000    0.1379   0.165    0.1911   0.2164   0.2411   0.2652   0.2605   0.2604   2.4342   ;
0.000    0.1171   0.1416   0.1652   0.1881   0.2106   0.2326   0.2289   0.2291   2.2126   ];
CFN_S_MAP(:,:,5) = ...
[0.000   0.3213   0.3463   0.3727   0.4003   0.4294   0.4601   0.4474   0.4462   3.2231   ;
0.000    0.2792   0.3187   0.356    0.3918   0.4263   0.4481   0.4273   0.4338   2.4101   ;
0.000    0.2397   0.2794   0.315    0.3477   0.3792   0.4098   0.4005   0.3996   3.1638   ;
0.000    0.2067   0.2428   0.2774   0.3105   0.3426   0.3718   0.3638   0.3631   2.9998   ;
0.000    0.1815   0.2144   0.2461   0.2766   0.3061   0.335    0.3288   0.3285   2.936    ;
0.000    0.1514   0.1809   0.2093   0.237    0.2636   0.2899   0.285    0.285    2.6569   ;
0.000    0.1281   0.1548   0.1805   0.2054   0.23     0.2539   0.2501   0.2504   2.4049   ];
```

```
CFN_S_MAP(:,:,6) = ...
[0.000   0.3276  0.3618  0.3927  0.4264  0.46    0.4769  0.4723  0.4661  1.9979  ;
0.000    0.3064  0.3488  0.3894  0.4279  0.4553  0.3022  0.464   0.4617  2.926   ;
0.000    0.268   0.3109  0.3479  0.3831  0.4174  0.4508  0.4406  0.4396  3.4568  ;
0.000    0.2309  0.2701  0.3077  0.3435  0.3781  0.4086  0.4000  0.3993  3.1536  ;
0.000    0.202   0.238   0.2725  0.3057  0.3379  0.3695  0.3628  0.3624  3.2135  ;
0.000    0.169   0.2011  0.2321  0.262   0.2912  0.3197  0.3144  0.3143  2.8847  ;
0.000    0.1438  0.1728  0.2007  0.228   0.2545  0.2804  0.2764  0.2767  2.6114  ];
CFN_S_MAP(:,:,7) = ...
[0.000   0.3276  0.3618  0.3927  0.4264  0.46    0.4769  0.4723  0.4661  1.9979  ;
0.000    0.3064  0.3488  0.3894  0.4279  0.4553  0.3022  0.464   0.4617  2.926   ;
0.000    0.268   0.3109  0.3479  0.3831  0.4174  0.4508  0.4406  0.4396  3.4568  ;
0.000    0.2309  0.2701  0.3077  0.3435  0.3781  0.4086  0.4000  0.3993  3.1536  ;
0.000    0.202   0.238   0.2725  0.3057  0.3379  0.3695  0.3628  0.3624  3.2135  ;
0.000    0.169   0.2011  0.2321  0.262   0.2912  0.3197  0.3144  0.3143  2.8847  ;
0.000    0.1438  0.1728  0.2007  0.228   0.2545  0.2804  0.2764  0.2767  2.6114  ];
end
function [AISP_S_MAP] = AISP_S()
AISP_S_MAP(:,:,1) = ...
[0.000   1805.5  1717.6  1645.4  1532.9  1438.5  1367.5  1203.9  1102.8  1102.8  ;
0.000    1679.3  1615.7  1550    1493.9  1446.2  1404.3  1242.1  1135.2  1135.2  ;
0.000    1537    1496    1457.7  1423    1383.9  1344.5  1191.1  1089.3  1089.3  ;
0.000    1380.5  1354.5  1328.4  1303.3  1279.5  1257.3  1115.9  1020.8  1020.8  ;
0.000    1248.7  1234    1217.2  1199.7  1182.9  1166.5  1036.4  949.1   949.1   ;
0.000    1130.0  1126.5  1117.5  1106.4  1095.4  1084.2  964.0   883.6   883.6   ;
0.000    1027.7  1032.3  1029.9  1024.7  1018.4  1011    900.8   826.7   826.7   ];
AISP_S_MAP(:,:,2) = ...
[0.000   1787.8  1703.1  978.9   1485.7  1400.3  1336    1175.8  1076.8  1076.8  ;
0.000    1676.2  1603.1  1540.2  1485.1  1437.7  1397.7  1235    1129.4  1129.4  ;
0.000    1536.6  1496.1  1458.2  1422.7  1378.3  1339.4  1186.5  1084.9  1084.9  ;
0.000    1378.4  1353.3  1327.6  1302.9  1279.3  1257.6  1116    1020.8  1020.8  ;
0.000    1245.2  1231.9  1215.9  1198.8  1182.1  1166.2  1036.3  948.8   948.8   ;
0.000    1114.5  1113.6  1106.7  1097.3  1087.4  1077.2  958.4   878.3   878.3   ;
0.000    1002    1010.1  1011.4  1008.9  1004.4  998.8   889.9   816.7   816.7   ];
AISP_S_MAP(:,:,3) = ...
[0.000   1781.6  1698.7  1554.4  1449    1370.4  1312.9  1155.2  1057.4  1057.4  ;
0.000    1668.7  1594.5  1531.6  1477.8  1432    1392.4  1231.4  1124.6  1124.6  ;
0.000    1533.4  1493.5  1456.3  1413.7  1369.6  1332    1179.7  1078.9  1078.9  ;
0.000    1380.5  1355.3  1329.5  1304.7  1281.4  1259.3  1117.9  1022.6  1022.6  ;
0.000    1254.2  1239.9  1222.9  1205.2  1187.8  1172    1041.2  953.4   953.4   ;
0.000    1113.1  1112.6  1106.2  1097.5  1087.5  1077.6  959.1   878.9   878.9   ;
0.000    992.7   1003.4  1006    1004    1000.2  995.3   887.5   814.4   814.4   ];
AISP_S_MAP(:,:,4) = ...
[0.000   1778.4  1639    1508.4  1413.9  1344.2  1292.9  1137.6  1040.7  1040.7  ;
0.000    1699.1  1619.2  1553.3  1496.6  1448.5  736.1   1219.6  1118.1  1118.1  ;
0.000    1610.3  1558.3  1511.4  1454.6  1406.4  1365.6  1209.2  1105.9  1105.9  ;
0.000    1428    1395.2  1364.7  1335.3  1308.1  1283.5  1139.1  1042.6  1042.6  ;
0.000    1270.7  1253.7  1234.7  1216.4  1197.7  1180.3  1049.3  960.7   960.7   ;
0.000    1129.4  1126.9  1118.4  1108.5  1098.3  1086.9  967.8   887.1   887.1   ;
0.000    1012    1019.5  1019.8  1016.5  1011.5  1005.4  897.1   823.3   823.3   ];
AISP_S_MAP(:,:,5) = ...
[0.000   1861.3  1673.5  1545.2  1453.1  1386.2  1337.5  1178.5  1077.4  1077.4  ;
0.000    1730.4  1645.6  1575    1516.6  1452.4  1372.5  655.9   1096.6  1096.6  ;
0.000    1557.9  1513.9  1462.8  1412.7  1369.5  1331.8  1179.5  1078.6  1078.6  ;
0.000    1407.9  1378.9  1350.1  1322.7  1297.3  1267.2  1123.5  1028    1028    ;
0.000    1288.3  1269    1248.4  1227.5  1207.9  1189.7  1058.2  969     969     ;
0.000    1141    1136.3  1127.1  1116.7  1104.5  1092.9  973.7   892.5   892.5   ;
0.000    1018.7  1025.6  1025.5  1021.6  1016.9  1010.3  902.2   827.8   827.8   ];
AISP_S_MAP(:,:,6) = ...
[0.000   1790.9  1626.1  1513.8  1435.5  1380.7  1288.3  1153.9  1045.3  1045.3  ;
0.000    1757.6  1667.6  1595.3  1534.8  1431.3  1359.3  1192.7  1093    1093    ;
0.000    1606.8  1553.6  1490.2  1435.9  1390.2  1351    1196.8  1094.5  1094.5  ;
0.000    1446.7  1410.6  1377.6  1346    1317    1280.7  1136.2  1039.9  1039.9  ;
0.000    1315.4  1291.8  1267.6  1244.7  1222.7  1203.7  1071.1  980.7   980.7   ;
0.000    1170.7  1161.3  1149    1135.1  1121.4  1108.3  987.9   905.5   905.5   ;
0.000	1054	1055.5	1051.2	1044.8	1036.9	1028.5	918.9	843.4	843.4	];
AISP_S_MAP(:,:,7) = ...
[0.000	1790.9	1626.1	1513.8	1435.5	1380.7	1288.3	1153.9	1045.3	1045.3	;
0.000	1757.6	1667.6	1595.3	1534.8	1431.3	1359.3	1192.7	1093	1093	;
0.000	1606.8	1553.6	1490.2	1435.9	1390.2	1351	1196.8	1094.5	1094.5	;
0.000	1446.7	1410.6	1377.6	1346	1317	1280.7	1136.2	1039.9	1039.9	;
0.000	1315.4	1291.8	1267.6	1244.7	1222.7	1203.7	1071.1	980.7	980.7	;
0.000	1170.7	1161.3	1149	1135.1	1121.4	1108.3	987.9	905.5	905.5	;
0.000	1054	1055.5	1051.2	1044.8	1036.9	1028.5	918.9	843.4	843.4	];
end
```

B.3 Performance Matching

B.3.1 *PM_MD0003*

```
% PREALLOCATE VECTORS
AISP_EFF_V		= zeros(TRAJ_NSTEP,1);
AISP_V	= zeros(TRAJ_NSTEP,1);
AISP_V_HW		= zeros(TRAJ_NSTEP,length(VEHICLE_HW));
ALD_V	= zeros(TRAJ_NSTEP,1);
AMACH_V		= zeros(TRAJ_NSTEP,1);
AN_V	= zeros(TRAJ_NSTEP,1);
AOA_V	= zeros(TRAJ_NSTEP,1);
CD_V	= zeros(TRAJ_NSTEP,1);
CD_V_HW		= zeros(TRAJ_NSTEP,length(VEHICLE_HW));
CL_V	= zeros(TRAJ_NSTEP,1);
CL_V_HW		= zeros(TRAJ_NSTEP,length(VEHICLE_HW));
D_V	= zeros(TRAJ_NSTEP,1);
DGAM_V = zeros(TRAJ_NSTEP,1);
DPSI_V	= zeros(TRAJ_NSTEP,1);
DR_V	= zeros(TRAJ_NSTEP,1);
DT_V	= zeros(TRAJ_NSTEP,1);
DUCT_PRESSURE_V = zeros(TRAJ_NSTEP,1);
DW_V	= zeros(TRAJ_NSTEP,1);
DWF_V	= zeros(TRAJ_NSTEP,1);
DWO_V	= zeros(TRAJ_NSTEP,1);
DX_V	= zeros(TRAJ_NSTEP,1);
DY_V	= zeros(TRAJ_NSTEP,1);
EDOT_V	= zeros(TRAJ_NSTEP,1);
EI_V	= zeros(TRAJ_NSTEP,1);
FT_AVAIL_MAX_V = zeros(TRAJ_NSTEP,1);
FT_V	= zeros(TRAJ_NSTEP,1);
%FT_V_HW		= zeros(TRAJ_NSTEP,length(VEHICLE_HW));
G_V	= zeros(TRAJ_NSTEP,1);
GAMDOT_V		= zeros(TRAJ_NSTEP,1);
L_V	= zeros(TRAJ_NSTEP,1);
OF_V	= zeros(TRAJ_NSTEP,1);
OF_V_HW		= zeros(TRAJ_NSTEP,length(VEHICLE_HW));
PSIDOT_V		= zeros(TRAJ_NSTEP,1);
QBAR_V	= zeros(TRAJ_NSTEP,1);
SELECTED_V_FUNCMODE	= cell(TRAJ_NSTEP,length(VEHICLE_FUNCTION));
SIGMA_V= zeros(TRAJ_NSTEP,1);
W_V = zeros(TRAJ_NSTEP,1);
%INITIAL POINTS FROM TRAJECTORY
I = max(I_V);
% CALCULTE CHANGE IN AMACH PER STEP
ALT_START = ALT_V(I);
V_START = V_V(I);
%%%%%%%%% Analysis %%%%%%
FLTCOND = fltcon(ALT_START,0,V_START,0);
Q_CONST = FLTCOND.QBAR;
% ASSIGN CONTROL VARIABLES TO traj
AN_MAX = TRAJ_AN_MAX;
AN_MIN = TRAJ_AN_MIN;
```

```
% CALCULTE CHANGE IN ALT PER STEP
DALT = (TRAJ_ALT_END - ALT_START)/(TRAJ_NSTEP);
% ITERATE FOR EACH ENERGY LEVEL
for CT = 1:TRAJ_NSTEP %ending at E(N+1) in order to store derivatives for E(N)
        %INPUTS FROM TRAJECTORY
        I = max(I_V);
        WR = WR_V(I);
        GAM = GAM_V(I)*DTR;
        PSI = PSI_V(I)*DTR;
        V = V_V(I);%sqrt((V_V(I+1).^2 + V_V(I).^2)/2);
        ALT = ALT_V(I);%(ALT_V(I+1) + ALT_V(I))/2;
        %ANALYSIS
        FLTCOND = fltcon(ALT,0,V,0);
        QBAR=FLTCOND.QBAR; AMACH=FLTCOND.AMACH;
        W = WS*SPLN/WR;
        G = G0./(1 + ALT./RE).^2;
        GAMDOT = 0;
        PSIDOT = 0;
        SIGMA = 0;
        L = W./G0 .* (G - V.^2./(RE + ALT));
        CL_REQ = L./(QBAR*SPLN);
        VAR_IN.ALT = ALT;
        VAR_IN.V = V;
        [AOA_OUT, VAR_IN] = runsolver(@AOAFUNC, 0, VAR_IN, 1, 1e-2);
        AOA = AOA_OUT;
        D = QBAR.*CD*SPLN;
        ALD = L/D;
        THRL_VAR = 1;
        FT_AVAIL_MAX = FT_AVAIL;
        FT_MAX_LIM = W*(AN_MAX + D/W + G/G0*sin(GAM)); % Thrust requirement for max acceleration
        if(FT_AVAIL_MAX > FT_MAX_LIM)
                AN = AN_MAX;
                FT = FT_MAX_LIM;
        else
                FT = FT_AVAIL_MAX;
                AN = FT./W - D./W - G./G0.*sin(GAM);
        end
        if (AN < AN_MIN & INSUFF_THRUST_CHECK == 'Y')
                disp('         I        ALT          V         GAM          W         AN   FT/W         D/W    G./G0.*sin(GAM)')
                disp([I ALT V GAM/DTR W AN FT/W D/W  G./G0.*sin(GAM)])
                error('INSUFFICIENT THRUST')
        elseif AN < AN_MIN
                AN = AN_MIN;
                FT = W*(AN_MIN + D/W + G/G0*sin(GAM));
        end
        VAR_IN.AOA = AOA;
        VAR_IN.ALT = ALT;
        VAR_IN.V = V;
        VAR_IN.FT = FT;
        [THRL_VAR_OUT, VAR_IN] = runsolver(@THRL_VARFUNC, 1, VAR_IN, 1, 1e-3);
        THRL_VAR = THRL_VAR_OUT; % Change in THRL_VAR triggers code to call FT function
        FT = FT_AVAIL;
        AISP = AISP;
        OF = OF;
        DUCT_PRESSURE = DUCT_PRESSURE;
        THRL_VAR_HW = zeros(size(FT_AVAIL_HW));
        THRL_VAR_HW(FT_AVAIL_HW~=0) = THRL_VAR;
        AISP_EFF = (FT - D - W*sin(GAM))/(FT/AISP);
        FLTCOND = fltcon(ALT+DALT,0,0,0);
        RHO_NEXT = FLTCOND.RHO;
        ALT_NEXT = ALT + DALT;
        V_NEXT = sqrt(2*Q_CONST/RHO_NEXT);
        E0 = ALT_V(I)*RE/(RE+ ALT_V(I)) + V_V(I)^2/(2*G0);
        EI = ALT_NEXT*RE/(RE+ALT_NEXT) + V_NEXT^2/(2*G0);
```

```
% CALCULATE DELTAS TO GET TO CURRENT POINT
RDOT = V*cos(GAM)*RE/(RE+ALT);
EDOT = V*AN;%sqrt((V^2+V_V(I+1)^2)/2)*AN;
DT = (EI-E0)/EDOT;
DW = - (FT/AISP)*DT;
DWF = - DW/(1+OF);
DWO = OF*DWF;
WRNEXT = 1/(1/WR + DW/(WS*SPLN));
DR = RDOT*DT;
DX = DR*cos(PSI);
DY = DR*sin(PSI);
DPSI = 0;
DGAM = GAMDOT*DT;

% ASSIGN VALUES AT CURRENT POINT
I_V(I+1,1) = I+1;
ALT_V(I+1,1) = ALT + DALT;
FF_V(I+1,1) = FF_V(I) + DWF/(WS*SPLN);
GAM_V(I+1,1) = (GAM+DGAM)/DTR;
PSI_V(I+1,1) = (PSI+DPSI)/DTR;
TIME_V(I+1,1) = TIME_V(I)+DT;
RANGE_V(I+1,1) = RANGE_V(I)+DR;
V_V(I+1,1) = sqrt(2*Q_CONST/RHO_NEXT);
WR_V(I+1,1) = WRNEXT;
X_V(I+1,1) = X_V(I) + DX;
Y_V(I+1,1) = Y_V(I) + DY;
FT_V_HW(I+1,:)  = FT_AVAIL_HW;
DUCT_PRESSURE_V_HW(I+1,:)  = DUCT_PRESSURE_HW;
THRL_VAR_REQ_V_HW(I+1,:)  = THRL_VAR_HW;
TRAJSEG_V(I+1,1)  = METHOD_TRAJSEG;

RANGE = RANGE_V(I+1,1);
X_RANGE = X_V(I+1,1);
ENDURANCE = TIME_V(I+1,1);
WR = WR_V(I+1,1);
FF = FF_V(I+1,1);
FT_MAX_HW =  max(FT_V_HW,[],1);
DUCT_PRESSURE_MAX_HW = max(DUCT_PRESSURE_V_HW,[],1);
THRL_VAR_MAX = max(max(THRL_VAR_REQ_V_HW,[],1));

AISP_EFF_V(CT,1) = AISP_EFF;
AISP_V(CT,1) = AISP;
AISP_V_HW(CT,:) = AISP_HW;
ALD_V(CT,1)  = ALD;
AMACH_V(CT,1)  = AMACH;
AN_V(CT,1)  = AN;
AOA_V(CT,1)  = AOA;
CD_V(CT,1)  = CD;
CD_V_HW(CT,:)  = CD_HW;
CL_V(CT,1)  = CL;
CL_V_HW(CT,:)  = CL_HW;
D_V(CT,1) = D;
DGAM_V(CT,1)  = DGAM/DTR;
DPSI_V(CT,1) = DPSI/DTR;
DR_V(CT,1)  = DR;
DT_V(CT,1)  = DT;
DUCT_PRESSURE_V(CT,1)  = DUCT_PRESSURE;
%DUCT_PRESSURE_V_HW(CT,1)  = DUCT_PRESSURE_HW;
DW_V(CT,1)  = DW;
DWF_V(CT,1)  = DWF;
DWO_V(CT,1)  = DWO;
DX_V(CT,1) = DX;
DY_V(CT,1) = DY;
EDOT_V(CT,1)  = EDOT;
```

```
        EI_V(CT,1)  = E0;
        FT_AVAIL_MAX_V(CT,1)  = FT_AVAIL_MAX;
        FT_V(CT,1)  = FT;
        %FT_V_HW(CT,:)  = FT_AVAIL_HW;
        G_V(CT,1)  = G;
        GAMDOT_V(CT,1)  = GAMDOT;
        L_V(CT,1)  = L;
        OF_V(CT,1)  = OF;
        OF_V_HW(CT,:) = OF_HW;
        QBAR_V(CT,1)  = QBAR;
        SELECTED_V_FUNCMODE(CT,:)  = SELECTED_FUNCMODE;
        SIGMA_V(CT,1) = SIGMA/DTR;
        W_V(CT,1) = W;
end

%% SubFunction
function [Err, VAR_IN] = AOAFUNC(AOA_IN,VAR_IN)
        AOA = AOA_IN; % Change in AOA triggers code to call CL function
        ALT = VAR_IN.ALT;
        V = VAR_IN.V;
        Err = CL-CL_REQ;
end

%% SubFunction
function [Err, VAR_IN] = THRL_VARFUNC(THRL_VAR_IN, VAR_IN)
        AOA = VAR_IN.AOA;
        ALT = VAR_IN.ALT;
        V = VAR_IN.V;
        FT = VAR_IN.FT;
        THRL_VAR = THRL_VAR_IN; % Change in THRL_VAR triggers code to call CL function

        Err = abs(FT-FT_AVAIL);
End
```

B.3.2 *PM_MD0008*

```
% PREALLOCATE VECTORS
AISP_EFF_V          = zeros(TRAJ_NSTEP,1);
AISP_V    = zeros(TRAJ_NSTEP,1);
AISP_V_HW           = zeros(TRAJ_NSTEP,length(VEHICLE_HW));
ALD_V     = zeros(TRAJ_NSTEP,1);
AMACH_V             = zeros(TRAJ_NSTEP,1);
AN_V      = zeros(TRAJ_NSTEP,1);
AOA_V     = zeros(TRAJ_NSTEP,1);
CD_V      = zeros(TRAJ_NSTEP,1);
CD_V_HW             = zeros(TRAJ_NSTEP,length(VEHICLE_HW));
CL_V      = zeros(TRAJ_NSTEP,1);
CL_V_HW             = zeros(TRAJ_NSTEP,length(VEHICLE_HW));
D_V       = zeros(TRAJ_NSTEP,1);
DGAM_V = zeros(TRAJ_NSTEP,1);
DPSI_V    = zeros(TRAJ_NSTEP,1);
DR_V      = zeros(TRAJ_NSTEP,1);
DT_V      = zeros(TRAJ_NSTEP,1);
DUCT_PRESSURE_V = zeros(TRAJ_NSTEP,1);
DW_V      = zeros(TRAJ_NSTEP,1);
DWF_V     = zeros(TRAJ_NSTEP,1);
DWO_V     = zeros(TRAJ_NSTEP,1);
DX_V      = zeros(TRAJ_NSTEP,1);
DY_V      = zeros(TRAJ_NSTEP,1);
EDOT_V    = zeros(TRAJ_NSTEP,1);
EI_V      = zeros(TRAJ_NSTEP,1);
FT_AVAIL_MAX_V = zeros(TRAJ_NSTEP,1);
FT_V      = zeros(TRAJ_NSTEP,1);
```

```
%FT_V_HW            = zeros(TRAJ_NSTEP,length(VEHICLE_HW));
G_V         = zeros(TRAJ_NSTEP,1);
GAMDOT_V            = zeros(TRAJ_NSTEP,1);
L_V         = zeros(TRAJ_NSTEP,1);
OF_V        = zeros(TRAJ_NSTEP,1);
OF_V_HW             = zeros(TRAJ_NSTEP,length(VEHICLE_HW));
PSIDOT_V            = zeros(TRAJ_NSTEP,1);
QBAR_V  = zeros(TRAJ_NSTEP,1);
SELECTED_V_FUNCMODE     = cell(TRAJ_NSTEP,length(VEHICLE_FUNCTION));
SIGMA_V= zeros(TRAJ_NSTEP,1);
W_V = zeros(TRAJ_NSTEP,1);
%INITIAL POINTS FROM TRAJECTORY
I = max(I_V);
% CALCULTE CHANGE IN AMACH PER STEP
ALT_START = ALT_V(I);
% CALCULTE CHANGE IN ALT PER STEP
DT = ENDURANCE_CRUISE/(TRAJ_NSTEP);
% ITERATE FOR EACH ENERGY LEVEL
for CT = 1:TRAJ_NSTEP %ending at E(N+1) in order to store derivatives for E(N)
%INPUTS FROM TRAJECTORY
I = max(I_V);
WR = WR_V(I);
GAM = GAM_V(I)*DTR;
PSI = PSI_V(I)*DTR;
V = V_V(I);%sqrt((V_V(I+1).^2 + V_V(I).^2)/2);
ALT = ALT_V(I);%(ALT_V(I+1) + ALT_V(I))/2;
%ANALYSIS
FLTCOND = fltcon(ALT,0,V,0);
QBAR=FLTCOND.QBAR; AMACH=FLTCOND.AMACH;
W = WS*SPLN/WR;
G = G0./(1 + ALT./RE).^2;
GAMDOT = 0;
PSIDOT = 0;
SIGMA = 0;
L = W./G0 .* (G - V.^2./(RE + ALT));
CL_REQ = L./(QBAR*SPLN);
VAR_IN.ALT = ALT;
VAR_IN.V = V;
[AOA_OUT, VAR_IN] = runsolver(@AOAFUNC, 0, VAR_IN, 1, 1e-2);
AOA = AOA_OUT;
D = QBAR.*CD*SPLN;
ALD = L/D;
AN = 0;
THRL_VAR = 1;
FT_AVAIL_MAX = FT_AVAIL;
FT = W*(AN + D/W + G/G0*sin(GAM)); % Thrust requirement for max acceleration
if (FT_AVAIL_MAX < FT & INSUFF_THRUST_CHECK == 'Y')
disp('         I         ALT          V         GAM          W FT_AVAIL_MAX          FT         D    G./G0.*sin(GAM)')
disp([I ALT V GAM/DTR W FT_AVAIL_MAX FT D  G./G0.*sin(GAM)])
error('INSUFFICIENT THRUST')
end
VAR_IN.AOA = AOA;
VAR_IN.ALT = ALT;
VAR_IN.V = V;
VAR_IN.FT = FT;
[THRL_VAR_OUT, VAR_IN] = runsolver(@THRL_VARFUNC, 1, VAR_IN, 1, 1e-3);
THRL_VAR = THRL_VAR_OUT; % Change in THRL_VAR triggers code to call FT function
FT = FT_AVAIL;
AISP = AISP;
OF = OF;
DUCT_PRESSURE = DUCT_PRESSURE;
THRL_VAR_HW = zeros(size(FT_AVAIL_HW));
THRL_VAR_HW(FT_AVAIL_HW~=0) = THRL_VAR;
AISP_EFF = (FT - D - W*sin(GAM))/(FT/AISP);
```

```
% CALCULATE DELTAS TO GET TO CURRENT POINT
DR = DT*V;
DW = - (FT/AISP)*DT;
DWF = - DW/(1+OF);
DWO = OF*DWF;
WRNEXT = 1/(1/WR + DW/(WS*SPLN));
DX = DR*cos(PSI);
DY = DR*sin(PSI);
DPSI = 0;
DGAM = 0;
VAR_IN.AMACH = AMACH;
VAR_IN.WS = WS;
VAR_IN.SPLN = SPLN;
VAR_IN.WRNEXT = WRNEXT;
VAR_IN.G0 = G0;
VAR_IN.RE = RE;
VAR_IN.CL = CL;
[ALT_NEXT, VAR_IN] = runsolver(@ALTFUNC, ALT, VAR_IN, 1, 1e-1);
FLTCOND = fltcon(ALT_NEXT,AMACH,0,0);
V_NEXT = FLTCOND.V;
E0 = ALT_V(I)*RE/(RE+ ALT_V(I)) + V_V(I)^2/(2*G0);
EI = ALT_NEXT*RE/(RE+ALT_NEXT) + V_NEXT^2/(2*G0);
RDOT = V*cos(GAM)*RE/(RE+ALT);
EDOT = (EI-E0)/DT;%sqrt((V^2+V_V(I+1)^2)/2)*AN;
% ASSIGN VALUES AT CURRENT POINT
I_V(I+1,1) = I+1;
ALT_V(I+1,1) = ALT_NEXT;
FF_V(I+1,1) = FF_V(I) + DWF/(WS*SPLN);
GAM_V(I+1,1) = (GAM+DGAM)/DTR;
PSI_V(I+1,1) = (PSI+DPSI)/DTR;
TIME_V(I+1,1) = TIME_V(I)+DT;
RANGE_V(I+1,1) = RANGE_V(I)+DR;
V_V(I+1,1) = V;
WR_V(I+1,1) = WRNEXT;
X_V(I+1,1) = X_V(I) + DX;
Y_V(I+1,1) = Y_V(I) + DY;
FT_V_HW(I+1,:)  = FT_AVAIL_HW;
DUCT_PRESSURE_V_HW(I+1,:)  = DUCT_PRESSURE_HW;
THRL_VAR_REQ_V_HW(I+1,:)  = THRL_VAR_HW;
RANGE = RANGE_V(I+1,1);
X_RANGE = X_V(I+1,1);
ENDURANCE = TIME_V(I+1,1);
WR = WR_V(I+1,1);
FF = FF_V(I+1,1);
FT_MAX_HW =  max(FT_V_HW,[],1);
DUCT_PRESSURE_MAX_HW = max(DUCT_PRESSURE_V_HW,[],1);
THRL_VAR_MAX = max(max(THRL_VAR_REQ_V_HW,[],1));
TRAJSEG_V(I+1,1)  = METHOD_TRAJSEG;
AISP_EFF_V(CT,1) = AISP_EFF;
AISP_V(CT,1) = AISP;
AISP_V_HW(CT,:) = AISP_HW;
ALD_V(CT,1)  = ALD;
AMACH_V(CT,1)  = AMACH;
AN_V(CT,1)  = AN;
AOA_V(CT,1)  = AOA;
CD_V(CT,1)  = CD;
CD_V_HW(CT,:)  = CD_HW;
CL_V(CT,1)  = CL;
CL_V_HW(CT,:)  = CL_HW;
D_V(CT,1) = D;
DGAM_V(CT,1)  = DGAM/DTR;
DPSI_V(CT,1) = DPSI/DTR;
DR_V(CT,1)  = DR;
DT_V(CT,1)  = DT;
DUCT_PRESSURE_V(CT,1)  = DUCT_PRESSURE;
```

```
%DUCT_PRESSURE_V_HW(CT,1)  = DUCT_PRESSURE_HW;
DW_V(CT,1)  = DW;
DWF_V(CT,1)  = DWF;
DWO_V(CT,1)  = DWO;
DX_V(CT,1) = DX;
DY_V(CT,1) = DY;
EDOT_V(CT,1)  = EDOT;
EI_V(CT,1)  = E0;
FT_AVAIL_MAX_V(CT,1)  = FT_AVAIL_MAX;
FT_V(CT,1)  = FT;
%FT_V_HW(CT,:)  = FT_AVAIL_HW;
G_V(CT,1)  = G;
GAMDOT_V(CT,1)  = GAMDOT;
L_V(CT,1)  = L;
OF_V(CT,1)  = OF;
OF_V_HW(CT,:) = OF_HW;
QBAR_V(CT,1)  = QBAR;
SELECTED_V_FUNCMODE(CT,:)  = SELECTED_FUNCMODE;
SIGMA_V(CT,1) = SIGMA/DTR;
W_V(CT,1) = W;
end
%% SubFunction
function [Err, VAR_IN] = AOAFUNC(AOA_IN,VAR_IN)
AOA = AOA_IN; % Change in AOA triggers code to call CL function
ALT = VAR_IN.ALT;
V = VAR_IN.V;
Err = CL-CL_REQ;
end
%% SubFunction
function [Err, VAR_IN] = ALTFUNC(ALT_NEXT,VAR_IN)
AMACH = VAR_IN.AMACH;
WS = VAR_IN.WS;
SPLN = VAR_IN.SPLN;
WRNEXT = VAR_IN.WRNEXT;
G0 = VAR_IN.G0;
RE = VAR_IN.RE;
CL = VAR_IN.CL;
FLTCOND = fltcon(ALT_NEXT,AMACH,0,0);
QBAR_NEXT=FLTCOND.QBAR;
V_NEXT = FLTCOND.V;
L_NEXT = QBAR_NEXT*SPLN*CL;
W_REQ = WS*SPLN/WRNEXT;
G = G0./(1 + ALT_NEXT./RE).^2;
L_REQ = W_REQ./G0 .* (G - V_NEXT.^2./(RE + ALT_NEXT));
Err = L_NEXT-L_REQ;
end
%% SubFunction
function [Err, VAR_IN] = THRL_VARFUNC(THRL_VAR_IN, VAR_IN)
AOA = VAR_IN.AOA;
ALT = VAR_IN.ALT;
V = VAR_IN.V;
FT = VAR_IN.FT;
THRL_VAR = THRL_VAR_IN; % Change in THRL_VAR triggers code to call CL function

Err = abs(FT-FT_AVAIL);
End
```

B.3.3 *PM_MD0009*

```
% PREALLOCATE VECTORS

AISP_EFF_V          = zeros(TRAJ_NSTEP,1);

AISP_V    = zeros(TRAJ_NSTEP,1);
```

```
AISP_V_HW           = zeros(TRAJ_NSTEP,length(VEHICLE_HW));
ALD_V     = zeros(TRAJ_NSTEP,1);
AMACH_V             = zeros(TRAJ_NSTEP,1);
AN_V      = zeros(TRAJ_NSTEP,1);
AOA_V     = zeros(TRAJ_NSTEP,1);
CD_V      = zeros(TRAJ_NSTEP,1);
CD_V_HW             = zeros(TRAJ_NSTEP,length(VEHICLE_HW));
CL_V      = zeros(TRAJ_NSTEP,1);
CL_V_HW             = zeros(TRAJ_NSTEP,length(VEHICLE_HW));
D_V       = zeros(TRAJ_NSTEP,1);
DGAM_V = zeros(TRAJ_NSTEP,1);
DPSI_V    = zeros(TRAJ_NSTEP,1);
DR_V      = zeros(TRAJ_NSTEP,1);
DT_V      = zeros(TRAJ_NSTEP,1);
DUCT_PRESSURE_V = zeros(TRAJ_NSTEP,1);
DW_V      = zeros(TRAJ_NSTEP,1);
DWF_V     = zeros(TRAJ_NSTEP,1);
DWO_V     = zeros(TRAJ_NSTEP,1);
DX_V      = zeros(TRAJ_NSTEP,1);
DY_V      = zeros(TRAJ_NSTEP,1);
EDOT_V  = zeros(TRAJ_NSTEP,1);
EI_V      = zeros(TRAJ_NSTEP,1);
FT_AVAIL_MAX_V  = zeros(TRAJ_NSTEP,1);
FT_V      = zeros(TRAJ_NSTEP,1);
%FT_V_HW             = zeros(TRAJ_NSTEP,length(VEHICLE_HW));
G_V       = zeros(TRAJ_NSTEP,1);
GAMDOT_V            = zeros(TRAJ_NSTEP,1);
L_V       = zeros(TRAJ_NSTEP,1);
OF_V      = zeros(TRAJ_NSTEP,1);
OF_V_HW             = zeros(TRAJ_NSTEP,length(VEHICLE_HW));
PSIDOT_V            = zeros(TRAJ_NSTEP,1);
QBAR_V  = zeros(TRAJ_NSTEP,1);
SELECTED_V_FUNCMODE   = cell(TRAJ_NSTEP,length(VEHICLE_FUNCTION));
```

```
SIGMA_V= zeros(TRAJ_NSTEP,1);

W_V = zeros(TRAJ_NSTEP,1);

% ITERATE FOR EACH ENERGY LEVEL

for CT = 1:TRAJ_NSTEP %ending at E(N+1) in order to store derivatives for E(N)

%INPUTS FROM TRAJECTORY

I = max(I_V);

WR = WR_V(I);

GAM = GAM_V(I)*DTR;

PSI = PSI_V(I)*DTR;

V = V_V(I);%sqrt((V_V(I+1).^2 + V_V(I).^2)/2);

ALT = ALT_V(I);%(ALT_V(I+1) + ALT_V(I))/2;

%ANALYSIS

FLTCOND = fltcon(ALT,0,V,0);

QBAR=FLTCOND.QBAR; AMACH=FLTCOND.AMACH;

W = WS*SPLN/WR;

G = G0./(1 + ALT./RE).^2;

GAMDOT = 0;

PSIDOT = 0;

SIGMA = 0;

CL = 0;

CL_HW = 0;

CD = 0;

CD_HW = 0;

L = 0;

AOA = 0;

D = 0;

ALD = 0;

AN = 0;

THRL_VAR = 0;

THRL_VAR_HW = zeros(size(VEHICLE_HW));

FT_AVAIL_MAX = 0;

FT = 0; % Thrust requirement for max acceleration

AISP = 0;
```

```
AISP_HW = 0;

OF = 0;

OF_HW = 0;

DUCT_PRESSURE = 0;

DUCT_PRESSURE_HW = 0;

AISP_EFF = 0;

ALT_NEXT = ALT_V(I);

V_NEXT = V_V(I);

E0 = ALT_V(I)*RE/(RE+ ALT_V(I)) + V_V(I)^2/(2*G0);

EI = ALT_NEXT*RE/(RE+ALT_NEXT) + V_NEXT^2/(2*G0);

% CALCULATE DELTAS TO GET TO CURRENT POINT

RDOT = 0;

EDOT = 0;%sqrt((V^2+V_V(I+1)^2)/2)*AN;

DT = 0;

DW = 0;

DWF = 0;

DWO = 0;

WRNEXT = TRAJ_WR;

DR = 0;

DX = 0;

DY = 0;

DPSI = 0;

DGAM = 0;

% ASSIGN VALUES AT CURRENT POINT

I_V(I+1,1) = I+1;

ALT_V(I+1,1) = ALT_V(I);

FF_V(I+1,1) = FF_V(I) + DWF/(WS*SPLN);

GAM_V(I+1,1) = (GAM+DGAM)/DTR;

PSI_V(I+1,1) = (PSI+DPSI)/DTR;

TIME_V(I+1,1) = TIME_V(I)+DT;

RANGE_V(I+1,1) = RANGE_V(I)+DR;

V_V(I+1,1) = V_NEXT;

WR_V(I+1,1) = WRNEXT;
```

```
X_V(I+1,1) = X_V(I) + DX;

Y_V(I+1,1) = Y_V(I) + DY;

FT_V_HW(I+1,:)  = zeros(size(VEHICLE_HW));

DUCT_PRESSURE_MAX_HW(I+1,:)  = zeros(size(VEHICLE_HW));

THRL_VAR_REQ_V_HW(I+1,:)  = THRL_VAR_HW;

TRAJSEG_V(I+1,1)  = METHOD_TRAJSEG;

RANGE = RANGE_V(I+1,1);

X_RANGE = X_V(I+1,1);

ENDURANCE = TIME_V(I+1,1);

WR = WR_V(I+1,1);

FF = FF_V(I+1,1);

FT_MAX_HW =  max(FT_V_HW,[],1);

DUCT_PRESSURE_MAX_HW = max(DUCT_PRESSURE_V_HW,[],1);

THRL_VAR_MAX = max(max(THRL_VAR_REQ_V_HW,[],1));

AISP_EFF_V(CT,1) = AISP_EFF;

AISP_V(CT,1) = AISP;

AISP_V_HW(CT,:) = AISP_HW;

ALD_V(CT,1)  = ALD;

AMACH_V(CT,1)  = AMACH;

AN_V(CT,1)  = AN;

AOA_V(CT,1)  = AOA;

CD_V(CT,1)  = CD;

CD_V_HW(CT,:)  = CD_HW;

CL_V(CT,1)  = CL;

CL_V_HW(CT,:)  = CL_HW;

D_V(CT,1) = D;

DGAM_V(CT,1)  = DGAM/DTR;

DPSI_V(CT,1) = DPSI/DTR;

DR_V(CT,1)  = DR;

DT_V(CT,1)  = DT;

DUCT_PRESSURE_V(CT,1)  = DUCT_PRESSURE;

%DUCT_PRESSURE_V_HW(CT,1)  = DUCT_PRESSURE_HW;

DW_V(CT,1)  = DW;
```

```
DWF_V(CT,1)  = DWF;

DWO_V(CT,1)  = DWO;

DX_V(CT,1) = DX;

DY_V(CT,1) = DY;

EDOT_V(CT,1)  = EDOT;

EI_V(CT,1)  = E0;

FT_AVAIL_MAX_V(CT,1)  = FT_AVAIL_MAX;

FT_V(CT,1)  = FT;

%FT_V_HW(CT,:)  = FT_AVAIL_HW;

G_V(CT,1)  = G;

GAMDOT_V(CT,1)  = GAMDOT;

L_V(CT,1)  = L;

OF_V(CT,1)  = OF;

OF_V_HW(CT,:) = OF_HW;

QBAR_V(CT,1)  = QBAR;

SELECTED_V_FUNCMODE(CT,:)  = SELECTED_FUNCMODE;

SIGMA_V(CT,1) = SIGMA/DTR;

W_V(CT,1) = W;

end
```

B.3.4 *PM_MD0011*

```
% PREALLOCATE VECTORS
AISP_EFF_V          = zeros(TRAJ_NSTEP,1);
AISP_V    = zeros(TRAJ_NSTEP,1);
AISP_V_HW           = zeros(TRAJ_NSTEP,length(VEHICLE_HW));
ALD_V     = zeros(TRAJ_NSTEP,1);
AMACH_V             = zeros(TRAJ_NSTEP,1);
AN_V      = zeros(TRAJ_NSTEP,1);
AOA_V     = zeros(TRAJ_NSTEP,1);
CD_V      = zeros(TRAJ_NSTEP,1);
CD_V_HW             = zeros(TRAJ_NSTEP,length(VEHICLE_HW));
CL_V      = zeros(TRAJ_NSTEP,1);
CL_V_HW             = zeros(TRAJ_NSTEP,length(VEHICLE_HW));
D_V       = zeros(TRAJ_NSTEP,1);
DGAM_V = zeros(TRAJ_NSTEP,1);
DPSI_V    = zeros(TRAJ_NSTEP,1);
DR_V      = zeros(TRAJ_NSTEP,1);
DT_V      = zeros(TRAJ_NSTEP,1);
DUCT_PRESSURE_V = zeros(TRAJ_NSTEP,1);
DW_V      = zeros(TRAJ_NSTEP,1);
DWF_V     = zeros(TRAJ_NSTEP,1);
DWO_V     = zeros(TRAJ_NSTEP,1);
DX_V      = zeros(TRAJ_NSTEP,1);
DY_V      = zeros(TRAJ_NSTEP,1);
EDOT_V    = zeros(TRAJ_NSTEP,1);
EI_V      = zeros(TRAJ_NSTEP,1);
```

```
FT_AVAIL_MAX_V = zeros(TRAJ_NSTEP,1);
FT_V        = zeros(TRAJ_NSTEP,1);
%FT_V_HW             = zeros(TRAJ_NSTEP,length(VEHICLE_HW));
G_V         = zeros(TRAJ_NSTEP,1);
GAMDOT_V            = zeros(TRAJ_NSTEP,1);
L_V         = zeros(TRAJ_NSTEP,1);
OF_V        = zeros(TRAJ_NSTEP,1);
OF_V_HW             = zeros(TRAJ_NSTEP,length(VEHICLE_HW));
PSIDOT_V            = zeros(TRAJ_NSTEP,1);
QBAR_V  = zeros(TRAJ_NSTEP,1);
SELECTED_V_FUNCMODE     = cell(TRAJ_NSTEP,length(VEHICLE_FUNCTION));
SIGMA_V= zeros(TRAJ_NSTEP,1);
W_V = zeros(TRAJ_NSTEP,1);
%INITIAL POINTS FROM TRAJECTORY
I = max(I_V);
%INITIAL POINTS FROM TRAJECTORY
V_START = V_V(I);
ALT_START = ALT_V(I);
X_START = X_V(I);
Y_START = Y_V(I);
PSI_START = PSI_V(I)*DTR;
% ASSIGN CONTROL VARIABLES TO traj
AN_LIM = TRAJ_AN_MAX;
PSI_CHANGE = TRAJ_PSI_TURN*DTR;
G = G0./(1 + ALT_START./RE).^2;
RTURN = V_START^2/sqrt(AN_LIM^2*G0^2-(G-V_START^2/(RE+ALT_START))^2);
SIGMA = acos(1/(AN_LIM*G0)*(G-V_START^2/(RE+ALT_START)));
PSI_FINAL = PSI_START+PSI_CHANGE;
% CALCULTE CHANGE IN PSI PER STEP
DPSI = (PSI_FINAL - PSI_START)/(TRAJ_NSTEP+1);
PSI_V(I) = PSI_V(I)+DPSI/DTR;
% ITERATE FOR EACH ENERGY LEVEL
for CT = 1:TRAJ_NSTEP %ending at E(N+1) in order to store derivatives for E(N)
%INPUTS FROM TRAJECTORY
I = max(I_V);
WR = WR_V(I);
GAM = GAM_V(I)*DTR;
PSI = PSI_V(I)*DTR;
V = V_V(I);%sqrt((V_V(I+1).^2 + V_V(I).^2)/2);
ALT = ALT_V(I);%(ALT_V(I+1) + ALT_V(I))/2;
%ANALYSIS
FLTCOND = fltcon(ALT,0,V,0);
QBAR=FLTCOND.QBAR; AMACH=FLTCOND.AMACH;
W = WS*SPLN/WR;
G = G0./(1 + ALT./RE).^2;
GAMDOT = 0;
L = W./(G0*cos(SIGMA)) .* (G - V.^2./(RE + ALT));
CL_REQ = L./(QBAR*SPLN);
VAR_IN.ALT = ALT;
VAR_IN.V = V;
[AOA_OUT, VAR_IN] = runsolver(@AOAFUNC, 0, VAR_IN, 1, 1e-2);
AOA = AOA_OUT;
D = QBAR.*CD*SPLN;
ALD = L/D;
AN = 0;
THRL_VAR = 1;
FT_AVAIL_MAX = FT_AVAIL;
FT = W*(AN + D/W + G/G0*sin(GAM)); % Thrust requirement for max acceleration
if (FT_AVAIL_MAX < FT & INSUFF_THRUST_CHECK == 'Y' )
disp('          I          ALT          V          GAM          W FT_AVAIL_MAX          FT          D    G./G0.*sin(GAM)')
disp([I ALT V GAM/DTR W FT_AVAIL_MAX FT D  G./G0.*sin(GAM)])
error('INSUFFICIENT THRUST')
end
VAR_IN.AOA = AOA;
VAR_IN.ALT = ALT;
VAR_IN.V = V;
VAR_IN.FT = FT;
[THRL_VAR_OUT, VAR_IN] = runsolver(@THRL_VARFUNC, 1, VAR_IN, 1, 1e-3);
THRL_VAR = THRL_VAR_OUT; % Change in THRL_VAR triggers code to call FT function
FT = FT_AVAIL;
AISP = AISP;
OF = OF;
DUCT_PRESSURE = DUCT_PRESSURE;
THRL_VAR_HW = zeros(size(FT_AVAIL_HW));
THRL_VAR_HW(FT_AVAIL_HW~=0) = THRL_VAR;
AISP_EFF = (FT - D - W*sin(GAM))/(FT/AISP);
ALT_NEXT = ALT;
V_NEXT = V;
E0 = ALT_V(I)*RE/(RE+ ALT_V(I)) + V_V(I)^2/(2*G0);
EI = ALT_NEXT*RE/(RE+ALT_NEXT) + V_NEXT^2/(2*G0);
% CALCULATE DELTAS TO GET TO CURRENT POINT
EDOT = V*AN;%sqrt((V^2+V_V(I+1)^2)/2)*AN;
PSIDOT = AN_LIM*G0/V*sin(SIGMA);
XDOT = V*cos(GAM)*cos(PSI)*RE/(RE+ALT);
YDOT = V*cos(GAM)*sin(PSI)*RE/(RE+ALT);
DT = DPSI/PSIDOT;
DW = - (FT/AISP)*DT;
DWF = - DW/(1+OF);
DWO = OF*DWF;
WRNEXT = 1/(1/WR + DW/(WS*SPLN));
DX = XDOT*DT;
DY = YDOT*DT;
DR = sqrt(DX^2+DY^2);
DGAM = GAMDOT*DT;
% ASSIGN VALUES AT CURRENT POINT
I_V(I+1,1) = I+1;
ALT_V(I+1,1) = ALT;
FF_V(I+1,1) = FF_V(I) + DWF/(WS*SPLN);
GAM_V(I+1,1) = (GAM+DGAM)/DTR;
PSI_V(I+1,1) = (PSI+DPSI)/DTR;
TIME_V(I+1,1) = TIME_V(I)+DT;
RANGE_V(I+1,1) = RANGE_V(I)+DR;
V_V(I+1,1) = V;
WR_V(I+1,1) = WRNEXT;
X_V(I+1,1) = X_V(I) + DX;
Y_V(I+1,1) = Y_V(I) + DY;
FT_V_HW(I+1,:)  = FT_AVAIL_HW;
DUCT_PRESSURE_V_HW(I+1,:)  = DUCT_PRESSURE_HW;
THRL_VAR_REQ_V_HW(I+1,:)  = THRL_VAR_HW;
TRAJSEG_V(I+1,1)  = METHOD_TRAJSEG;
RANGE = RANGE_V(I+1,1);
X_RANGE = X_V(I+1,1);
ENDURANCE = TIME_V(I+1,1);
WR = WR_V(I+1,1);
FF = FF_V(I+1,1);
FT_MAX_HW =  max(FT_V_HW,[],1);
DUCT_PRESSURE_MAX_HW = max(DUCT_PRESSURE_V_HW,[],1);
THRL_VAR_MAX = max(max(THRL_VAR_REQ_V_HW,[],1));
AISP_EFF_V(CT,1) = AISP_EFF;
AISP_V(CT,1) = AISP;
AISP_V_HW(CT,:) = AISP_HW;
ALD_V(CT,1)  = ALD;
AMACH_V(CT,1)  = AMACH;
AN_V(CT,1)  = AN;
AOA_V(CT,1)  = AOA;
CD_V(CT,1)  = CD;
CD_V_HW(CT,:)  = CD_HW;
CL_V(CT,1)  = CL;
CL_V_HW(CT,:)  = CL_HW;
D_V(CT,1) = D;
```

```
DGAM_V(CT,1)  = DGAM/DTR;
DPSI_V(CT,1) = DPSI/DTR;
DR_V(CT,1)  = DR;
DT_V(CT,1)  = DT;
DUCT_PRESSURE_V(CT,1)  = DUCT_PRESSURE;
%DUCT_PRESSURE_V_HW(CT,1)  = DUCT_PRESSURE_HW;
DW_V(CT,1)  = DW;
DWF_V(CT,1)  = DWF;
DWO_V(CT,1)  = DWO;
DX_V(CT,1) = DX;
DY_V(CT,1) = DY;
EDOT_V(CT,1)  = EDOT;
EI_V(CT,1)  = E0;
FT_AVAIL_MAX_V(CT,1)  = FT_AVAIL_MAX;
FT_V(CT,1)  = FT;
%FT_V_HW(CT,:)  = FT_AVAIL_HW;
G_V(CT,1)  = G;
GAMDOT_V(CT,1)  = GAMDOT;
L_V(CT,1)  = L;
OF_V(CT,1)  = OF;
OF_V_HW(CT,:) = OF_HW;
QBAR_V(CT,1)  = QBAR;
SELECTED_V_FUNCMODE(CT,:)  = SELECTED_FUNCMODE;
SIGMA_V(CT,1) = SIGMA/DTR;
W_V(CT,1) = W;
end
PSI_V(I+1,1) = PSI_FINAL/DTR; % Eliminate roundoff errors
%% SubFunction
function [Err, VAR_IN] = AOAFUNC(AOA_IN,VAR_IN)
AOA = AOA_IN; % Change in AOA triggers code to call CL function
ALT = VAR_IN.ALT;
V = VAR_IN.V;
Err = CL-CL_REQ;
end
%% SubFunction
function [Err, VAR_IN] = THRL_VARFUNC(THRL_VAR_IN, VAR_IN)
AOA = VAR_IN.AOA;
ALT = VAR_IN.ALT;
V = VAR_IN.V;
FT = VAR_IN.FT;
THRL_VAR = THRL_VAR_IN; % Change in THRL_VAR triggers code to call CL function
Err = abs(FT-FT_AVAIL);
end
```

B.4 Weight & Balance

B.4.1 *WB_MD0005*

```
%%%%%% Analysis %%%%%%%%%
if WR < 1
WR
error('WR < 1 vehicle gained weight over trajectory')
end
WOX_WF = (1-1/WR)/FF - 1;
RHO_PPL=(WOX_WF+1)/(WOX_WF/RHO_OX + 1/RHO_FUEL);
MDOT0_X_S = ...
[1         2         3         4         5];
WENG_S = ...
[126.34    243.04    367.05    480.37    600.40].*G0;
WENG = interp1(MDOT0_X_S,WENG_S,MDOT0_X,'linear','extrap');
EBAND_CORR = -3.77287e-07*TIME_HYP^2+4.96880e-04*TIME_HYP+7.08950e-02;
EBAND = max(EBAND_CORR+EBAND,EBAND);
AKSTR=(0.317+EBAND)*TAU^0.206;
```

```
WSTR_OEW = AKSTR*SPLN^0.138;
AITPS = (6.0717*TIME_HYP^(0.2555));
WTPS_OEW = (0.0179*TIME_HYP^(-0.35))*(SPLN*AKW);
% WEIGHT BUDGET CONSTANTS
WFIX = WUN;
WPAY = WCARGO;
% WEIGHT BUDGET OEW = (1+AMUA)*(WSTR + WTPS + WENG + WSYS);  WSYS = WFIX + FWSYS*OEW;
OEW_W = (WFIX+WENG+AITPS*(SPLN*AKW)) / (1/(1+AMUA)-WSTR_OEW-FWSYS);
OWE_W = OEW_W+WPAY;
if ((1/(1+AMUA)-WSTR_OEW-FWSYS) < 0.0)
fprintf('AKSTR = %f\n',AKSTR);
fprintf('WR = %f\n',WR);
fprintf('1/(1+AMUA) = %f\n',1/(1+AMUA));
fprintf('1/(1+AMUA)-WSTR_OEW-FWSYS = %f\n',(1/(1+AMUA)-WSTR_OEW-FWSYS));
fprintf('FWSYS = %f\n',FWSYS);
error('CONVERGENCE FAILURE:');
end
% VOLUME BUDGET OWE
VTOTAL = TAU*SPLN^1.5;
VFIX = VUN;
VSYS = VFIX+AKVS*VTOTAL;
VPAY = (WCARGO/RHO_CARGO/G0);
VSTR = VTOTAL*2.85072e-1*exp(TIME_HYP*-2.98649e-4);
VTPS = VTOTAL*AKVTPS;
VVOID = VTOTAL*AKVV;
% from VPPL = OWE_V*(WR-1)/(RHO_PPL*G0) = VTOTAL - VVOID - VSYS - VENG - VPAY - VCREW - VCHUTE
OWE_V = (VTOTAL-VSYS-VP-VPAY-VSTR-VTPS-VVOID)/((WR-1)/(RHO_PPL*G0));
AIP = RHO_PPL/(WR-1);
% WEIGHT AND VOLUME BREAKFORWN
OWE = OWE_W;
OEW = OEW_W;
WSTR = WSTR_OEW*OEW;
WTPS = AITPS*(SPLN*AKW);
AISTR = WSTR/(SPLN*AKW);
TOGW = OWE*WR;
WPPL = TOGW*(1-1/WR);
WFUEL = TOGW*FF;
WOX = WOX_WF*WFUEL;
WP = WENG;
WSYS = WFIX + FWSYS*OEW;
AMZFW = OWE+WPAY;
AMWE = OWE;
WMARGIN = OEW-(WSYS+WSTR+WTPS+WP);
VENG = VP;
VPPL = WPPL/RHO_PPL/9.81;
VFUEL = WFUEL/RHO_FUEL/9.81;
VOX = WOX/RHO_FUEL/9.81;
```

Appendix C

GHV Verification CMDS

Appendix B content goes on this page

C.1 Input File

```
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%                 AVD_ABE Input File For GHVVerificationAeroPropLUT
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
function [Variable] = GHVVerificationAeroPropLUT (Variable)

% ***********************************************************************
% ArchGen: GHVVerificationAeroPropLUT Control Variables
% ***********************************************************************

%Set X-Vector Variable for FZERO solver %**********************************
% SPLN_INIT          m^2          Planform area
% WS_INIT            N/m^2         Wing loading (i.e. TOGW/S)
% X0                              Numerical values for X-Vector
%************************************************************************
Variable.SYSPROC.INPUT.SPLN_INIT = 15.5;
Variable.SYSPROC.INPUT.WS_INIT = 1799.591;
Variable.SYSPROC.INPUT.X0 = [Variable.SYSPROC.INPUT.SPLN_INIT, Variable.SYSPROC.INPUT.WS_INIT];

%Multipoint Variation %***************************************************
% MODE_DESIGN                Design mode
%                    = 1  Analysis Points (Single), CMDS Optimization (No),  CMDS Convergence (No)
%                    = 2  Analysis Points (Single), CMDS Optimization (No),  CMDS Convergence (Yes)
%                    = 3  Analysis Points (Single), CMDS Optimization (Yes), CMDS Convergence (No)
%                    = 4  Analysis Points (Single), CMDS Optimization (Yes), CMDS Convergence (Yes)
%                    = 5  Analysis Points (Multi),  CMDS Optimization (No),  CMDS Convergence (No)
%                        Array Type (N X 1)
%                    = 6  Analysis Points (Multi),  CMDS Optimization (No),  CMDS Convergence (Yes)
%                        Array Type (N X 1)
%                    = 7  Analysis Points (Multi),  CMDS Optimization (Yes), CMDS Convergence (No)
%                        Array Type (N X 1)
%                    = 8  Analysis Points (Multi),  CMDS Optimization (Yes), CMDS Convergence (Yes)
%                        Array Type (N X 1)
%                    = 9  Analysis Points (Multi),  CMDS Optimization (No),  CMDS Convergence (No)
%                        Array Type (N X N)
%                    = 10 Analysis Points (Multi),  CMDS Optimization (No),  CMDS Convergence (Yes)
%                        Array Type (N X N)
%                    = 11 Analysis Points (Multi),  CMDS Optimization (Yes), CMDS Convergence (No)
%                        Array Type (N X N)
%                    = 12 Analysis Points (Multi),  CMDS Optimization (Yes), CMDS Convergence (Yes)
%                        Array Type (N X N)
% MV_NAMES                   Variables to be traded
% MV_init                  Initial value of trade variables
% MV_SS                     Variable step sizes
% MV_NS                     Number of Steps
% ***********************************************************************
Variable.SYSPROC.INPUT.MODE_DESIGN = 6;
Variable.SYSPROC.INPUT.MV_NAMES = { ...
'Variable.HW.TotalVehicle.GEO.TotalVehicle_GEO_MD0003.INPUT.TAU', ...
'Variable.TRAJSEG.ConstantMachEnduranceCruise_02_PM_MD0008.INPUT.ENDURANCE_CRUISE', ...
'Variable.TRAJSEG.SteadyLevelTurn_01_PM_MD0011.INPUT.TRAJ_AN_MAX', ...
};
Variable.SYSPROC.INPUT.MV_init = [0.055,200];
Variable.SYSPROC.INPUT.MV_SS = [0.005,150];
Variable.SYSPROC.INPUT.MV_NS = [4,4];
Variable.SYSPROC.INPUT.MV_POINTS = [0.067781039, 225.02, 2.402625;
                                    0.067370748, 438.87, 2.264747;
                                    0.065519560, 677.38, 2.242015];
```

```
% ***************************************************************************
% Constants
% ***************************************************************************

%Constant %****************************************************************
%G0              m/s^2        Gravitational acceleration at sealevel
%DTR              /degrees      Conversion from degrees to radians
%RE              m           Radius of the Earth
%***************************************************************************
Variable.SYSPROC.INPUT.G0 = 9.81;
Variable.SYSPROC.INPUT.DTR = pi/180;
Variable.SYSPROC.INPUT.RE = 6371e3;

% ***************************************************************************
% Look-Up Table Array Variables
% ***************************************************************************

%Look-Up Table Input Arrays %***********************************************
%ALT_RANGE         m           Flight Altitude Range: [Start,End]
%ALT_RES           m           Flight Altitude Resolution
%V_RANGE           m/s          Flight Velocity Range: [Start,End]
%V_RES            m/s          Flight Velocity Resolution
%AOA_RANGE         m           Flight Altitude Range: [Start,End]
%AOA_RES           m           Flight Altitude Resolution
%THRL_VAR_RANGE     m           Flight Altitude Range: [Start,End]
%THRL_VAR_RES       m           Flight Altitude Resolution
%***************************************************************************
Variable.SYSPROC.INPUT.ALT_RANGE = [19000,25000];
Variable.SYSPROC.INPUT.ALT_RES = 2000;
Variable.SYSPROC.INPUT.V_RANGE = [1100,2100];
Variable.SYSPROC.INPUT.V_RES = 100;
Variable.SYSPROC.INPUT.AOA_RANGE = [-4.0,4.0];
Variable.SYSPROC.INPUT.AOA_RES = 2.0;
Variable.SYSPROC.INPUT.THRL_VAR_RANGE = [0.25,1.75];
Variable.SYSPROC.INPUT.THRL_VAR_RES = 0.25;


% ***************************************************************************
% Geometry Disciplinary & Method Variables
% ***************************************************************************

%Method: GEO_MD0003   Hardware: TotalVehicle %****************************
%AOA_T            degrees       Thrust incidence angle
%TAU                         Küchemann’s tau
%***************************************************************************
Variable.HW.TotalVehicle.GEO.TotalVehicle_GEO_MD0003.INPUT.AOA_T = 0.675; %0.694301;
Variable.HW.TotalVehicle.GEO.TotalVehicle_GEO_MD0003.INPUT.TAU = 0.067370748;

% ***************************************************************************
% Propulsion Disciplinary & Method Variables
% ***************************************************************************

%Method: PROP_MD0006   Hardware: Scramjet_01 %****************************
%PHI_FUEL_REF                   Reference Fuel Equivalence Ratio
%***************************************************************************
Variable.HW.Scramjet_01.PROP.Scramjet_01_PROP_MD0006.INPUT.PHI_FUEL_REF = 1.2;

% ***************************************************************************
% Performance Matching Disciplinary & Method Variables
% ***************************************************************************

%Performance Matching Disciplinary Process Input Variables%****************
%TRAJ_ALT_V_START    m          Start Point For Vector of altitudes
%TRAJ_FF_V_START                Start Point For Vector of Fuel fractions
%TRAJ_GAM_V_START    degrees       Start Point For Vector of flight path angles
```

```
%TRAJ_PSI_V_START    degrees        Start Point For Vector of heading angles
%TRAJ_RANGE_V_START  m              Start Point For Vector of total range
%TRAJ_TIME_V_START   s              Start Point For Vector of trajetory time
%TRAJ_TRAJSEG_V_START               Start Point For Vector of current flight segment string
%TRAJ_V_V_START      m/s            Start Point For Vector of vel
%TRAJ_WR_V_START                    Start Point For Vector of ratios of final mass at each point in the trajectory to init
%TRAJ_X_V_START      m              Start Point For Vector of position in x-directio
%TRAJ_Y_V_START      m              Start Point For Vector of position in y-directio
%*************************************************************************
Variable.MISSION.INPUT.TRAJ_ALT_V_START = 19054.267;
Variable.MISSION.INPUT.TRAJ_FF_V_START = 0.0;
Variable.MISSION.INPUT.TRAJ_GAM_V_START = 0.0;
Variable.MISSION.INPUT.TRAJ_PSI_V_START = 0;
Variable.MISSION.INPUT.TRAJ_RANGE_V_START = 0.0;
Variable.MISSION.INPUT.TRAJ_TIME_V_START = 0.0;
Variable.MISSION.INPUT.TRAJ_TRAJSEG_V_START = {'START'};
Variable.MISSION.INPUT.TRAJ_V_V_START = 1180.27704;
Variable.MISSION.INPUT.TRAJ_WR_V_START = 1;
Variable.MISSION.INPUT.TRAJ_X_V_START = 0;
Variable.MISSION.INPUT.TRAJ_Y_V_START = 0;

%Method: PM_MD0009   Trajectory Segment: Booster Separation_01 %***********
%TRAJ_NSTEP                 Number of steps in current trajectory segment
%TRAJ_WR                    Ratio of final mass to initial mass for trajectory segment
%*************************************************************************
Variable.TRAJSEG.BoosterSeparation_01_PM_MD0009.INPUT.TRAJ_NSTEP = 1;
Variable.TRAJSEG.BoosterSeparation_01_PM_MD0009.INPUT.TRAJ_WR = 1;

%Method: PM_MD0003   Trajectory Segment: Constant Q Climb_01 %*************
%DUCT_PRESSURE       N/m^2         Engine Duct Pressure
%DUCT_PRESSURE_HW    N/m^2         Engine duct pressure for each hardware on the vehicle
%                           [Scramjet_01, TotalVehicle, WingBody_01]
%INSUFF_THRUST_CHECK              Check for inssufficient thrust in PM
%TRAJ_ALT_END                     Altitude desired at the end of the trajectory segment
%TRAJ_AN_MAX         g's          Maximum acceleration allowed for current trajectory segment
%TRAJ_AN_MIN         g's          Minimum acceleration allowed for current trajectory segment
%TRAJ_NSTEP                       Number of steps in current trajectory segment
%*************************************************************************
Variable.TRAJSEG.ConstantQClimb_01_PM_MD0003.INPUT.DUCT_PRESSURE = 0;
Variable.HW.TotalVehicle.PM.TotalVehicle_PM_MD0003.INPUT.DUCT_PRESSURE_HW = [0, 0, 0];
Variable.TRAJSEG.ConstantQClimb_01_PM_MD0003.INPUT.INSUFF_THRUST_CHECK = 'N';
Variable.TRAJSEG.ConstantQClimb_01_PM_MD0003.INPUT.TRAJ_ALT_END = 24235;
Variable.TRAJSEG.ConstantQClimb_01_PM_MD0003.INPUT.TRAJ_AN_MAX = 2.0;
Variable.TRAJSEG.ConstantQClimb_01_PM_MD0003.INPUT.TRAJ_AN_MIN = 0.15;
Variable.TRAJSEG.ConstantQClimb_01_PM_MD0003.INPUT.TRAJ_NSTEP = 20;

%Method: PM_MD0008   Trajectory Segment: Constant Mach Endurance Cruise_01 %
%DUCT_PRESSURE       N/m^2         Engine Duct Pressure
%DUCT_PRESSURE_HW    N/m^2         Engine duct pressure for each hardware on the vehicle
%                           [Scramjet_01, TotalVehicle, WingBody_01]
%ENDURANCE_CRUISE    s            Flight time during cruise
%INSUFF_THRUST_CHECK              Check for inssufficient thrust in PM
%TRAJ_NSTEP                       Number of steps in current trajectory segment
%*************************************************************************
Variable.TRAJSEG.ConstantMachEnduranceCruise_01_PM_MD0008.INPUT.DUCT_PRESSURE = 0;
Variable.HW.TotalVehicle.PM.TotalVehicle_PM_MD0008.INPUT.DUCT_PRESSURE_HW = [0, 0, 0];
Variable.TRAJSEG.ConstantMachEnduranceCruise_01_PM_MD0008.INPUT.ENDURANCE_CRUISE = 0.01;
Variable.TRAJSEG.ConstantMachEnduranceCruise_01_PM_MD0008.INPUT.INSUFF_THRUST_CHECK = 'N';
Variable.TRAJSEG.ConstantMachEnduranceCruise_01_PM_MD0008.INPUT.TRAJ_NSTEP = 20;

%Method: PM_MD0011   Trajectory Segment: Steady Level Turn_01 %************
%DUCT_PRESSURE       N/m^2         Engine Duct Pressure
%DUCT_PRESSURE_HW    N/m^2         Engine duct pressure for each hardware on the vehicle
%                           [Scramjet_01, TotalVehicle, WingBody_01]
%INSUFF_THRUST_CHECK              Check for inssufficient thrust in PM
```

```
%TRAJ_AN_MAX         g's           Maximum acceleration allowed for current trajectory segment
%TRAJ_NSTEP                        Number of steps in current trajectory segment
%TRAJ_PSI_TURN       degrees       Angle to change heading by
%*************************************************************************
Variable.TRAJSEG.SteadyLevelTurn_01_PM_MD0011.INPUT.DUCT_PRESSURE = 0;
Variable.HW.TotalVehicle.PM.TotalVehicle_PM_MD0011.INPUT.DUCT_PRESSURE_HW = [0, 0, 0];
Variable.TRAJSEG.SteadyLevelTurn_01_PM_MD0011.INPUT.INSUFF_THRUST_CHECK = 'N';
Variable.TRAJSEG.SteadyLevelTurn_01_PM_MD0011.INPUT.TRAJ_AN_MAX = 2.264794;
Variable.TRAJSEG.SteadyLevelTurn_01_PM_MD0011.INPUT.TRAJ_NSTEP = 20;
Variable.TRAJSEG.SteadyLevelTurn_01_PM_MD0011.INPUT.TRAJ_PSI_TURN = 180.0;

%Method: PM_MD0008   Trajectory Segment: Constant Mach Endurance Cruise_02 %
%DUCT_PRESSURE       N/m^2         Engine Duct Pressure
%DUCT_PRESSURE_HW    N/m^2         Engine duct pressure for each hardware on the vehicle
%                             [Scramjet_01, TotalVehicle, WingBody_01]
%ENDURANCE_CRUISE    s             Flight time during cruise
%INSUFF_THRUST_CHECK               Check for inssufficient thrust in PM
%TRAJ_NSTEP                        Number of steps in current trajectory segment
%*************************************************************************
Variable.TRAJSEG.ConstantMachEnduranceCruise_02_PM_MD0008.INPUT.DUCT_PRESSURE = 0;
Variable.HW.TotalVehicle.PM.TotalVehicle_PM_MD0008.INPUT.DUCT_PRESSURE_HW = [0, 0, 0];
Variable.TRAJSEG.ConstantMachEnduranceCruise_02_PM_MD0008.INPUT.ENDURANCE_CRUISE = 439.11;
Variable.TRAJSEG.ConstantMachEnduranceCruise_02_PM_MD0008.INPUT.INSUFF_THRUST_CHECK = 'N';
Variable.TRAJSEG.ConstantMachEnduranceCruise_02_PM_MD0008.INPUT.TRAJ_NSTEP = 20;

% *************************************************************************
% Weight and Balance Disciplinary & Method Variables
% *************************************************************************

%Method: WB_MD0005   Hardware: TotalVehicle %*****************************
%AKVS            m^3/m^3       Volume of variable systems per total vehicle volume
%AKVTPS          m^3/m^3       Volume of vehicle TPS per total vehicle volume
%AKVV            m^3/m^3       Volume of vehicle void space per total vehicle volume
%AMUA                          Minimum OWE weight margin
%EBAND           m^-0.138      Error band around the structural fraction EBAND (+/- 0.049)
%FWSYS           kg/kg         Weight of variable systems per vehicle dry weight (FSYS in hypersonic convergence)
%RHO_CARGO       kg/m^3        Density of the cargo
%RHO_FUEL        kg/m^3        Density of fuel (formerly FUEL_DEN)
%RHO_OX          kg/m^3        Density of oxidizer (formerly OX_DEN)
%TIME_HYP        s             Total Time Flown at Hypersonic Mach Number
%VUN             m^3           Volume of unmanned fixed system
%WCARGO          N             Weight of cargo
%WUN             N             Weight of unmanned fixed systems (CUN in Hypersonic Convergence)
%*************************************************************************
Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.AKVS = 0.057995;
Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.AKVTPS = 0.013454;
Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.AKVV = 0.050495;
Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.AMUA = 0.107958;
Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.EBAND = 0.2040815;
Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.FWSYS = 0.060439;
Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.RHO_CARGO = 240;
Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.RHO_FUEL = 418.74752;
Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.RHO_OX = 1287.0;
% Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.TIME_HYP = ;
Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.VUN = 0.042758;
Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.WCARGO = 0;
Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.WUN = 133.356*9.81;


end
```

C.2 Results

Attribute	DefaultUnits	VariableName	GHV_5X
AKW	(blank)	Ratio of wetted surface area to planform area	2.1848
AL	m	Vehicle length	10.0116
AOA_T	degrees	Thrust incidence angle	0.675
BPLN	m	Span of the vehicle	3.3372
MDOT0_X	(blank)	Engine Massflow Rate Scale (MDOT0/10)	5.0032
SF	m^2	Frontal Area	1.7551
SFSPLN	(blank)	Ratio of frotal area to planform area	0.089944
SPLN_SF	(blank)	Planform Geometric Scale Factor	2.2397
SWET	m^2	Wetted surface area	42.6332
TAU	(blank)	Küchemann’s tau	0.06552
TAU_SF	(blank)	TAU Scale Factor	1.7876
VP	m^3	Volume of propulsion system	0.27975
VTOTAL	m^3	Volume of total vehicle	5.6478
AIP	kg/m^3	Propulsion index	657.998
AISTR	N/m^2	Structural Index	265.8671
OWE_V	N	Operational Weight Empty based on volume	22011.8962
OWE_W	N	Operational Weight Empty based on weights	22011.897
TOGW	N	Take-off Gross Weight	36020.188
AKVS	m^3/m^3	Volume of variable systems per total vehicle volume	0.057995
AKVTPS	m^3/m^3	Volume of vehicle TPS per total vehicle volume	0.013454
AKVV	m^3/m^3	Volume of vehicle void space per total vehicle volume	0.050495
AMUA	(blank)	Minimum OWE weight margin	0.10796
EBAND	m^-0.138	Error band around the structural fraction EBAND (+/- 0.049)	0.20408
FWSYS	kg/kg	Weight of variable systems per vehicle dry weight (FSYS in hypersonic convergence)	0.060439
RHO_CARGO	kg/m^3	Density of the cargo	240
RHO_FUEL	kg/m^3	Density of fuel (formerly FUEL_DEN)	418.7475
RHO_OX	kg/m^3	Density of oxidizer (formerly OX_DEN)	1287
VUN	m^3	Volume of unmanned fixed system	0.042758
WCARGO	N	Weight of cargo	0
WUN	N	Weight of unmanned fixed systems (CUN in Hypersonic Convergence)	1308.2224
AIP	kg/m^3	Propulsion index	657.998
AISTR	N/m^2	Structural Index	265.8671
AITPS	N/m^2	TPS Areal Weight (WTPS/SWET)	34.6318
AKSTR	m^-0.138	Structural correlation parameter i.e. structural fraction per unit surface area.	0.29722
AMZFW	N	Zero fuel weight	22011.897
OEW	N	Operational Empty Weight	22011.897

OWE	N	Operational Weight Empty	22011.897
OWE_V	N	Operational Weight Empty based on volume	22011.8962
OWE_W	N	Operational Weight Empty based on weights	22011.897
RHO_PPL	kg/m^3	Density of propellant	418.7475
TOGW	N	Take-off Gross Weight	36020.188
VFIX	m^3	Volume of fixed equipment	0.042758
VFUEL	m^3	Volume of fuel	3.4101
VOX	m^3	Volume of oxidizer	-1.0222E-14
VP	m^3	Volume of propulsion system	0.27975
VPAY	m^3	Volume of payload	0
VPPL	m^3	Volume of propellant	3.4101
VSTR	m^3	Volume of vehicle structural components	1.2265
VSYS	m^3	Volume of total systems	0.3703
VTOTAL	m^3	Volume of total vehicle	5.6478
VTPS	m^3	Volume of vehicle TPS	0.075985
VVOID	m^3	Volume of void space	0.28519
WFIX	N	Weight of fixed system (CSYS in Hypersonic Convergence)	1308.2224
WFUEL	N	Weight of fuel	14008.291
WMARGIN	N	Weight margin (OEW-WOPER-WSYS-WSTR-WP)	2144.8109
WOX	N	Weight of oxidizer	-4.1991E-11
WP	N	Weight of propulsion system	5893.7157
WPAY	N	Weight of payload	0
WPPL	N	Weight of propellant	14008.291
WSTR	N	Weight of structure	9858.3054
WSYS	N	Weight of systems	2638.5994
WTPS	N	Weight of Thermal Protection System	1476.4657
FF	(blank)	Fuel fraction	0.3889
THRL_VAR_MAX	(blank)	Maximum required fraction of max thrust for current hardware over the entire trajectory	1
WR	(blank)	Ratio of final mass to initial mass	1.6364
ALT_RES	(blank)	(blank)	2000
AOA_RES	(blank)	(blank)	2
DTR	/degrees	Conversion from degrees to radians	0.017453
G0	m/s^2	Gravitational acceleration at sealevel	9.81
MODE_DESIGN	(blank)	(blank)	6
RE	m	Radius of the Earth	6371000
RunNum	(blank)	(blank)	3
SPLN	m^2	Planform area	19.5137
SPLN_INIT	(blank)	(blank)	15.5
THRL_VAR_RES	(blank)	(blank)	0.25
V_RES	(blank)	(blank)	100

WS	N/m^2	Wing loading (i.e. TOGW/S)	1845.8968
WS_INIT	(blank)	(blank)	1799.591
F.1	(blank)	(blank)	-0.00081765
F.2	(blank)	(blank)	7.3909E-06
DUCT_PRESSURE	N/m^2	Engine Duct Pressure	0
ENDURANCE_CRUISE	s	Flight time during cruise	677.38
TRAJ_NSTEP	(blank)	Number of steps in current trajectory segment	20

Appendix D

GHV Adaptation CMDS

D.1 Input File

```
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%                  AVD_ABE Input File For GHV_Verification
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
function [Variable] = GHV_Verification (Variable)
% ************************************************************************
% ArchGen: GHV_Verification Control Variables
% ************************************************************************
%Set X-Vector Variable for FZERO solver %**********************************
% SPLN_INIT          m^2          Planform area
% WS_INIT            N/m^2        Wing loading (i.e. TOGW/S)
% X0                              Numerical values for X-Vector
%*************************************************************************
Variable.SYSPROC.INPUT.SPLN_INIT = 25;
Variable.SYSPROC.INPUT.WS_INIT = 1850.0;
Variable.SYSPROC.INPUT.X0 = [Variable.SYSPROC.INPUT.SPLN_INIT, Variable.SYSPROC.INPUT.WS_INIT];
%Multipoint Variation %***************************************************
% MODE_DESIGN                Design mode
%                    = 1  Analysis Points (Single), CMDS Optimization (No),  CMDS Convergence (No)
%                    = 2  Analysis Points (Single), CMDS Optimization (No),  CMDS Convergence (Yes)
%                    = 3  Analysis Points (Single), CMDS Optimization (Yes), CMDS Convergence (No)
%                    = 4  Analysis Points (Single), CMDS Optimization (Yes), CMDS Convergence (Yes)
%                    = 5  Analysis Points (Multi),  CMDS Optimization (No),  CMDS Convergence (No)
%                        Array Type (N X 1)
%                    = 6  Analysis Points (Multi),  CMDS Optimization (No),  CMDS Convergence (Yes)
%                        Array Type (N X 1)
%                    = 7  Analysis Points (Multi),  CMDS Optimization (Yes), CMDS Convergence (No)
%                        Array Type (N X 1)
%                    = 8  Analysis Points (Multi),  CMDS Optimization (Yes), CMDS Convergence (Yes)
%                        Array Type (N X 1)
%                    = 9  Analysis Points (Multi),  CMDS Optimization (No),  CMDS Convergence (No)
%                        Array Type (N X N)
%                    = 10 Analysis Points (Multi),  CMDS Optimization (No),  CMDS Convergence (Yes)
%                          Array Type (N X N)
%                      = 11 Analysis Points (Multi),  CMDS Optimization (Yes), CMDS Convergence (No)
%                          Array Type (N X N)
%                      = 12 Analysis Points (Multi),  CMDS Optimization (Yes), CMDS Convergence (Yes)
%                          Array Type (N X N)
% MV_NAMES                   Variables to be traded
% MV_init                    Initial value of trade variables
% MV_SS                      Variable step sizes
% MV_NS                      Number of Steps
% ***********************************************************************
Variable.SYSPROC.INPUT.MODE_DESIGN = 10;
Variable.SYSPROC.INPUT.MV_NAMES = { ...
'Variable.HW.TotalVehicle.GEO.TotalVehicle_GEO_MD0003.INPUT.TAU', ...
'Variable.TRAJSEG.ConstantMachEnduranceCruise_02_PM_MD0008.INPUT.ENDURANCE_CRUISE', ...
'Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.WCARGO', ...
% 'Variable.TRAJSEG.SteadyLevelTurn_01_PM_MD0011.INPUT.TRAJ_AN_MAX', ...
};
% Variable.SYSPROC.INPUT.MV_init = [0.05,700,0];
Variable.SYSPROC.INPUT.MV_init = [0.05,0,0];
% Variable.SYSPROC.INPUT.MV_SS = [0.01,140,3*500*4.4482];
Variable.SYSPROC.INPUT.MV_SS = [0.01,5,3*500*4.4482];
Variable.SYSPROC.INPUT.MV_NS = [3,5,0];
% Variable.SYSPROC.INPUT.MV_NS = [0,5,0];
% Variable.SYSPROC.INPUT.MV_POINTS = [0.067781039, 240.02, 2.402625, 11.7, 1656.878;
% 0.067370748, 458.87, 2.264747, 15.5, 1799.591;
% 0.065519560, 690.38, 2.242015, 19.5, 1869.527];
% Variable.SYSPROC.INPUT.MV_POINTS = [0.067781039, 240.02, 2.402625;
% 0.067370748, 458.87, 2.264747;
% 0.065519560, 690.38, 2.242015];
Variable.SYSPROC.INPUT.MV_POINTS = [0.067781039, 225.02, 2.402625;
0.067370748, 438.87, 2.264747;
0.065519560, 677.38, 2.242015];
```

```
% **************************************************************************
% Constants
% **************************************************************************
%Constant %*****************************************************************
%G0              m/s^2        Gravitational acceleration at sealevel
%DTR             /degrees     Conversion from degrees to radians
%RE              m            Radius of the Earth
%**************************************************************************
Variable.SYSPROC.INPUT.G0 = 9.81;
Variable.SYSPROC.INPUT.DTR = pi/180;
Variable.SYSPROC.INPUT.RE = 6371e3;
% **************************************************************************
% Look-Up Table Array Variables
% **************************************************************************
%Look-Up Table Input Arrays %*********************************************
%ALT_RANGE          m            Flight Altitude Range: [Start,End]
%ALT_RES            m            Flight Altitude Resolution
%V_RANGE            m/s          Flight Velocity Range: [Start,End]
%V_RES              m/s          Flight Velocity Resolution
%AOA_RANGE          m            Flight Altitude Range: [Start,End]
%AOA_RES            m            Flight Altitude Resolution
%THRL_VAR_RANGE     m            Flight Altitude Range: [Start,End]
%THRL_VAR_RES       m            Flight Altitude Resolution
%**************************************************************************
Variable.SYSPROC.INPUT.ALT_RANGE = [19000,25000];
Variable.SYSPROC.INPUT.ALT_RES = 2000;
Variable.SYSPROC.INPUT.V_RANGE = [1100,2100];
Variable.SYSPROC.INPUT.V_RES = 100;
Variable.SYSPROC.INPUT.AOA_RANGE = [-4.0,4.0];
Variable.SYSPROC.INPUT.AOA_RES = 2.0;
Variable.SYSPROC.INPUT.THRL_VAR_RANGE = [0.25,1.75];
Variable.SYSPROC.INPUT.THRL_VAR_RES = 0.25;
```

```
% ***************************************************************************
% Geometry Disciplinary & Method Variables
% ***************************************************************************
%Method: GEO_MD0003   Hardware: TotalVehicle %****************************
%AOA_T           degrees        Thrust incidence angle
%TAU                            Küchemann's tau
%***************************************************************************
Variable.HW.TotalVehicle.GEO.TotalVehicle_GEO_MD0003.INPUT.AOA_T = 0.675; %0.694301;
Variable.HW.TotalVehicle.GEO.TotalVehicle_GEO_MD0003.INPUT.TAU = 0.067370748;
% ***************************************************************************
% Aerodynamics Disciplinary & Method Variables
% ***************************************************************************
%Method: AERO_MD0008   Hardware: WingBody_01 %****************************
%ALD_K_FACT                      Lift / Drag Correction K Factor
%ALIND_ADD                       Additional Lift Induced Drag Factor (Typically = 0.1)
%AOA_CL0                         Zero Lift Angle of Attack
%CDTW_COR                        Transonic drag rise correction factor
%CLAS            /degree         Subsonic Lift Cureve Slope
%E_OS                            Oswalds Efficiency Factor
%ECDF                            Ratio of square of oswald efeciency factor to skin friction drag coefficient (e^2/CDF). (HYFAC
Vol 2pt2 fig 413 use 160, 200, 240, 280 for wing Body). 280 is recommed for very efficient vehicle
%TW_LIMIT        K               Wall Temperature Limit (Should be based on chosen material properties)
%***************************************************************************
Variable.HW.WingBody_01.AERO.WingBody_01_AERO_MD0008.INPUT.ALD_K_FACT = 0.785;
Variable.HW.WingBody_01.AERO.WingBody_01_AERO_MD0008.INPUT.ALIND_ADD = 0;
Variable.HW.WingBody_01.AERO.WingBody_01_AERO_MD0008.INPUT.AOA_CL0 = -2.892697;
Variable.HW.WingBody_01.AERO.WingBody_01_AERO_MD0008.INPUT.CDTW_COR = 0;
Variable.HW.WingBody_01.AERO.WingBody_01_AERO_MD0008.INPUT.CLAS = 0.025438;
Variable.HW.WingBody_01.AERO.WingBody_01_AERO_MD0008.INPUT.E_OS = 0.95;
Variable.HW.WingBody_01.AERO.WingBody_01_AERO_MD0008.INPUT.ECDF = 240;
Variable.HW.WingBody_01.AERO.WingBody_01_AERO_MD0008.INPUT.TW_LIMIT = 1750;
% ***************************************************************************
% Propulsion Disciplinary & Method Variables
% ************************************************************************
%Method: PROP_MD0006   Hardware: Scramjet_01 %****************************
%PHI_FUEL_REF                     Reference Fuel Equivalence Ratio
%************************************************************************
Variable.HW.Scramjet_01.PROP.Scramjet_01_PROP_MD0006.INPUT.PHI_FUEL_REF = 1.2;
% ************************************************************************
% Performance Matching Disciplinary & Method Variables
% ************************************************************************
%Performance Matching Disciplinary Process Input Variables%****************
%TRAJ_ALT_V_START    m             Start Point For Vector of altitudes
%TRAJ_FF_V_START                   Start Point For Vector of Fuel fractions
%TRAJ_GAM_V_START    degrees       Start Point For Vector of flight path angles
%TRAJ_PSI_V_START    degrees       Start Point For Vector of heading angles
%TRAJ_RANGE_V_START  m             Start Point For Vector of total range
%TRAJ_TIME_V_START   s             Start Point For Vector of trajetory time
%TRAJ_TRAJSEG_V_START              Start Point For Vector of current flight segment string
%TRAJ_V_V_START      m/s           Start Point For Vector of vel
%TRAJ_WR_V_START                   Start Point For Vector of ratios of final mass at each point in the trajectory to init
%TRAJ_X_V_START      m             Start Point For Vector of position in x-directio
%TRAJ_Y_V_START      m             Start Point For Vector of position in y-directio
%************************************************************************
Variable.MISSION.INPUT.TRAJ_ALT_V_START = 19054.267;
Variable.MISSION.INPUT.TRAJ_FF_V_START = 0.0;
Variable.MISSION.INPUT.TRAJ_GAM_V_START = 0.0;
Variable.MISSION.INPUT.TRAJ_PSI_V_START = 0;
Variable.MISSION.INPUT.TRAJ_RANGE_V_START = 0.0;
Variable.MISSION.INPUT.TRAJ_TIME_V_START = 0.0;
Variable.MISSION.INPUT.TRAJ_TRAJSEG_V_START = {'START'};
Variable.MISSION.INPUT.TRAJ_V_V_START = 1180.27704;
Variable.MISSION.INPUT.TRAJ_WR_V_START = 1;
Variable.MISSION.INPUT.TRAJ_X_V_START = 0;
Variable.MISSION.INPUT.TRAJ_Y_V_START = 0;
```

```
%Method: PM_MD0009   Trajectory Segment: Booster Separation_01 %***********
%TRAJ_NSTEP                     Number of steps in current trajectory segment
%TRAJ_WR                        Ratio of final mass to initial mass for trajectory segment
%*************************************************************************
Variable.TRAJSEG.BoosterSeparation_01_PM_MD0009.INPUT.TRAJ_NSTEP = 1;
Variable.TRAJSEG.BoosterSeparation_01_PM_MD0009.INPUT.TRAJ_WR = 1;
%Method: PM_MD0003   Trajectory Segment: Constant Q Climb_01 %*************
%DUCT_PRESSURE      N/m^2        Engine Duct Pressure
%DUCT_PRESSURE_HW   N/m^2        Engine duct pressure for each hardware on the vehicle
%                               [Scramjet_01, TotalVehicle, WingBody_01]
%INSUFF_THRUST_CHECK            Check for inssufficient thrust in PM
%TRAJ_ALT_END                   Altitude desired at the end of the trajectory segment
%TRAJ_AN_MAX        g's          Maximum acceleration allowed for current trajectory segment
%TRAJ_AN_MIN        g's          Minimum acceleration allowed for current trajectory segment
%TRAJ_NSTEP                     Number of steps in current trajectory segment
%*************************************************************************
Variable.TRAJSEG.ConstantQClimb_01_PM_MD0003.INPUT.DUCT_PRESSURE = 0;
Variable.HW.TotalVehicle.PM.TotalVehicle_PM_MD0003.INPUT.DUCT_PRESSURE_HW = [0, 0, 0];
Variable.TRAJSEG.ConstantQClimb_01_PM_MD0003.INPUT.INSUFF_THRUST_CHECK = 'N';
Variable.TRAJSEG.ConstantQClimb_01_PM_MD0003.INPUT.TRAJ_ALT_END = 24235;
Variable.TRAJSEG.ConstantQClimb_01_PM_MD0003.INPUT.TRAJ_AN_MAX = 2.0;
Variable.TRAJSEG.ConstantQClimb_01_PM_MD0003.INPUT.TRAJ_AN_MIN = 0.15;
Variable.TRAJSEG.ConstantQClimb_01_PM_MD0003.INPUT.TRAJ_NSTEP = 20;
%Method: PM_MD0008   Trajectory Segment: Constant Mach Endurance Cruise_01 %
%DUCT_PRESSURE      N/m^2        Engine Duct Pressure
%DUCT_PRESSURE_HW   N/m^2        Engine duct pressure for each hardware on the vehicle
%                               [Scramjet_01, TotalVehicle, WingBody_01]
%ENDURANCE_CRUISE   s            Flight time during cruise
%INSUFF_THRUST_CHECK            Check for inssufficient thrust in PM
%TRAJ_NSTEP                     Number of steps in current trajectory segment
%*************************************************************************
Variable.TRAJSEG.ConstantMachEnduranceCruise_01_PM_MD0008.INPUT.DUCT_PRESSURE = 0;
Variable.HW.TotalVehicle.PM.TotalVehicle_PM_MD0008.INPUT.DUCT_PRESSURE_HW = [0, 0, 0];
Variable.TRAJSEG.ConstantMachEnduranceCruise_01_PM_MD0008.INPUT.ENDURANCE_CRUISE = 0.01;
Variable.TRAJSEG.ConstantMachEnduranceCruise_01_PM_MD0008.INPUT.INSUFF_THRUST_CHECK = 'N';
Variable.TRAJSEG.ConstantMachEnduranceCruise_01_PM_MD0008.INPUT.TRAJ_NSTEP = 20;
%Method: PM_MD0011   Trajectory Segment: Steady Level Turn_01 %************
%DUCT_PRESSURE       N/m^2         Engine Duct Pressure
%DUCT_PRESSURE_HW    N/m^2         Engine duct pressure for each hardware on the vehicle
%                              [Scramjet_01, TotalVehicle, WingBody_01]
%INSUFF_THRUST_CHECK               Check for inssufficient thrust in PM
%TRAJ_AN_MAX         g's           Maximum acceleration allowed for current trajectory segment
%TRAJ_NSTEP                        Number of steps in current trajectory segment
%TRAJ_PSI_TURN       degrees       Angle to change heading by
%****************************************************************************
Variable.TRAJSEG.SteadyLevelTurn_01_PM_MD0011.INPUT.DUCT_PRESSURE = 0;
Variable.HW.TotalVehicle.PM.TotalVehicle_PM_MD0011.INPUT.DUCT_PRESSURE_HW = [0, 0, 0];
Variable.TRAJSEG.SteadyLevelTurn_01_PM_MD0011.INPUT.INSUFF_THRUST_CHECK = 'N';
Variable.TRAJSEG.SteadyLevelTurn_01_PM_MD0011.INPUT.TRAJ_AN_MAX = 2.242015;
Variable.TRAJSEG.SteadyLevelTurn_01_PM_MD0011.INPUT.TRAJ_NSTEP = 20;
Variable.TRAJSEG.SteadyLevelTurn_01_PM_MD0011.INPUT.TRAJ_PSI_TURN = 180.0;
%Method: PM_MD0008   Trajectory Segment: Constant Mach Endurance Cruise_02 %
%DUCT_PRESSURE       N/m^2         Engine Duct Pressure
%DUCT_PRESSURE_HW    N/m^2         Engine duct pressure for each hardware on the vehicle
%                              [Scramjet_01, TotalVehicle, WingBody_01]
%ENDURANCE_CRUISE    s             Flight time during cruise
%INSUFF_THRUST_CHECK               Check for inssufficient thrust in PM
%TRAJ_NSTEP                        Number of steps in current trajectory segment
%****************************************************************************
Variable.TRAJSEG.ConstantMachEnduranceCruise_02_PM_MD0008.INPUT.DUCT_PRESSURE = 0;
Variable.HW.TotalVehicle.PM.TotalVehicle_PM_MD0008.INPUT.DUCT_PRESSURE_HW = [0, 0, 0];
Variable.TRAJSEG.ConstantMachEnduranceCruise_02_PM_MD0008.INPUT.ENDURANCE_CRUISE = 439.11;
Variable.TRAJSEG.ConstantMachEnduranceCruise_02_PM_MD0008.INPUT.INSUFF_THRUST_CHECK = 'N';
Variable.TRAJSEG.ConstantMachEnduranceCruise_02_PM_MD0008.INPUT.TRAJ_NSTEP = 20;
```

```
% *************************************************************************
% Weight and Balance Disciplinary & Method Variables
% *************************************************************************
%Method: WB_MD0005   Hardware: TotalVehicle %*****************************
%AKVS          m^3/m^3      Volume of variable systems per total vehicle volume
%AKVTPS        m^3/m^3      Volume of vehicle TPS per total vehicle volume
%AKVV          m^3/m^3      Volume of vehicle void space per total vehicle volume
%AMUA                       Minimum OWE weight margin
%EBAND         m^-0.138     Error band around the structural fraction EBAND (+/- 0.049)
%FWSYS         kg/kg        Weight of variable systems per vehicle dry weight (FSYS in hypersonic convergence)
%RHO_CARGO     kg/m^3       Density of the cargo
%RHO_FUEL      kg/m^3       Density of fuel (formerly FUEL_DEN)
%RHO_OX        kg/m^3       Density of oxidizer (formerly OX_DEN)
%TIME_HYP      s            Total Time Flown at Hypersonic Mach Number
%VUN           m^3          Volume of unmanned fixed system
%WCARGO        N            Weight of cargo
%WUN           N            Weight of unmanned fixed systems (CUN in Hypersonic Convergence)
%*************************************************************************
Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.AKVS = 0.057995;
Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.AKVTPS = 0.013454;
Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.AKVV = 0.050495;
Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.AMUA = 0.107958;
Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.EBAND = 0.2040815;
Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.FWSYS = 0.060439;
Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.RHO_CARGO = 240;
Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.RHO_FUEL = 418.74752;
Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.RHO_OX = 1287.0;
% Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.TIME_HYP = ;
Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.VUN = 0.042758;
Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.WCARGO = 0;
Variable.HW.TotalVehicle.WB.TotalVehicle_WB_MD0005.INPUT.WUN = 133.356*9.81;
end
```

D.2 Results

Attribute	DefaultUnits	VariableName	GHV_5X
AKW	(blank)	Ratio of wetted surface area to planform area	2.1848
AL	m	Vehicle length	10.0779
ALLE	degrees	Sweep angle of the leading edge	80
AOA_T	degrees	Thrust incidence angle	0.675
BPLN	m	Span of the vehicle	3.3593
DIA_BODY	m	Diameter of Body	1.079
MDOT0_X	(blank)	Engine Massflow Rate Scale (MDOT0/10)	5.0641
SF	m^2	Frontal Area	1.7785
SFSPLN	(blank)	Ratio of frotal area to planform area	0.089944
SPLN_SF	(blank)	Planform Geometric Scale Factor	2.2546
SWET	m^2	Wetted surface area	43.1997
TAU	(blank)	Küchemann's tau	0.06552
TAU_SF	(blank)	TAU Scale Factor	1.7876
VP	m^3	Volume of propulsion system	0.28367
VTOTAL	m^3	Volume of total vehicle	5.7607
AIP	kg/m^3	Propulsion index	667.4388
AISTR	N/m^2	Structural Index	240.984
OWE_V	N	Operational Weight Empty based on volume	22776.5687
OWE_W	N	Operational Weight Empty based on weights	22776.5691
TOGW	N	Take-off Gross Weight	37066.4653
AKVS	m^3/m^3	Volume of variable systems per total vehicle volume	0.057995
AKVTPS	m^3/m^3	Volume of vehicle TPS per total vehicle volume	0.013454
AKVV	m^3/m^3	Volume of vehicle void space per total vehicle volume	0.050495
AMUA	(blank)	Minimum OWE weight margin	0.10796
EBAND	m^-0.138	Error band around the structural fraction EBAND (+/- 0.049)	0.002838
FWSYS	kg/kg	Weight of variable systems per vehicle dry weight (FSYS in hypersonic convergence)	0.060439
RHO_CARGO	kg/m^3	Density of the cargo	240
RHO_FUEL	kg/m^3	Density of fuel (formerly FUEL_DEN)	418.7475
RHO_OX	kg/m^3	Density of oxidizer (formerly OX_DEN)	1287
VUN	m^3	Volume of unmanned fixed system	0.042758
WCARGO	N	Weight of cargo	2.242
WUN	N	Weight of unmanned fixed systems (CUN in Hypersonic Convergence)	1308.2224
AIP	kg/m^3	Propulsion index	667.4388
AISTR	N/m^2	Structural Index	240.984
AITPS	N/m^2	TPS Areal Weight (WTPS/SWET)	34.6001
AKSTR	m^-0.138	Structural correlation parameter i.e. structural fraction per unit surface area.	0.30281

AMZFW	N	Zero fuel weight	22778.8111
OEW	N	Operational Empty Weight	22774.3271
OWE	N	Operational Weight Empty	22776.5691
OWE_V	N	Operational Weight Empty based on volume	22776.5687
OWE_W	N	Operational Weight Empty based on weights	22776.5691
RHO_PPL	kg/m^3	Density of propellant	418.7475
TOGW	N	Take-off Gross Weight	37066.4653
VFIX	m^3	Volume of fixed equipment	0.042758
VFUEL	m^3	Volume of fuel	3.4786
VOX	m^3	Volume of oxidizer	-1.1586E-15
VP	m^3	Volume of propulsion system	0.28367
VPAY	m^3	Volume of payload	0.00095227
VPPL	m^3	Volume of propellant	3.4786
VSTR	m^3	Volume of vehicle structural components	1.2522
VSYS	m^3	Volume of total systems	0.37685
VTOTAL	m^3	Volume of total vehicle	5.7607
VTPS	m^3	Volume of vehicle TPS	0.077505
VVOID	m^3	Volume of void space	0.29089
WFIX	N	Weight of fixed system (CSYS in Hypersonic Convergence)	1308.2224
WFUEL	N	Weight of fuel	14289.8963
WMARGIN	N	Weight margin (OEW-WOPER-WSYS-WSTR-WP)	2219.1011
WOX	N	Weight of oxidizer	-4.7595E-12
WP	N	Weight of propulsion system	5965.3843
WPAY	N	Weight of payload	2.242
WPPL	N	Weight of propellant	14289.8963
WSTR	N	Weight of structure	10410.445
WSYS	N	Weight of systems	2684.6799
WTPS	N	Weight of Thermal Protection System	1494.7168
FF	(blank)	Fuel fraction	0.38552
THRL_VAR_MAX	(blank)	Maximum required fraction of max thrust for current hardware over the entire trajectory	1
WR	(blank)	Ratio of final mass to initial mass	1.6274
ALT_RES	(blank)	(blank)	2000
AOA_RES	(blank)	(blank)	2
DTR	/degrees	Conversion from degrees to radians	0.017453
G0	m/s^2	Gravitational acceleration at sealevel	9.81
MODE_DESIGN	(blank)	(blank)	6
RE	m	Radius of the Earth	6371000
RunNum	(blank)	(blank)	3
SPLN	m^2	Planform area	19.773
SPLN_INIT	(blank)	(blank)	15.5

THRL_VAR_RES	(blank)	(blank)	0.25
V_RES	(blank)	(blank)	100
WS	N/m^2	Wing loading (i.e. TOGW/S)	1874.6046
WS_INIT	(blank)	(blank)	1799.591
F.1	(blank)	(blank)	-0.00036647
F.2	(blank)	(blank)	9.3703E-06
DUCT_PRESSURE	N/m^2	Engine Duct Pressure	0
ENDURANCE_CRUISE	s	Flight time during cruise	677.38
TRAJ_NSTEP	(blank)	Number of steps in current trajectory segment	20

References

Bar-Yam, Yaneer. 1997. *Dynamics of Complex Systems*. Reading, Mass: Addison-Wesley.

Bradford, John Edward. 2001. "A Technique for Rapid Prediction of Aftbody Nozzle Performance for Hypersonic Launch Vehicle Design.".

Chudoba, Bernd. 2001. "Stability and Control of Conventional and Unconventional Aircraft Configurations : A Generic Approach."Books on Demand.

Coleman, Gary John. 2010. "Aircraft Conceptual Design - an Adaptable Parametric Sizing." PhD, University of Texas at Arlington.

Corning, Gerald. 1976. *Supersonic and Subsonic, CTOL and VTOL, Airplane Design*. College Park, Md.: Corning.

Czysz, P. 2004*. Hypersonic Convergence - Volume 1*. Dayton, Ohio: Air Force Research Laboratory.

Davies, C. and Soremekun, G. "Phoenix Integration and the Skunk Works® A History of Success, A Path to the Future." Phoenix Integration Inc, http://www.phoenix-int.com/resources/webinars/2015/history-of-success.php.

Finck, R. D., D. E. Hoak, and Douglas Aircraft Company. 1978. *USAF Stability and Control Datcom*. Dayton, Ohio: Flight Control Division, Air Force Dynamics Laboratory, Wright-Patterson Air Force Base.

Haney, Eric. 2016. "Data Engineering in Aerospace Systems Design & Forecasting." Ph.D. Aerospace Engineering, The University of Texas at Arlington.

Heinze, Wolfgang. 1994. "Ein Beitrag Zur Quantitativen Analyse Der Technischen Und Wirtschaftlichen Auslegungsgrenzen Verschiedener Flugzeugkonzepte Für Den Transport Grosser Nutzlasten."Inst. für Flugzeugbau und Leichtbau, Techn. Univ.

Howe, Denis. 2000. *Aircraft Conceptual Design Synthesis*. London: Professional Engineering Pub.

Huang, Xiao. 2006. "A Prototype Computerized Synthesis Methodology for Generic Space Access Vehicle (SAV) Conceptual Design.".

IEEE. 2007. *ISO/IEC 26702 IEEE Std 1220-2005 First Edition 2007-07-15 ISO/IEC Standard for Systems Engineering - Application and Management of the Systems Engineering Process* Institute of Electrical and Electronics Engineers Incorporated.

Jackson, Scott. 1997. *Systems Engineering for Commercial Aircraft*. Aldershot, England; Brookfield, Vt., USA: Ashgate.

JAYARAM, S., A. MYKLEBUST, and P. GELHAUSEN. 1992. "ACSYNT - A Standards-Based System for Parametric, Computer Aided Conceptual Design of Aircraft." In : American Institute of Aeronautics and Astronautics. doi:doi:10.2514/6.1992-1268.

Jenkinson, Lloyd R., Simpkin, Paul.,Rhodes, Darren,,. 1999. *Civil Jet Aircraft Design*. Reston, VA: American Insitute of Aeronautics and Astronautics.

Kline, S. J. 1995. *Conceptual Foundations for Multidisciplinary Thinking*. Stanford, Calif.: Stanford University Press.

Kockler, F., T. Withers, J. Poodiack, M. Gierman, and DEFENSE SYSTEMS MANAGEMENT COLL FORT BELVOIR VA. 1990. *Systems Engineering Management Guide*.

Kroo, I. 2006. "Aircraft Design: Synthesis and Analysis." .

Kumar, Bharat, Dale De Remer, and Douglas M. Marshall J.D. 2005. *An Illustrated Dictionary of Aviation*. New York: McGraw-Hill.

Loftin, Laurence K. 1980. *Subsonic Aircraft : Evolution and the Matching of Size to Performance*. Washington, D.C.; [Springfield, Va.]: National Aeronautics and Space Administration, Scientific and Technical Branch.

Lovell, D. A. 1980. "Some Experiences with Numerical Optimization in Aircraft Specification and Preliminary Design Studies."ICAS, Paper 80-2.4, .

McCullers, L. A. 1987*. Aircraft Configuration Optimization Including Optimized Flight Profiles*. Hampton, Virginia: Kentron International, Inc.

McMillan, B. 1964. "Aerospace Management." *Aerospace Management.*: pp. 62.

Miele, Angelo. 1962. *Flight Mechanics*. Reading, Mass: Addison-Wesley Pub. Co.

MINZNER, R. A., K. S. CHAMPION, H. L. POND, and AIR FORCE CAMBRIDGE RESEARCH LABS HANSCOM AFB MA. 1959. *The Ardc Model Atmosphere, 1959*.

MSDN. "Visual Basic: Windows Controls - TreeView Control." Microsoft Developers Network2016, https://msdn.microsoft.com/en-us/library/aa443492(v=vs.60).aspx.

NASA. 2007*. NASA Systems Engineering Handbook*.

NASA and Zell, H. "Earth's Atmoshperic Layers." NASA, last modified July 15, 20152016, http://www.nasa.gov/mission_pages/sunearth/science/atmosphere-layers2.html.

Nicolai, Leland M. and Carichner, Grant. "Fundamentals of Aircraft and Airship Design Volume 1." American Institute of Aeronautics and Astronautics.

OFFICE OF THE UNDER SECRETARY OF DEFENSE. 1997. *Joint Modeling and Simulation System (JMASS), Joint Initial Requirements Document (JIRD)*.

Omoragbon, Amen. 2010. *An Integration of A Modern Flight Control System Design Technique into A Conceptual Design Stability and Controls Tool, AeroMech* Aerospace Engineering.

Perez, RubenE, PeterW Jansen, and JoaquimR R. A. Martins. 2012. "pyOpt: A Python-Based Object-Oriented Framework for Nonlinear Constrained Optimization." *Structural and Multidisciplinary Optimization* 45 (1): 101-118. doi:10.1007/s00158-011-0666-3.

Petty, M. and Eric Werner Weisel. 2003. "A Composability Lexicon." Orlando Fl, Simulation Interoperability Standards Organization, March 30 - April 4.

Raymer, Daniel P. 1999. *Aircraft Design : A Conceptual Approach*. Reston, VA: American Institute of Aeronautics and Astronautics.

Roskam, J. and W. Anematt. 1991. "AAA (Advanced Aircraft Analysis): A User-Friendly Approach to Preliminary Aircraft Design."ICAS, .

Roskam, Jan. 2004. *Airplane Design*. Lawrence (Kansas): DARcorporation.

Ruttle, B., J. Stork, and G. Liston. 2012*. Generic Hypersonic Vehicles for Conceptual Design Analyses*. Wright-Patterson AFB, OH: AFLR/RQHT.

Ryan, A. 2007. "A Multidisciplinary Approach to Complex Systems Design." PhD, University of Adelaide, School of Mathematical Sciences.

Schaufele, Roger D. 2000. *The Elements of Aircraft Preliminary Design*.

Sforza, P. M. 2016. *Manned Spacecraft Design Principles*.

Shaw, Mary. 1995. "Architectural Issues in Software Reuse: It's Not just the Functionality, it's the Packaging." *ACM SIGSOFT Software Engineering Notes* 20 (SI): 3-6. doi:10.1145/223427.211783.

SISO. 2004. *Base Object Model (BOM) Template Specification Volume I - Interface BOM*. SISO-STD-003.1-2003-DRAFTV0.7 ed.

Stinton, Darrol. 1998. *The Anatomy of the Airplane*. Reston, VA; Oxford, UK: American Institute of Aeronautics and Astronautics ; Blackwell Science.

Szabo, C. and Yong Meng Teo. 2007. "On Syntactic Composability and Model Reuse."IEEE, . doi:10.1109/AMS.2007.74.

Torenbeek, Egbert. 2013. *Advanced Aircraft Design : Conceptual Design, Analysis, and Optimization of Subsonic Civil Airplanes*.

———. 1982. *Synthesis of Subsonic Airplane Design : An Introduction to the Preliminary Design, of Subsonic General Aviation and Transport Aircraft, with Emphasis on Layout, Aerodynamic Design, Propulsion, and Performance*. Delft; The Hague; Hingham, MA: Delft University Press ; Nijhoff ; Sold and distributed in the U.S. and Canada by Kluwer Boston.

United States., Air Force.,Systems Command.,. 1965. *Space Planners Guide.* [Washington, D.C.]: U.S. G.P.O.

United States., Department of Defense. 1974. *Engineering Management.* Washington, D.C.: Department of Defense, United States of America.

Vinh, Nguyen X. 1981a. *Optimal Trajectories in Atmospheric Flight*. Vol. 2. Amsterdam; New York: Elsevier Scientific Pub. Co.

———. 1981b. *Optimal Trajectories in Atmospheric Flight*. Vol. 2. Amsterdam; New York: Elsevier Scientific Pub. Co.

Vought, Aero. 1985*. Aircraft Synthesis Analysis Program Description Volumes II - IX*. Dallas, Texas: LTV Aerospace and Defense, Vought Aero Products Division.

Weisel, Eric Werner. 2004. "Models, Composability, and Validity." PhD, Old Dominion University.

Wittman, Jr, L. Robert, and Cynthia T. Harrison. 2001. *OneSAF: A Product Line Approach to Simulation Development*.

Wood, Karl Dawson. 1963. *Aerospace Vehicle Design : Vol. 1. Aircraft Design.* [Place of publication not identified]: Johnson.

Biographical Information

Lex Gonzalez graduated with a Degree in Aerospace Engineering from the University of Texas at Arlington in 2009. He joined the Aerospace Vehicle Design Laboratory during his final undergraduate semester. During this time he was involved in the conceptual design vehicles for the NASA LaRC future commercial transport program, supported the conceptual design of Mach 6 to 8 endurance cruise vehicles for NASA Vehicle Analysis Branch, and was a member of the team working on architectural design for a manned geo satellite servicing vehicle.

He also served two fellowships working with the Air Force Research Laboratory (AFRL). The first as a member of the AFRL Space Scholars Program at Kirtland Air Force Base, where he worked at the Space Vehicles Directorate on the increasing the capability of an in-house satellite modelling and simulation code. The second, as part of the AFRL Summer Faculty Fellowship Program at Wright-Patterson Air Force Base, where he helped with the technology forecasting and conceptual design of an hypersonic endurance cruise vehicle.