AFRL-RQ-WP-TR-2013-0030

STRUCTURAL TECHNOLOGY EVALUATION AND ANALYSIS PROGRAM (STEAP)

Delivery Order 0035: Dynamics and Control and Computational Design of Flapping Wing Micro Air Vehicles

David O. Sigthorsson and Isaac Weintraub

General Dynamics Information Technology

Christopher Smithson

Data Science Automation

OCTOBER 2012
Final Report

AIR FORCE RESEARCH LABORATORY
AEROSPACE SYSTEMS DIRECTORATE
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7542
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

NOTICE AND SIGNATURE PAGE

*//Signature//
MICHAEL W. OPPENHEIMER
Project Manager
Control Automation Branch
Power and Control Division

//Signature//
PHILIP S. BERAN
Design and Analysis Branch
Aerospace Vehicles Division

//Signature//
JEFFREY C. TROMP, Chief
Control Automation Branch
Power and Control Division
Aerospace Systems Directorate

//Signature//
DANIEL B. THOMPSON, Principal Scientist
Power and Control Division
Aerospace Systems Directorate

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. REPORT DATE ***(DD-MM-YY)*** October 2012	**2. REPORT TYPE** Final	**3. DATES COVERED** ***(From - To)*** 31 December 2010 – 07 October 2012

4. TITLE AND SUBTITLE STRUCTURAL TECHNOLOGY EVALUATION AND ANALYSIS PROGRAM (STEAP) Delivery Order 0035: Dynamics and Control and Computational Design of Flapping Wing Micro Air Vehicles	**5a. CONTRACT NUMBER** FA8650-04-D-3446-0035
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 62201F
6. AUTHOR(S) David O. Sigthorsson and Isaac Weintraub (General Dynamics Information Technology) Christopher Smithson (Data Science Automation)	**5d. PROJECT NUMBER** 2403
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER Q079
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES): General Dynamics Information Technology, 5100 Springfield Pike, Suite 509, Dayton, OH 45431; Data Science Automation, 375 Valley Brook Road, McMurray, PA 15317	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory, Aerospace Systems Directorate, Wright-Patterson Air Force Base, OH 45433-7542, Air Force Materiel Command, United States Air Force	**10. SPONSORING/MONITORING AGENCY ACRONYM(S)** AFRL/RQQA, AFRL/RQVC
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S) AFRL-RQ-WP-TR-2013-0030

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES
PA Case Number: 88ABW-2013-1005; Clearance Date: 01 Mar 2013. Report contains color.

14. ABSTRACT
The task objectives were to: understand the flight dynamics of flapping-wing micro air vehicles (FWMAVs), design and test flight control laws on flapping FWMAV prototypes that enable controlled flight with the ability to hover and maneuver forward/backward and sideways. The effort was aimed at developing a knowledge base that will lead to intelligence, surveillance, and reconnaissance (ISR) platforms that can hide in plain sight, reject disturbances and operate in cluttered/confined spaces. The related key science issues were to develop a control-oriented model of the aerodynamics and structure; design and implementation of a control theory, where the control forces/moments are constrained to a periodic form, and to use these periodic forces to precisely direct ISR sensors in cluttered/confined spaces in the presence of disturbances such as gusts and steady winds.

15. SUBJECT TERMS
intelligence, surveillance, and reconnaissance (IRS), flapping wing micro air vehicles (FWMAV), degrees of freedom (DOF)

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT:**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** (Monitor)
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			
Unclassified	Unclassified	Unclassified	SAR	80	Michael W. Oppenheimer (AFRL/RQQA) Philip S. Beran (AFRL/RQVC) **19b. TELEPHONE NUMBER** *(Include Area Code)* N/A

TABLE OF CONTENTS

Section **Page**

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

1.0 SUMMARY

OBJECTIVE I

The task objectives were to: understand the flight dynamics of flapping-wing micro air vehicles (FWMAVs), and design and test flight control laws on flapping FWMAV prototypes that enable controlled flight with the ability to hover and maneuver forward/backward and sideways. The effort was aimed at developing a knowledge base that will lead to Intelligence, Surveillance, and Reconnaissance (ISR) platforms that can hide in plain sight, reject disturbances and operate in cluttered/confined spaces. The related key science issues were to develop a control-oriented model of the aerodynamics and structure; design and implementation of a control theory, where the control forces/moments are constrained to a periodic form, and to use these periodic forces to precisely direct ISR sensors in cluttered/confined spaces in the presence of disturbances such as gusts and steady winds.

The focus of the research effort shifted as work was performed on the task. Initial efforts were focused on theoretical and computer based simulation work on modeling and control design for flapping wing micro air vehicles (FWMAV). In collaboration with on-site researchers, we successfully devised simulation models based on first-principles or quasi-steady models. Moreover, control oriented models were also developed. The results are presented in numerous publications listed in Section 4. These models were used for control design resulting in controllers using a variety of control inputs including: split-cycle frequency modulation, stroke amplitude and bias modulation, and angle-of-attack modulation. The controllers were successfully tested, maintaining waypoint tracking when applied to six degree-of-freedom (DOF) simulation models.

Later the work focused on producing prototype vehicles, particularly addressing verification of the split-cycle modulation for control purposes. Emphasis was placed on building a small bird size flapping wing vehicle with two independently actuated wings. Such a vehicle was built. With an iterative design and fabrication process, the mechanical and electrical design was improved to produce a satisfactory prototype. The resulting vehicles were tested on both an electronic sensor force balance and an air-table. The force balance experiments produced somewhat inconclusive results due to signal to noise ratio issues with the available sensors. The air-table experiment has been a milestone success, showing clearly a proof of concept for the split-cycle wingbeat velocity modulation approach for control.

OBJECTIVE II

The Flapping Sciences Integration (FSI) Project was initiated in 2008 as part of a larger Micro Air Vehicle (MAV) Hover Science Program. The FSI effort, a collaborative activity between industry, government, and academia, had three inter-related components: FSI-interface, FSI-methods, and FSI-controls. This report describes FSI-interface, a contractual activity between Air Force, Data Science Automation (DSA), and General Dynamics Information Technology (GDIT). The contract representing FSI-interface was FA8650-04-D-3446, Task Order 35, on which GDIT served as prime contractor, and DSA served as sub-contractor. The work carried out in support of FSI-interface was conducted on-site by DSA at Air Force Research Laboratory

in the Multidisciplinary Science and Technology Center (MSTC), denoted by office symbol AFRL/RQVC.

The primary technical goal of FSI was to develop an extensible, multidisciplinary design methodology for the AFRL MAV team, its collaborators, and its potential customers. The design methodology was anticipated to be experimental in nature and integrate new design analysis capabilities developed in FSI-methods, an in-house basic research activity supported by Air Force Office of Scientific Research (Drs Philip Beran and Richard Snyder, PIs). FSI-interface also was intended to incorporate methods developed in FSI-controls, an add-on project to the MSTC Collaborative Center for Multidisciplinary Science (CCMS) led by Professors Mayur Patil and Craig Woolsey. In short, the role of FSI-interface was to integrate and transition fundamental research in two other project areas for the purpose of future MAV design.

A separate project to bridge between FSI and MAV Hover Sciences is Multi-Physics Prototyping (MPP). Like FSI, MPP was a technical effort executed primarily in-house. MPP was intended to be a collaborative activity with two components: MPP-support and MPP-QTA. MPP-QTA was to be a Quantitative Technical Assessment of flapping technology for MAV (to be carried out external to AFRL/ RQSE), and was to use the tools developed under FSI. Unfortunately, this project was cancelled prior to contract award. However, MPP-support, following a separate track, completed. The contract representing MPP-support was FA8650-04-D-3446, Task Order 49; again, GDIT served as prime contractor, and DSA served as sub-contractor. MPP-support was designed to provide technical support to MPP-QTA, including FSI methodology tool deployment, software support, trouble-shooting, and liaise with an internal MSTC QTA effort. With the cancellation of MPP-QTA, the MPP-support activity was re-focused to provide internal support for use of FSI methodology tools and to apply them to design cases as part of a "QTA-lite" study. This downsized QTA activity was strongly coupled with the FSI-controls work, including development of gust-based design scenarios.

This document provides an architect's perspective of FSI-interface and related software, including:

- Architecture of the top-level user-interface: Multidisciplinary Multi-fidelity Model-based Computational Tool (M^3CT)

- Descriptions of the SORCER Service-Oriented Architecture that M^3CT is built on, including documentation about how-to-construct SORCER "jobs" and "exertions" (e.g., how to run a M^3CT study)

- FSI-methods and FSI-controls software modules ("providers") integrated within this project via M^3CT and SORCER Nomenclature

- Lessons learned during the project that could impact the future architectural development of multidisciplinary design software

- Installation notes

Elements of this document will be repeated in the FA8650-04-D-3446, Task Order 35, Final Report, along with elements of the FSI-interface Developer's Guide and MPP-support's Final Report. The FSI-interface Final Report will serve as a concise summary of technical work that appropriately references other source material containing more detailed descriptions. The technical details contained herein are documented to enable maintenance of FSI software, to create an enduring knowledge base of FSI software, and to enable the future development of innovative multidisciplinary design software.

The Air Force initiator and technical point-of-contact for this project is Dr. Phil Beran (AFRL/RQVC). The FSI-Interface programmer was Mr. Michael Robbeloth (DSA) from project initiation to May 2012; Mr. Christopher Smithson (DSA) succeeded Mr. Robbeloth in June 2012 to develop project documentation, improve software integrity, and provide consultation on M^{3}CT architectural weaknesses and alternative architectural concepts for future design software. Mr. Robbeloth and Mr. Smithson were both supervised by Mr. Jeff Scott (DSA).

The FSI-interface team is appreciative for the opportunity, assistance, and collaboration from AFRL/RQ and Virginia Tech in this effort and wishes to acknowledge that the developments in this effort directly resulted from this partnership. The authors gratefully acknowledge the technical support of Mr. Dean Bryson, Mr. Gary Clayman (DSA), Dr. Aaron McClung, and Dr. Bret Stanford (AFRL/ RQVC), and Dr. Manav Bhatia, Dr. Mayur Patil, and Dr. Craig Woolsey (Virginia Tech).

2.0 INTRODUCTION

OBJECTIVE I

The task objectives were to: understand the flight dynamics of flapping-wing micro air vehicles (FWMAVs), and design and test flight control laws on flapping FWMAV prototypes that enable controlled flight with the ability to hover and maneuver forward/backward and sideways. The effort was aimed at developing a knowledge base that will lead to Intelligence, Surveillance, and Reconnaissance (ISR) platforms that can hide in plain sight, reject disturbances and operate in cluttered/confined spaces. The related key science issues were to develop a control-oriented model of the aerodynamics and structure; design and implementation of a control theory, where the control forces/moments are constrained to a periodic form, and to use these periodic forces to precisely direct ISR sensors in cluttered/confined spaces in the presence of disturbances such as gusts and steady winds. The critical tasks to be accomplished included the following:

- Design flight control systems in cooperation with in-house researchers that can be implemented on real-time ground based and on-board embedded flight computers.
- Design 3- and 4- Degrees of Freedom (DOF) controllers so a FWMAV model can perform waypoint tracking in a 6-DOF simulation, utilizing power and control actuators, similar those on vehicles recently fabricated at the Harvard Microrobotics Lab.
- The use of Matlab/Simulink to devise, develop, and analyze FWMAV modeling prototypes
- Analyze the control authority obtained by manipulating the angle-of-attack and stroke-plane-angle of the wings through control oriented modeling.
- Develop a modular simulation framework that allows seamless transitions between aerodynamic models of varying fidelity. Compare results of simulations based on blade-element aerodynamic models to those generated by high fidelity computational fluid dynamics.
- Produce control-configured FWMAV designs; produce mathematical models of new FWMAV designs; document design methodologies.
- Utilize angle-of-attack and stroke-plane-angle for control to provide more freedom when choosing control inputs to provide 4 to 6-DOF control.
- Analyze the control authority obtained by dividing parameters of the wingbeat further down to quarter stroke parameters
- Develop control-oriented mathematical models and simulations of candidate MAV designs; develop flight control designs in cooperation with the in-house research team.
- Develop methods that enable analysis of FWMAV control system performance and stability.
- Devise control analysis tools which are applicable to FWMAV control designs similar to the 4- to 5-DOF controllers which have been shown to perform adequately in simulation; analyze robustness and performance.
- Develop and implement low-level motor control laws that enable wing beat motion profiles to be produced that generate desired cycle-averaged control forces and moments.
- Design and implement drive motor control algorithms on flight weight circuit boards that allow wireless control of the motion of onboard flapping mechanisms.
- Design and construct flight-weight circuit boards that make use of surface mount components to implement low level motion control of onboard flapping mechanisms.
- Produce new control-configured FWMAV designs; produce mathematical models of new FWMAV designs; document design methodologies.

- Validate mathematical models by comparing analytical and numerical results to those obtained from laboratory experiments.
- Implement fuselage position control laws on a surrogate flapping wing aircraft that is mounted on a puck to support motion control tests on an air table.
- Develop instrumentation and data analysis software that supports the analysis of the results of bench test, air table, and flight experiments.

The task research was initially focused on the theoretical modeling and control design. Emphasis was placed on devising Simulink models capable of sufficiently replicating the gross behavior of a FWMAV in order to conduct tests in simulation of control laws based on wingbeat manipulation. The models were used to produce control oriented models facilitating the analytic extraction of control derivative or sensitivity matrices for a variety of control input sensor suits. The results of this work and its derivatives are available in detail in our published works (see Appendix).

The last year of research focused on small bird sized vehicle fabrication to provide proof-of-concept experiments, in particular, of the split-cycle constant period wingbeat manipulation/modulation for control purposes. This included the design of a fuselage, wingbeat actuation assembly, electronics, software, wing design and production, and fabrication. The resulting FWMAVs were tested on both a force balance and an air-table to demonstrate control over the two primary degrees-of-freedom predicted to be provided by split-cycle constant period manipulation in the previously devised theoretical and simulation models.

OBJECTIVE II

The Multidisciplinary Multi-fidelity Model-based Computational Tool (M^{3}CT) was developed to enable the computational design of micro air vehicles through a generalizable, modeling and optimization framework. The M^{3}CT is integrated into the SORCER iGrid8 framework as a principal Requestor. Leveraging the synergistic principles of the Federated Service Object Oriented Architecture (FSOOA), the M^{3}CT is essentially providing a universal User eXperience (UX) in developing Exertion Oriented (EO) work packages without the end user requiring an explicit understanding, or requiring programmatic maintenance of, each highly technical service provider.

The core activity within FSI-Interface was the development of the M^{3}CT, including technical support associated with making engineering components (e.g., an optimization algorithm) available within SORCER and reflecting end user practices in the M^{3}CT workflows. From an engineering standpoint, the function of the M^{3}CT is to enable the end user to:

- Define a flapping wing MAV configuration
- Analyze this configuration (simulate its physical behavior)
- Define design variables (DVs), performance objectives, and engineering constraints for the configuration
- Optimize the configuration for the defined DVs, objectives, and constraints

Various engineering disciplines and modeling fidelities are represented in the M^{3}CT palette. 3D wing aerodynamics is modeled using blade-element, or quasi-steady, aerodynamics. Wing structure, when allowed to be flexible, is modeled using nonlinear beam-theory. When assumed rigid, the wing inertial properties are still modeled. For pinned analysis (the MAV is assumed to be fixed in space), the aerodynamics and structural dynamics are coupled for wing design purposes. For flight analysis, either open loop or closed loop, the MAV is not pinned and its rigid body degrees of freedom are modeled, including body mass (in this mode of use, the MAV wings are assumed rigid). A trim module is provided to set certain variables for trimmed hover or forward flight. An additional set of controller modules is provided to build and evaluate the performance of LQR-based closed-loop control. Optimization can be performed using DOT, CONMIN, or MMA.

The aerodynamic forces and flow fields produced by the motion of 2D airfoils are modeled at a higher level of fidelity using the Maelstrom provider, which is based on an unsteady, and nonlinear point-vortex methodology. This capability was incorporated into the M^{3}CT early in the FSI-Interface project as means to study end user workflows while the QSCSD and MUAV providers were being constructed.

Scope

This report describes at a high level the work carried out for FSI-Interface and FSI-Controls, with emphasis on (1) technology challenges associated with the development of a generic capability to design advanced aircraft and (2) lessons derived from the project that can lead to a more successful capability in the future. Section 3 addresses software architecture, including that of SORCER and the M^{3}CT. Section 4 provides an introduction to the engineering problem FSI-Interface is designed to study. Section 5 focuses on the technology challenges expected in extending FSI-Interface to treat a broader range of aircraft concepts. Finally, Section 6 provides a set of conclusions, including a detailed look at potential next steps to overcome the technology challenges introduced in Section 5.

The reader should note that details associated with the work are reported in two documents that fall outside the scope of this final report. The first document is the M^{3}CT Developers Guide. This document contains detailed information useful to those software developers intending to modify or further develop M^{3}CT (e.g., specification of the geometric data class). The second document is the M^{3}CT Users Guide. This document contains detailed information for potential users of M^{3}CT, including instructions on setting up and running certain benchmark cases. The reader is also encouraged to study the Task Order 49 final report, which has additional information about application of the FSI-Controls software.

Delivery

The following items are delivered per Task Order 35:

- M^{3}CT Final Report
- M^{3}CT Software DVD

The following items were provided as software documentation for Task Order 35:

- M^{3}CT Users' Guide
- M^{3}CT Developers' Guide

3.0 METHODS, ASSUMPTIONS AND PROCEDURES

OBJECTIVE I

Areas of research focus changed to further progress. Coarsely, the effort was divided into modeling and simulation, control design, vehicle fabrication, and experiments. The efforts were undeniably connected as the modeling and simulation informed the control design and vice versa. The control design informed the modeling by providing feedback on the effectiveness of different wingbeat modifications. The modeling and control design dictated which wingbeat manipulations to employ in fabrication. The experimental results were compared to predictions made by the models. The details of the experimental setup were partly dictated by the fabrication, while the fabrication was amenable to the experiments.

3.1 Modeling and Simulation

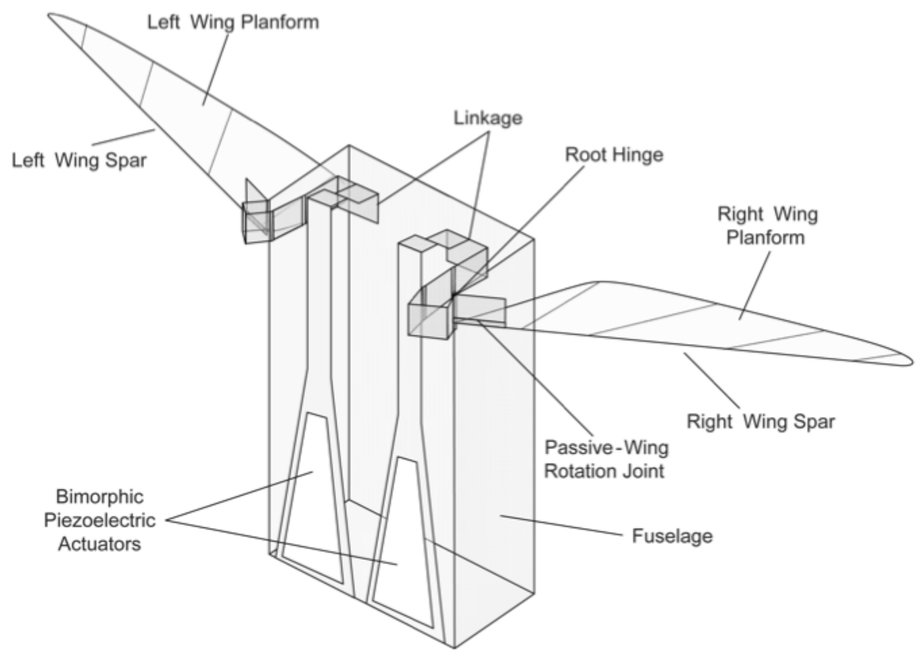


Figure 1. General Assembly of a Minimally Actuated FWMAV

The modeling and simulation effort focused on producing a quasi-steady aerodynamic model capable of producing a representation of a FWMAV similar to the one shown in Figure 1. The models employed blade-element theory with empirical data from the literature to represent lift and drag coefficients as functions of angle-of-attack or as lookup-tables; these type of models are commonly referred to as quasi-steady models. Gradual improvements and enhancements were made to the models. The final versions had online integration of the velocity vector field over the wings, accounting not only for stroke velocity but also the rigid body velocity and first principle based inertial effects. The models provide a six degree of freedom representation of a FWMAV in flight, in particular at near hover conditions. Models were implemented in Simulink and MATLAB.

The primary use of the models was threefold: to produce simplified control oriented models and design tools based on cycle averaging, to analyze the effect and effectiveness of various wingbeat manipulations, and to provide a truth model to test the eventual control laws. The cycle-averaging of forces and moments was performed analytically on blade element based models. Cycle-averaging involves taking the time averaged integral of the forces and moments produced by a wing stroke cycle assuming a sinusoidal stroke position waveform, i.e., from an initial wing position in a stroke, until the wing returns to that position. Sensitivity analyses, or partial derivative computations, were performed on the cycle-average representation of the forces and moments. Sensitivities were investigated for stroke frequency (number of complete strokes per second), bias (the shifting of the midpoint of the stroke), angle-of-attack (shifting of stops for the wing rotation about its spar), amplitude (total deviation of the stroke about its midpoint), stroke-plane (tilting the plane in which the wing stroke lies), and split-cycle (employing different velocity profiles for the up- and downstroke). Particular emphasis was placed on developing the split-cycle approach, illustrated in Figure 2. Split-Cycle control was a particularly novel control innovation of interest, facilitating control over four degrees-of-freedom using only two actuators. These two actuators dictate the stroke velocity profile of each wing. The models predicted that around hover, with the vehicle upright (top of head pointing upward, belly facing forward), split-cycle manipulation provides control over two degrees of freedom, forwards/backwards translation and roll rotation. Frequency manipulation adds vertical translation and yaw. Various options were considered and tested in simulation for pitch control including the use of a bob-weight, stroke bias, and stroke-plane manipulation.

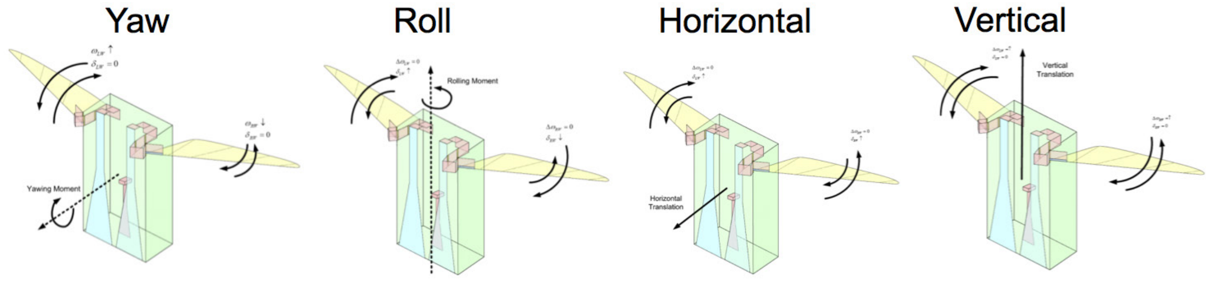


Figure 2. Split-Cycle Illustration, Longer Arrows Represent Faster Half Strokes and Shorter Arrows Represent Slower Half Strokes

3.2 Control Design

Control oriented models were derived from the quasi-steady FWMAV models. In particular, sensitivities of the cycle-averaged forces and moments to stroke manipulations were derived. This led to the investigation and simulation implementation of various control inputs as discussed in Section 3.1. The result was a variety of methods to obtain control over four or five degrees-of-freedom in a cycle-averaged sense. For example, utilizing control over stroke frequency, split-cycle modulation, and stroke bias independently for each wing, allows for control of cycle-averaged forward, backward, and vertical translation, as well as all three attitudes.

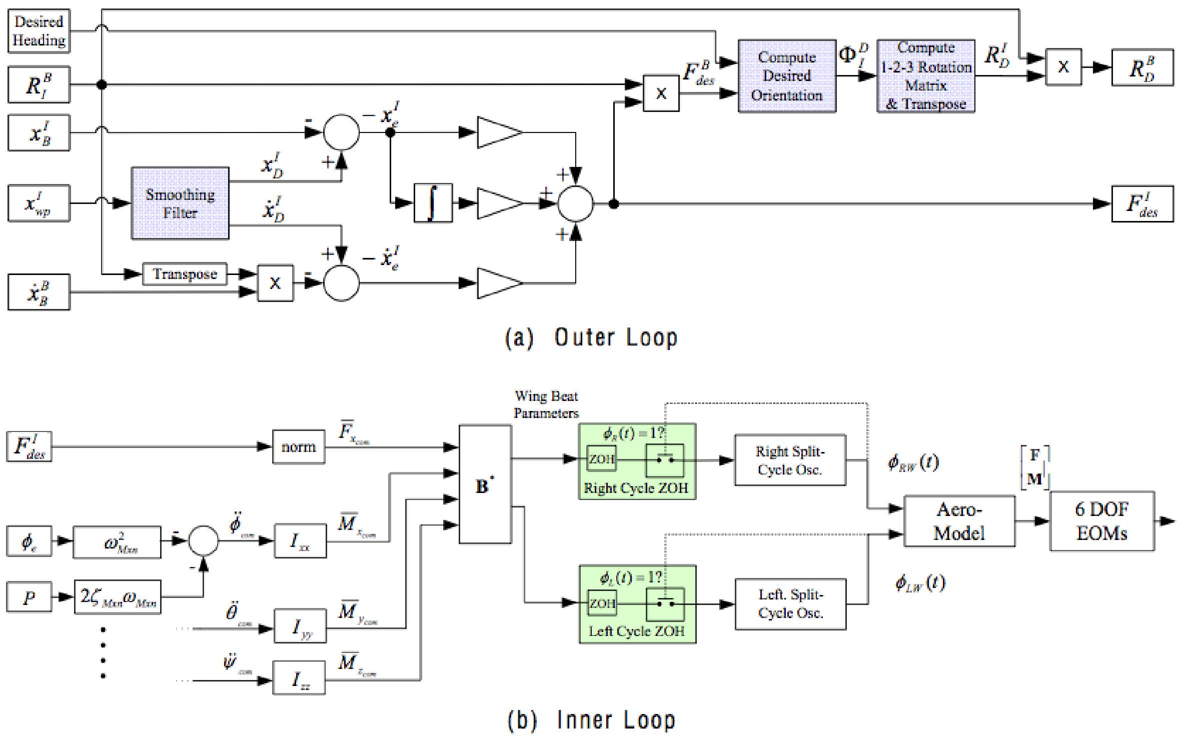


Figure 3. Block Diagram Example, 4-DOF Controller

The control scheme was devised under the assumption that cycle-zero-order-hold (Cycle-ZOH) is employed. This implies that at the start of each stroke, the stroke parameters are set and persist throughout the subsequent stroke, as illustrated in Figure 3 and detailed in our published work. Control laws were designed on a two level hierarchy. The inner-loop control is a proportional, derivative, and integral control scheme assuming independent loop closure for the controllable translations and attitudes. The outer-loop control law sets the reference signals for the inner loop controller in order to produce waypoint tracking. Two control schemes were devised and successfully provided waypoint navigation using either 4 or 5 degree-of-freedom control to manipulate the FWMAV's motion in our most sophisticated quasi-steady 6 degree-of-freedom models.

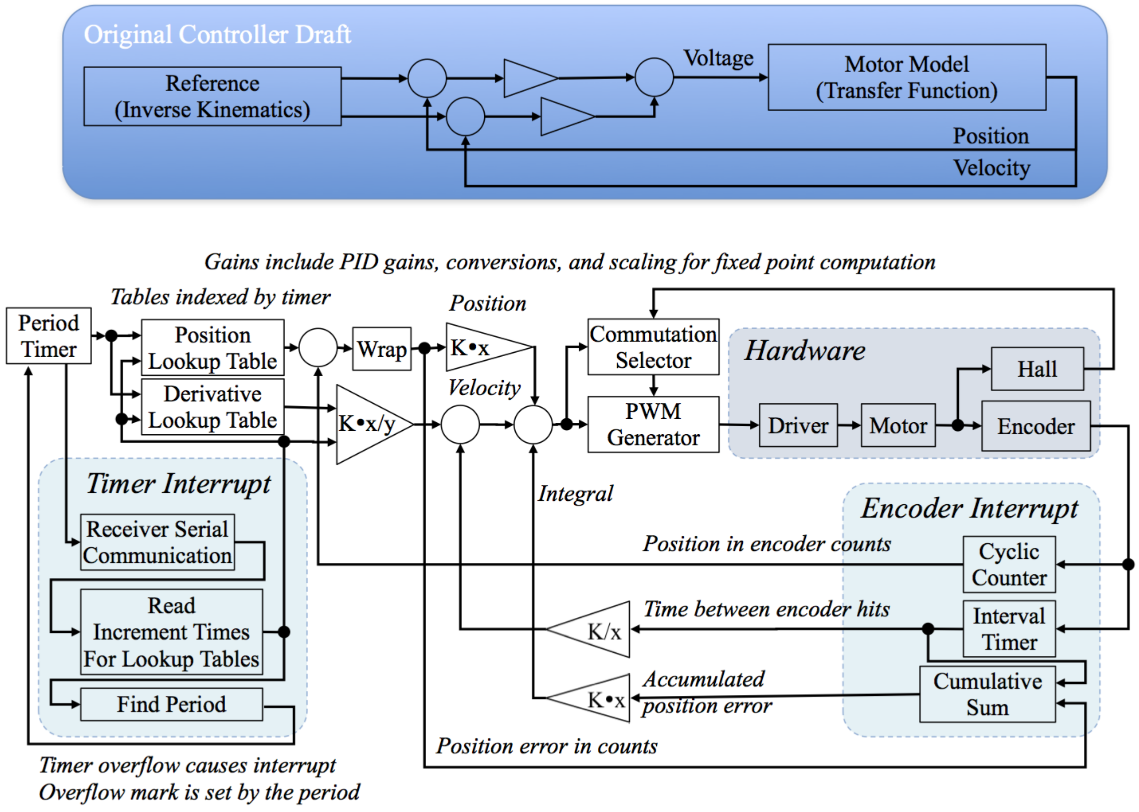

Original Controller Draft
Reference (Inverse Kinematics)
Voltage
Motor Model (Transfer Function)
Position
Velocity
Gains include PID gains, conversions, and scaling for fixed point computation
Tables indexed by timer
Period Timer
Position Lookup Table
Derivative Lookup Table
Wrap
Position
K•x
Velocity
K•x/y
Commutation Selector
PWM Generator
Hardware
Driver
Motor
Hall
Encoder
Timer Interrupt
Receiver Serial Communication
Read Increment Times For Lookup Tables
Find Period
Integral
Encoder Interrupt
Position in encoder counts
Cyclic Counter
Time between encoder hits
K/x
Interval Timer
Accumulated position error
K•x
Cumulative Sum
Position error in counts
Timer overflow causes interrupt
Overflow mark is set by the period

Figure 4. Block Diagram of Stroke Position/Velocity Control

A control design was also needed in practice to control the stroke position and velocity profile produced by the prototype FWMAVs. A brushless DC motor was used to drive a four-bar crank-rocker mechanism to transform rotational motion into a rocking motion, i.e., the wing stroke. The controller needed to drive the motor according to the inverse-kinematics of the 4-bar to produce a cosinusoidal stroke waveform. Moreover, the controller needed to facilitate frequency and split-cycle manipulation using an implementation sufficiently simple to run on 16-bit microcontrollers. The resulting control design is illustrated in Figure 4 and was successfully implemented for force-balance and air-table experiments. The design is based on using table lookup to produce reference signals for a PID (proportional, integral, derivative) controller. The tables were generated a priori in MATLAB based on the inverse-kinematics of the 4-bar mechanism geometry and a timer is used to index the position and differential values. Two distinct values are used to dictate the time difference between index increments during the first and second half of the table. By making one value smaller than the other, split-cycle modulation is achieved. A remote controller is used to dictate the timing of the stroke of each wing and each half stroke.

3.3 Fabrication

The primary fabrication objectives were to machine a fuselage, construct wings, and fabricate electronic circuits capable of implementing split-cycle control. Various design tests were also performed using 3D printing.

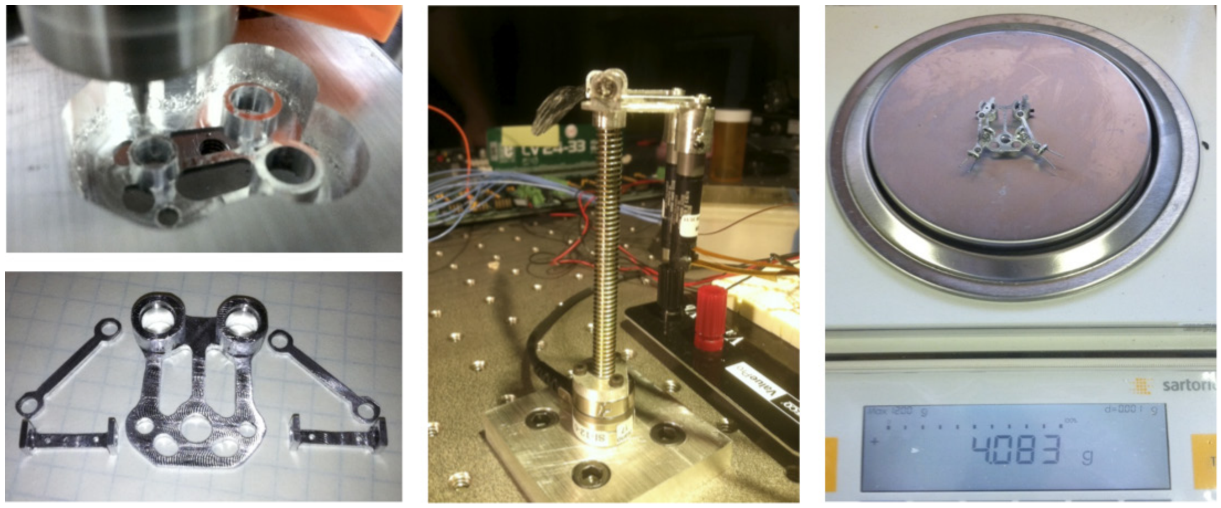


Figure 5. Fuselage Fabrication

The fuselage consists of a body structure which incorporates motor attachments and the 4-bar crank-rocker mechanism which drives the wing spar, as seen in Figure 5. Furthermore, there are protruding pins near the wing spar mounts which constrain wing rotation about the spar, thus enforcing desired bounds on the wings' angle-of-attack. All the parts except for off-the-shelf bearings and bolts, were milled from aluminum. The parts were drawn in computer-aided-design (CAD), specifically SolidWorks, and G-code generated by SoldidCAM was used for the machining. The weight of the fuselage, including the 4-bar mechanism and bearings, totals approximately 4 grams.

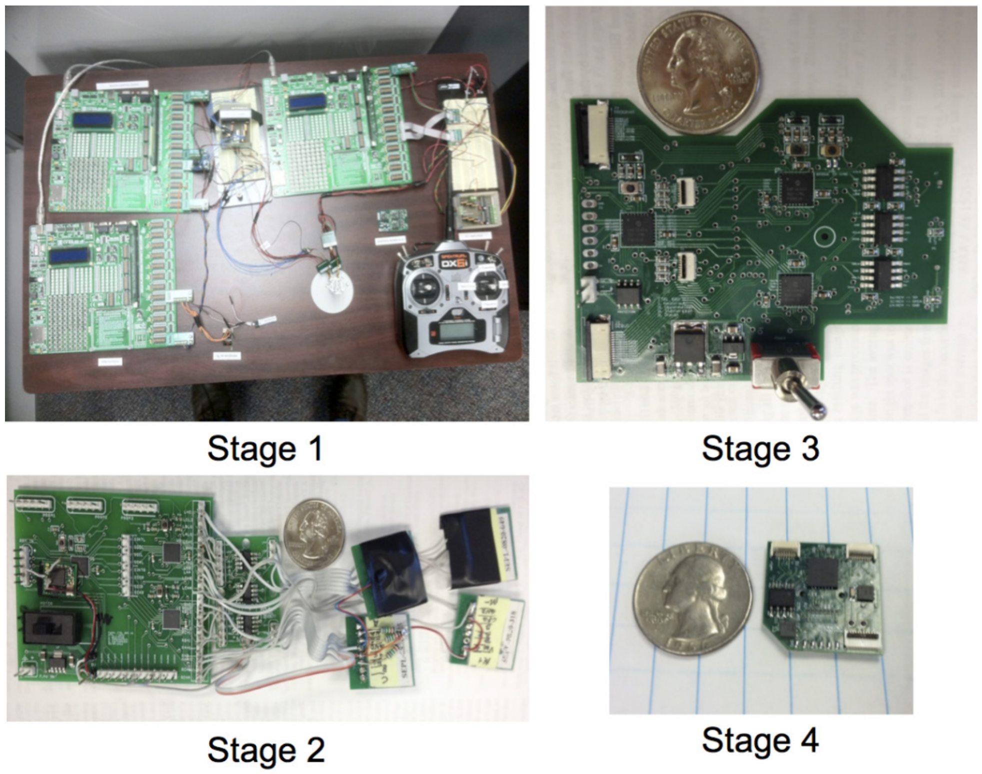


Figure 6. Evolution of The Circuit Board Design

The electronics circuits evolved as experience was gained. The purpose of the electronic circuits was to facilitate using three PIC24FJ64GA102 16bit microcontrollers to control the stroke actuation using brushless DC motors. Furthermore, the microcontrollers were required to communicate with a remote controller and with each other. Two microcontrollers were programmed with the motor control laws and used pulse-width-modulation (PWM) and Hall effect sensors to drive the motors via a set of half-H-bridges. The third microcontroller communicated with a remote control receiver using pulse-position-modulation (PPM), deciphers the signal, and communicated the remote control commands to the two microcontrollers which control the motors. The circuit board development evolved into four stages as shown in Figure 6. The first stage used developmental boards with similar microcontrollers. The second stage combined the microcontrollers on one board while using a separate driver board to connect to the motors. The third stage combined the microcontroller and driver boards. The fourth and final stage, minimized the board size to produce a flight weight board, weighing 1.71 grams.

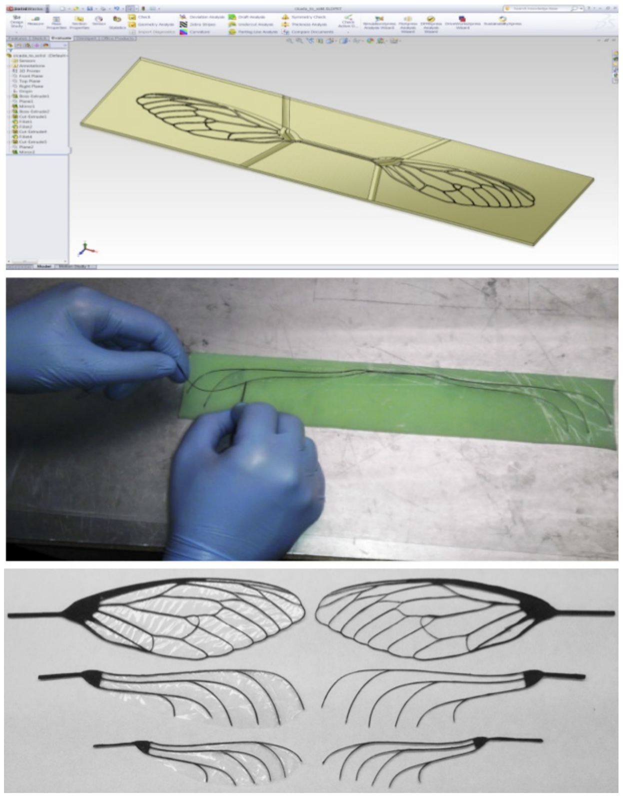


Figure 7. Wing Fabrication, CAD to Negative Mold to Carbon Fiber and Capran Wing

Wings were fabricated using carbon fiber for structural support and a capran nylon film for wing membrane. Three dimensional CAD designs of wings were 3D printed and a negative silicon mold made from the prints. Fibre Glast 2000 resin was applied to 1k carbon fiber tow then layered in the silicon mold, vacuum bagged, and baked. It was important to vacuum bag and bake the carbon fiber wing spars to ensure that they conformed to the molds. Figure 7 shows, top-to-bottom, a screen capture from the CAD software rendering, a photograph of a technician constructing a wing, and a set of finished wings. The carbon fiber wings proved to be lightweight and strong.

3.4 Experiments

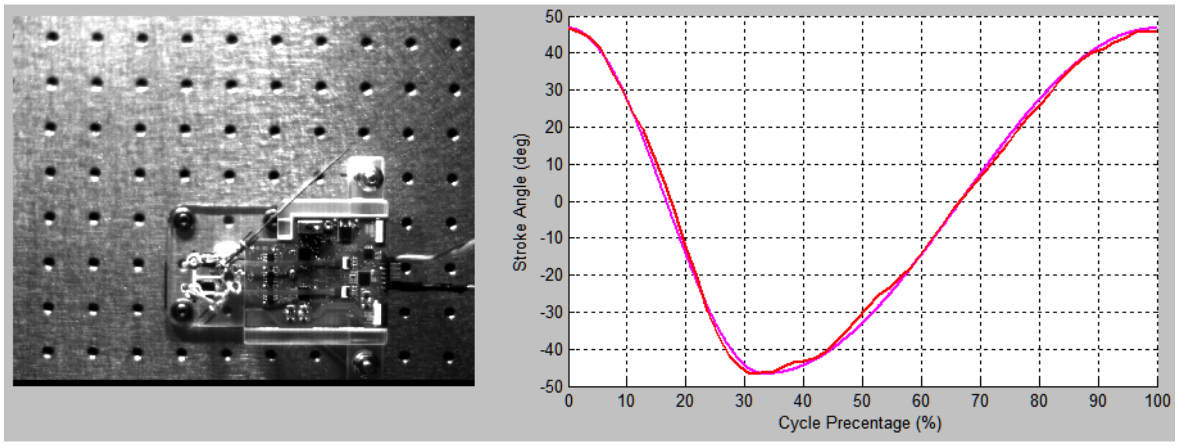


Figure 8. Split-Cycle High Speed Video Recording Demonstration

A variety of experiments were performed in the course of this research effort. They included the use of a prototype FWMAV mounted on a force-balance, high-speed video recording, and mounting a prototype FWMAV on a puck to float on an air-table. A Nano17 electronic force balance was used to measure forces and moments produced by prototype FWMAVs. The experiments have thus far been inconclusive because of the signal-to-noise ratios and related implementation issues. Controlled split-cycle modulation of the stroke was, however, confirmed using high-speed video recording and manual tracking, as seen in Figure 8. The magenta line shows the split-cycle waveform to be tracked and the red shows the measured stroke angle at the wing spar origin.

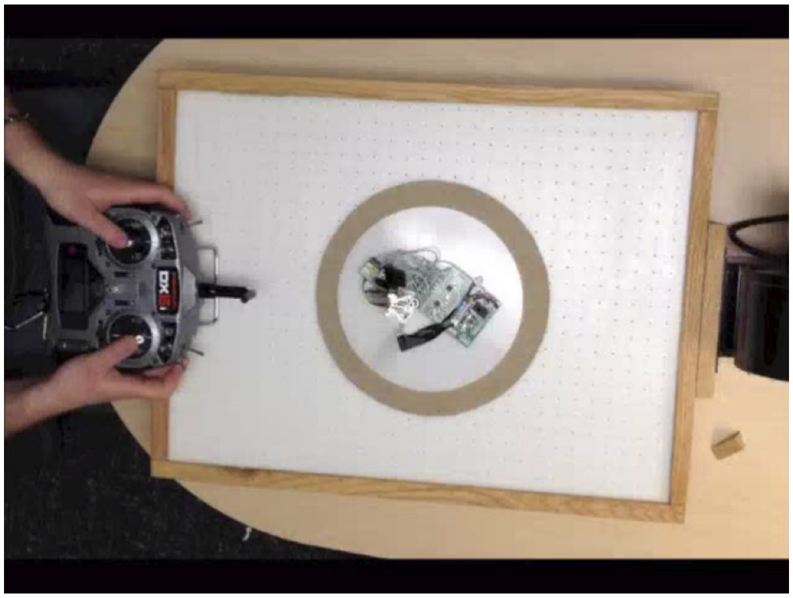


Figure 9. The Airtable Experiment Setup with a Prototype FWMAV Mounted on a Puck

Both an airtable and airtable mount (puck) for FWMAV prototypes were constructed, as shown in Figure 9. The airtable allows for testing FWMAV prototype capabilities to translate and rotate on a horizontal plane. This allows constrained maneuver testing, eliminating vertical motion (compensating for weight), and pitch/yaw rotation. In particular, recall that the primary effect of split-cycle modification of the stroke was to produce forward/backward translation and roll. Thus, the airtable and puck provide an ideal experiment for the viability of split-cycle modulation based control.

OBJECTIVE II

Three parallel efforts comprise the architecture of the M^{3}CT. The principal architecture is summarized here for a quick reference. The implementation architecture is unique to the M^{3}CT effort. Ongoing development of the Principal Architecture has had some effect on the variable implementation of M^{3}CT. The collection of providers leveraged by the M^{3}CT product are highly variable in design, language, and intent, which is another challenge of providing an agnostic approach to presenting these providers to the user.

Development of the program architecture is itself research. The unique distinctions of coding practices and implementation techniques from class to class are moderately variable by this very nature. It is important to consider each class as an experimental implementation. Therefore, differences in similar classes are derived by their research iteration.

3.5 Principal Architecture

The Principal Architecture of the M^{3}CT product is a Federated Service Object Oriented Architecture (FSOOA). Within the FSOOA architecture there are providers, requestors, and registrars. Similar to a Service Oriented Architecture (SOA), a provider presents work capabilities to the network, a registrar provides network connectivity information, and a requestor can be interpreted as a user of the network. However instead of a provider being substantially limited to a specific form or function, or limiting connectivity concurrency between requestors and providers, the SORCER API generates work packages that are exerted across the federation.

SORCER is a federated Service-to-Service (S2S) meta-computing environment that treats service providers as a network of peers within the well-defined semantics of the FSOOA. It is based on Jini semantics of service in the network and the Jini programming model with explicit leases, distributed events, transactions, and discovery and join protocols. Jini focuses on service management in a networked environment. SORCER focuses on Exertion Oriented (EO) programming, and the execution environment for exertions.

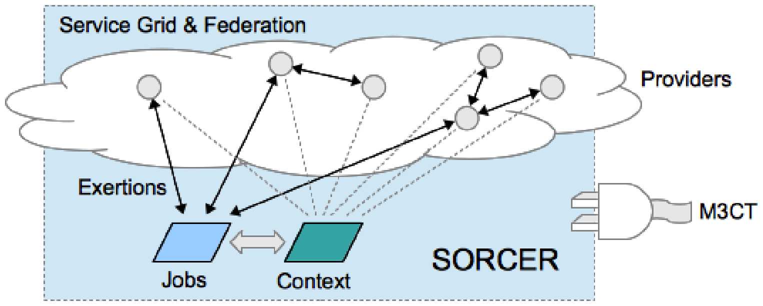


Figure 10. Architecture Environment

SORCER utilizes an integrated-grid platform (iGrid8) that maximizes the strengths of both compute grid and meta-compute grid platform strategies. This platform provides the added benefit of downward compatibility with legacy applications.

A great benefit of utilizing a framework such as SORCER, is that many various languages can be successfully integrated into the M^{3}CT environment. A disadvantage to the same, but not exclusive to SORCER or any other resource serving in its capacity, is that exposure to multiple languages can derive numerous variations in implementation standards. SORCER alleviates many of these lower level implementation issues through the use of context objects. These context objects can be standardized more easily, as they exist outside of platform specific requirements.

SORCER provides over a hundred variations of implementation. In the next steps of M^{3}CT, only a handful of these variations will be needed. Over the next decade of research and development, access to the remaining variations, and continued improvements to the SORCER framework, will be beneficial in providing access to agnostically accessible service platforms in am ever growing federation of computer science technologies.

3.6 Implementation Architecture

The Implementation Architecture of the M^{3}CT product is a Model View Controller (MVC) approach integrating into SORCER through its API library as a principal Requestor. The M^{3}CT requestor is written in Java, leveraging the cross platform deployment capabilities needed for a broadly executable user interface. M^{3}CT consists of numerous dialogs and forms hosted within a Multiple Document Interface (MDI) Graphical User Interface (GUI). This provides the user of the M^{3}CT application the ability to interact with these providers at a simplified and comprehensive level allowing for users to leverage available providers in the creation of exertion based work packages without explicit understanding of each provider.

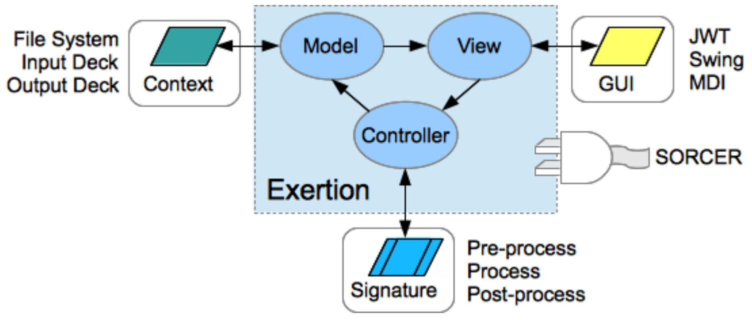


Figure 11. M^{3}CT Implementation

Numerous Providers are integrated into SORCER, primarily utilizing JNA wrapping techniques. These providers are discoverable and configurable through the M^{3}CT user interface. This design pattern circumvents the computer science background typically required for working in a distributed computing environment. Providers are written in their native languages, such as Fortran, C++, MATLAB, and other languages that are ideal for running highly demanding algorithms, or were simply written in a language that the designer of that algorithm is comfortable with. These algorithms can range from analytic resources such as aeroelasticity, fluid dynamics, and vorticity, to providers that perform iterative optimizations or improve performance through interpretive model reduction patterns.

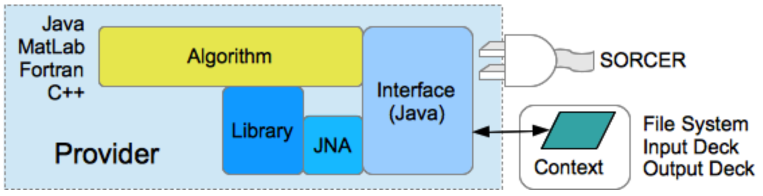


Figure 12. Provider Implementation

The M^{3}CT Requestor and the numerous experimental Provider companion implementations encourage and unify the collaborative efforts of multiple disciplines by converging methodologies and converting algorithmic discoveries and processes into a generically accessible form. This benefit is further enhanced through programmatic optimization resources that can help lead a research effort in the right direction with both performance enhancement and data integrity.

This implementation architecture research has both discovered and uncovered implementation details and associative development patterns that serve as a valuable lessons learned resource in the design and development of a next-generation architecture.

In order to expose each application to the Java programming language, the independently generated sources are wrapped with JNA or Groovy. The Java wrapping process allows for an application to be created according to the software language that best suites its execution.

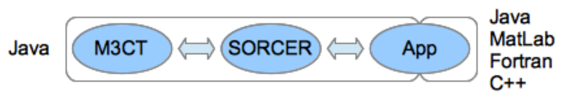


Figure 13. Leveraging Languages

Once the underlying product is available to the Java source as a library, the application leverages the SORCER API for interoperability with the M^{3}CT implementation of the FSOOA. Once running, an Application is discoverable and can be instantiated by other applications, such as the M^{3}CT requestor. This environment also allows the application to utilize the hardware solution that best processes the underlying algorithm.

4.0 RESULTS AND DISCUSSION

OBJECTIVE I

The results of the research tasks are divided into three sections: simulation, fabrication, and experimental results. The research efforts primarily focused initially on modeling and control design and producing results in simulation. In order to validate some of the abilities predicted by simulation, prototype FWMAVs needed to be constructed, thus producing fabrication results. Finally, actual experiments are presented, in particular, the airtable results. One of the primary metrics of success for the research efforts conducted for this research task is the significant number of peer-reviewed publications. In collaboration with in-house researchers, publications numbers included 2 journal papers, 1 book chapter, 1 technical report, and over 13 conference papers.

4.1 Simulation Results

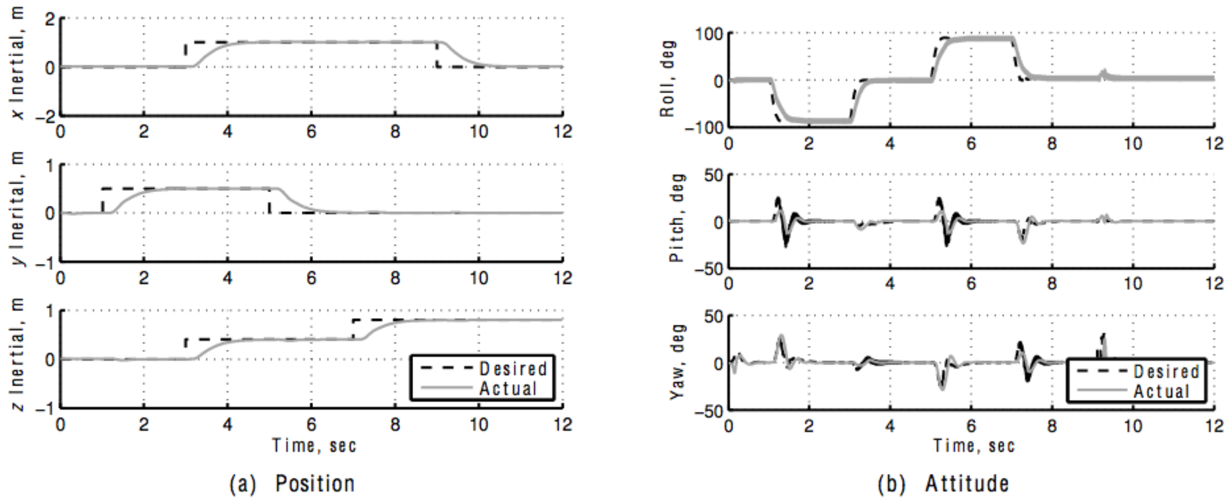


Figure 14. Simulation Results Using a 5-DOF Controller (Frequency, Split-Cycle, Bias) on the 6-DOF Quasi-Steady State FWMAV Model Including Rigid Body Velocity

Four and five degree-of-freedom control designs were devised and tested on 6 degree-of-freedom FWMAV models of insect size, 60-90 mg in weight with a wingspan of up to 6 cm. The models were devised to be consistent with state-of-the-art quasi-steady FWMAV models because higher fidelity computational fluid dynamics models are, as of yet, too computationally intensive for closed loop FWMAV simulation. Figure 14 shows a 5-DOF controller performing waypoint tracking. Note that the position and attitude, controlled by the inner-loop controller, tracks the reference generated by the outer-loop controller to perform guidance between waypoints. See our published work for further details.

4.2 Fabrication Results

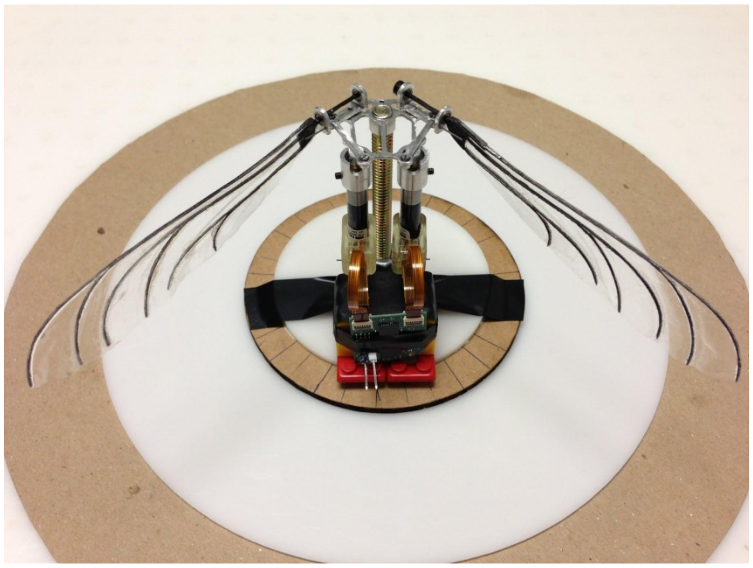


Figure 15. Complete Prototype FWMAV Mounted on an Airtable Puck

As shown in the previous section, the design of a FWMAV prototype was an incremental process. The fuselage holding the structure to support the motors, 4-bar crank-rocker mechanism, and wing mounts was completely constructed with mounting for either an airtable puck or force balance. The electronic circuit boards were designed and manufactured to drive and control the wing actuators and be remote controlled with a Spektrum Dx6i transmitter. Figure 15 shows a fully assembled FWMAV prototype mounted on a puck.

4.3 Experimental Results

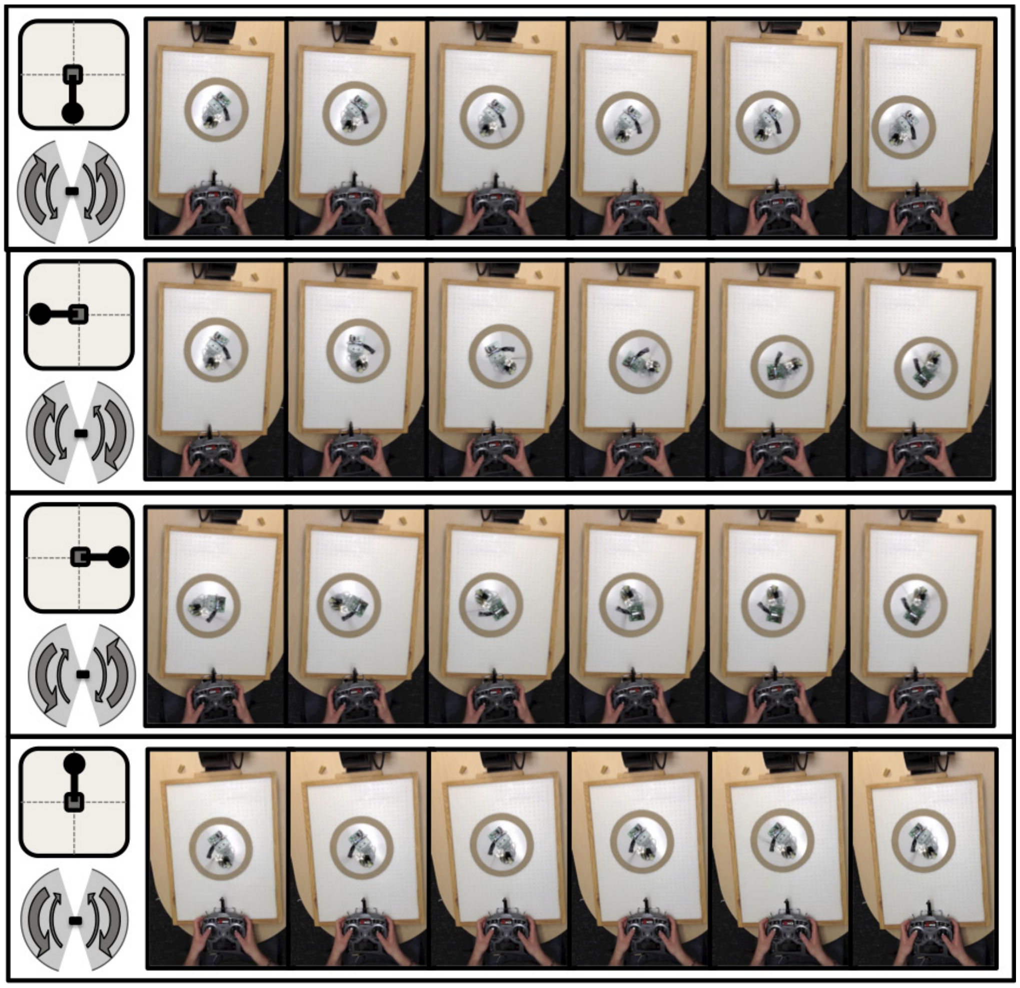


Figure 16. Airtable Experiment, Demonstrating a Remote Controlled Prototype FWMAV Employing Split-Cycle Stroke Modulation

Experiments on the prototype FWMAV included high-speed video recordings, force balance, and airtable controlled experiments. High-speed video confirmed the ability of our hardware and software to produce a controlled split-cycle modulated wing motion. Force balance measurements proved inconclusive due to signal to noise ratio issues. Finally, the FWMAV prototype was mounted on a puck and remote controlled on an airtable. Figure 16 reads as a filmstrip from left to right demonstrating the milestone achievement of being the first demonstration of the control effectiveness of split-cycle modulation. The remote controller is used to adjust the split-cycle stroke, as indicated on the figure, either symmetrically between the two wings or asymmetrically. The thick arrows indicating faster motion than thin arrows. Specifically, split-cycle modulation was shown to allow control of two degrees of freedom as predicted by the mathematical models and simulations produced earlier. Simulations showed that, in principle, the same type of mechanism can, with frequency modulation, control both vertical motion and yaw. Additional design measures need to be taken to provide pitch control, such as the addition of controlled stroke bias or stroke-plane modulation.

OBJECTIVE II

Section 4 contains a summary of results obtained with FSI-Methods, FSI-Controls, and FSI-Interface. First, the benchmark problems are described to provide the reader an understanding of analysis objectives. Second, the basic analytical process is described to provide the reader an understanding of the methodology elements. Lastly the results of the individual benchmark problems are presented. Further details are included in the MSCT Users Guide.

Two basic sets of results are shown. One set of results are shown for a MAV that is pinned (not free to move), and which has flexible wings. Of interest is the role wing flexibility plays in the production of aerodynamic forces. A second set of results are shown for a MAV that is free to move, but which has rigid wings. Of interest is the tradeoff between design of control system and wing and flapping characteristics for efficient operation in gust.

Prior to summarizing the results of canonical study cases, the methodology elements are described, starting first with brief definitions of basic concepts, followed by descriptions of the analytical capabilities.

4.4 Problem Description

In this section, the basic problem that is being solved by M^{3}CT and its related software capabilities is described. Further details can be found in technical papers by Stanford et al.[i] and Bhatia et al.[ii]

4.4.1 MAV Configuration (Geometry, Kinematics, Structure)

A flapping MAV configuration is considered, and the basic description involves five components: configuration geometry, kinematics, structure, vehicle motion, and vehicle control. This section reviews vehicle geometry, kinematics, and structure.

The physical geometry consists of a body and two wings, although only the wings are modeled aerodynamically. The geometric configuration is symmetric, left-to-right. Wings are offset from the body, and any physical connection between the wings and the body is not modeled. Potential physical contact between wings and body owing to motion and wing deformation is not explicitly modeled or avoided. Wing planform geometry is defined discretely and symmetrically about a geometric axis. Wings are flat and prescribed to have thickness, although thickness is primarily a structural consideration.

Each wing is prescribed to move (i.e., flap) by specifying instantaneous rotation (Euler) angles: three for each wing. Rotations are specified about wing root locations that are offset from the body center of mass. In particular, the wing roots are placed an adjustable distance from the vehicle symmetry plane. Feather rotation is specified about the wing's geometric axis.

The wing structure is considered to be that of a beam whose cross-sectional geometric properties vary with span-wise location. The wing structure extends from the wing root to the wing tip, and the cross-sectional geometry is defined to match the specified planform and thickness distributions (as functions of span-wise location). At the model level, the aerodynamic and structural character of the wings is defined at an adjustable number of chord-wise strips. For example, if the wing is decomposed into 10 strips, each strip has a uniform chord and a uniform thickness. For a flexible wing, beam axis is assumed to be at the geometric center of the wing, coincident with the wing's geometric axis.

4.4.2 Pinned MAV

Studies are divided along two lines: the MAV is either pinned or free-to-move. MAVs that are pinned are prescribed not to move (i.e., the body is specified to not translate or rotate with respect to a lab frame). MAVs of this configuration type are allowed to have flexible wings (i.e., wings that deform under loads), whose structure is described above.

4.4.3 Free-to-move MAV

Free-to-move MAVs are modeled as multi-body objects, whose motion is dictated by a combination of gravitational, inertial, and aerodynamic forces (the latter two produced by prescribed flapping), as applied to a 6-DOF set of rigid-body equations. As a multi-body object, wings are assumed to move in a manner reflecting attachment to the vehicle body (although physical connectivity is not defined). It should be noted that free-to-move MAVs are assumed to have rigid wings. These wings are modeled inertially (and can significantly influence body motion at large time scales), but not elastically, as true for pinned MAVs.

4.4.4 Free-to-move MAV with Control

A controller based on Linear-Quadratic-Regulator (LQR) theory is used to maintain stability of a free-to-move MAV under a specified mission (either hover or a prescribed motion)[ii]. It is assumed that the controller is able to observe all states of the vehicle (position, rotations, velocity and angular rates). Any deviations in these quantities are used by the controller to define perturbations to the trim kinematics in-order-to maintain stability. The user can specify the kinematic parameters that can be perturbed by the controller, and the relative importance of minimizing state-perturbations versus minimizing control-cost.

4.5 Fundamental Steps of Analysis and Optimization

This section reviews some key methodological aspects of the MAV studies, including both vehicle analysis and vehicle optimization. As described above, vehicle geometry, kinematics and structure are parameterized (i.e., can serve as design variables). For the free-to-move MAV, design variables can also be assigned to the control scheme. Vehicle behavior can be analyzed, meaning the time-dependent, multi-disciplinary responses of the vehicle can be numerically simulated for specified initial conditions and parameter values defining the configuration. Gradient-based optimization can then be used to optimize the configuration (change the design variables) to improve a design objective while meeting constraints.

4.5.1 MAV Analysis

MAV analysis consists of aerodynamic analysis, structural dynamic analysis, aeroelastic coupling, trim computation, simulation of rigid-body dynamics, and gust modeling.

Aerodynamic analysis is based on blade-element theory, following the implementation of Berman and Wang[iii]. Through this approach, aerodynamic loads are computed for each chordwise strip based on the instantaneous orientation and motion of the strip. Effects of interference between strips and between strips and the body are not accounted for.

The dynamic response of wing structure is computed for the pinned MAV (the wings of the free-to-move MAV are assumed rigid). A co-rotational finite-element formulation, implemented by AFRL[iv] is used to analyze the time-dependent deformation of the beam representing the wing structure. This formulation is nonlinear and allows for large, time-dependent deformations about a rigid and virtual reference line that extends from the wing root towards the wing tip. Three points are worthy of comment:

1. The dynamic response of the wing to periodic, rotary actuation is computed using a second-order-accurate, time-marching scheme. Consideration was originally given to a time-periodic approach using finite differences or spectral elements, but the potential benefits seemed to be outweighed by the additional complexity and fragility of the time-periodic approach.

2. If the wings of the pinned MAV are to be modeled as rigid, then the stiffness of the wing should be specified to be orders-of-magnitude larger than the default value. While the wing can be made nearly rigid, wing deformation and aeroelastic interaction are still computed at a cost on par with that of a more flexible wing. The current version of the code does not allow for the disabling of the structural deformation calculation (primarily for the purpose of decreasing computational cost).

3. If the actuation of the wing is too vigorous (e.g., too large a frequency), the nonlinearity of the wing response will become severe enough to prevent convergence of the numerical scheme. This is potentially a problem for optimization, when certain regions of the design space cannot be assigned converged solutions.

Aeroelastic coupling is achieved by sub-iteration at each time level until residuals are driven below a specified threshold.

The provider that executes the pinned MAV analysis is named QSCSD for Quasi-Steady aerodynamics and Computational Structural Dynamics.

For the free-to-move MAV, a symmetric, trimmed flight state in still air is computed to provide a baseline about which gust perturbations and control variations are imposed. When trimmed, the vehicle achieves a cycle-averaged (or stroke-averaged) balance between lift and weight, and does not experience a cycle-averaged pitch moment (cycle averaged refers to time integration over a complete flapping cycle). The trim calculation involves an iterative, shooting approach that meets the two trim constraints while achieving a time-periodic solution at the specified frequency (sensitivities employed in the iterative procedure are computed analytically). In meeting the two constraints, two design parameters are computed, which can be selected through their trim IDs. For the free-to-move MAV, the equations of motion solved in the shooting procedure are the equations governing the rigid motion of the multi-body system.

Details concerning gust modeling can be found in the manuscript authored by Clayman et al.[v]

4.5.2 MAV Optimization

For most of this research, two gradient based optimization methods were applied. The first method, CONMIN, was applied to the optimization of the pinned MAV. The CONMIN optimizer utilizes the Method of Feasible Directions to compute the minimization of constrained linear or nonlinear functions. Originally written in FORTRAN, CONMIN was developed in 1973 by Garret N. Vanderplaats at the Ames Research Center and U.S. Army Air Mobility Research and Development Laboratory. The objective function and constraint behavior are provided to CONMIN by the optimization problem in M^3CT by way of the SORCER context object. The analytical constraint gradients are also provided through the context object, but may (as an alternative to the method used) be calculated by CONMIN using the built in finite difference method. (Bryson et al. also explored the use of the commercial optimization method DOT, and this capability is integrated with M3CT, but is only available on Linux systems, and license restricted. For this reason, results are shown below that were obtained with CONMIN, and DOT is not further discussed.)

For the second portion of this research involving the analysis of the free-to-move MAV, the method of moving asymptotes (MMA) gradient based optimization was utilized. The MMA method, originally presented by Svanberg[vi], is coupled with the flapping wing and wind gust models. The optimization approach includes single gust and multiple gust profiles for mean efficiency optimization. In each optimization case, minimization of the cycle average power over the entire gust simulation is used as the objective function. The peak control power expended and the total flight orbit spherical displacement of the MAV are used as metrics for constraint behavior.

Given the generic design parameters $x=[x_1,\ldots,x_{j=n}]^T$, implicit constraint behavior $f_i(x)$, objective function $f_0(x)$, and their gradients $\Delta f_i(x)$ and $\Delta f_0(x)$, the MMA obtains an optimal solution for the given design iteration from a convex, separable sub-function in which the implicit constraint behavior is replaced with an explicit approximation. The MMA is unique in that the explicit constraint behavior is obtained as a linearization of the implicit constraints based on upper (U_j) and lower (L_j) asymptotes for which $L_j < x_j < U_j$. The magnitudes of the asymptotes are modified according to the behavior of the problem gradients. As an example, if the optimization process is oscillatory, the asymptotes may be squeezed closer to design parameters; conversely, the asymptotes may be relaxed if the optimization is too steady. A more detailed discussion related to the theory and applications related to the method of moving asymptotes can be found in Ref. vi.

4.5.3 Case Management (input deck, output deck, leveraging resources)

There are vast amounts of data generated that represent each step throughout the course of a study. These values are assigned to numerous variables, and these variables must be mapped between independently executing events in a meaningful way. This need for case management is where the FSOOA is leveraged by M^3CT in the pursuit of an extensible solution.

Even though these smaller test cases only take a few hours to resolve, it is reasonable to expect that larger test cases will require numerous computational systems that exceed the capabilities of a single computer. The challenges are generally:

1) How do we map variables between two independently authored and maintained interfaces?

2) How do we preserve data from each step for progress monitoring and data analysis?

3) How do we pass significantly large quantities of data without inflicting choke points to the network of services?

4) How do we monitor, interrupt, or recover active studies through a centralized control point?

5) How to we maintain repeatability of established studies over time and versioning?

Throughout the exploratory process of addressing these challenges, both studies are resolved using different methods of invocation and conclusion. It was desired to create a uniform way for both studies to be able to leverage CONMIN optimization. The incompatibilities between the CONMIN implementation and the MMA implementation was unable to be resolved without significant changes to both. Such proposed changes are introduced later in this document.

The first study leverages the QSCSD and CONMIN providers, where M^{3}CT is the principle requestor. This implementation leverages a method of tracking objects in JavaSpace, capturing those objects for interpretation, and passing state flags between optimization runs to determine the end condition. M^{3}CT plays a passive role in this method where it is only able to update its state if the providers send a properly formed message to satisfy the update. This method is not ideal, in that errors, progress, and any other quantification of important data can be completely bypassed without the end user of M^{3}CT being aware.

The second study leverages the CONTROLS and MMA providers, where M^{3}CT is the principle requestor. This implementation gives M^{3}CT the responsibility of acting as a logic gate controller for the study. Input Decks are passed to each provider, and Output Decks are returned to M^{3}CT. M^{3}CT is responsible for interpreting these Output Decks and forwarding them to the next provider in the pre-planned sequence. In order to retain history, each Deck iteration is saved to each providers hard drive, which is then manually stitched together with the updated report from their respective providers underlying service.

JavaSpace is used in both implementations, which is where CONMIN was approached to see if both studies could leverage the providers services uniquely. However with the first study and second study being fundamentally different in implementation, the JavaSpace commonality was insufficient. The first, passing provider data through JavaSpace. The second, passing URL locations to Decks through JavaSpace. In both implementations, the naming, order, and implementation of variables were also vastly different. i.e., The Kinematic Parameters, although mathematically identical in purpose, were stored inconsistently between the various providers.

Case management will require a broader approach then what was considered within the scope of the M^{3}CT effort. In order for disparate systems of systems to be able to communicate consistently, a schema needs to be researched and established that can properly compartmentalize universally acceptable standards for common definitions. Schema typically can be enforced through XML or JSON syntax. The use of Metadata can also lead toward the automation of case management, alleviating some probabilities of human error.

4.6 Study Examples

4 study examples are presented herein to document results obtained with M^3CT and integrated providers:

1. Pinned MAV analysis (QSCSD)
2. Pinned MAV optimization (QSCSD and CONMIN)
3. Free-to-move MAV with pinned constraint analysis (Controls)
4. Free-to-move MAV trim analysis (Controls)

Results of these study examples are summarized in the following sub-sections. Before presenting these results, we review nomenclature concerning the analysis.

Multiple (4) coordinate systems are defined, as shown in Figure . The MAV wing and body are tracked in an inertial, global frame of reference denoted by $I = \{I_X, I_Y, I_Z\}$, where I_Z corresponds to altitude (positive up). The vehicle body frame $R = \{R_X, R_Y, R_Z\}$, is attached to the MAV body (at the center-of-mass), and rotates and translates with respect to the global frame. The positive direction of the body frame component R_Y is aligned with body rotation axis and extends in what is considered the vehicle's forward flight direction. The positive R_X extends out the right wing (in an un-rotated position). The wing frame, $W_n = \{W_{nX}, W_{nY}, W_{nZ}\}$, where n is the wing number (right=1 or left=2), is rotated about R following a 3-2-1 Euler angle convention. In 3-2-1 order, the three Euler angles are ϕ (azimuth or stroke), θ (elevation or deviation), and η (pitch or feather).

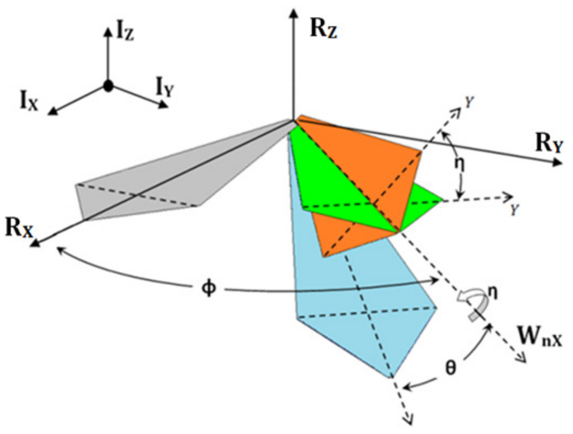


Figure 17. MAV Coordinate System

The geometry of the free-to-move MAV wing (both left and right wings are treated equivalently) is defined by several parameters: the wing radius (B_W; semi-span) of the wing, the chord (and thickness) at three sections, span-break ratio (SBR), and the geometry mode flag. The wing radius is measured along the center span line (50% chord line), as shown by the span-wise dotted line in Figure . The wing root chord (C_O), span-break chord (C_S), and wing tip chord (C_N) define the chord distribution, and are assigned values in meters. The wing root thickness (T_O), span-break thickness (T_S), and wing tip thickness (T_N) define the thickness distribution, and are assigned values in meters. The three chord sections are initialized parallel with each other and lying on the X-Y plane as shown in Figure . The chord lengths and thicknesses at any other point along the wing may be linear interpolated between the two adjacent prescribed sections. The span-break position is equivalent to SBR * B_W. The wing geometry mode specifies whether the wing planform is to be defined asymmetrically (Figure , top) with a linear leading edge or symmetrically (Figure , bottom) across the wing 50% chord line.

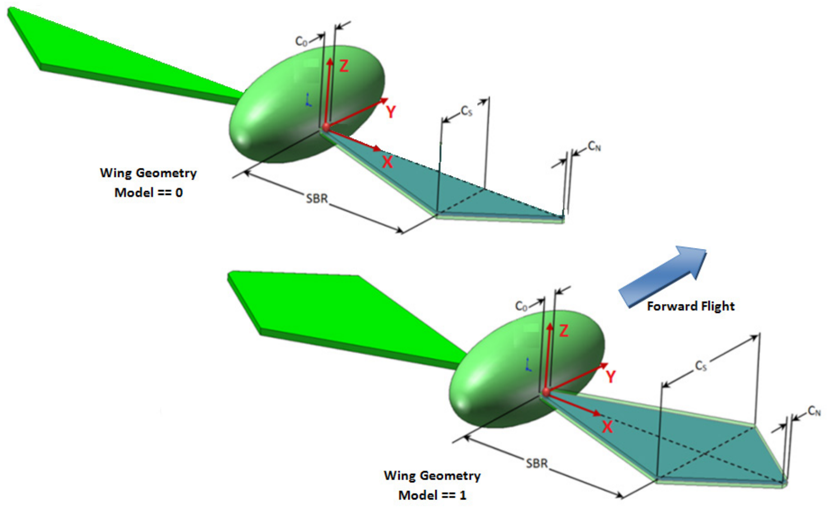


Figure 18. Free-to-Move MAV Wing Geometries

6 additional parameters are introduced to represent the positioning of the wing roots relative to the body center-of-gravity: δWn = {δWnX, δWnY, δWnZ}, where n is the wing number (1 or 2). In other words, the point at which the wing rotates can be moved. These values are interpreted as the wing root location minus the body center-of-gravity location in the body coordinate system.

In the Controls module, access is given to the parameters of the quasi-steady aerodynamics model, which follows that of Berman and Wang[iii]. The values used in the Controls study examples below use the values provided in Table 1, as set in the Berman-Wang entry location of M3CT.

Table 1. M^{3}CT Parameter Values for the Berman-Wang Dialogue

Parameter	Value
Translational Force Coefficient	1.833
Rotational Force Coefficient	3.142
First Viscous Term	0.200
Second Viscous Term	0.000

Kinematic and split-cycle parameters are grouped together and assigned Identification (ID) numbers, as recorded in Table 2 (with units; hereafter, "ND" refers to non-dimensional).

Table 2. M^{3}CT Parameter Identification (ID) Numbers

ID Number	Parameter (Units)	Description (Name)
1	ϕ_m(rad)	Stroke Amplitude (PHI_M)
2	ϕ_0 (rad)	Stroke Offset (PHI_O)
3	K_ϕ (ND)	Stroke Sharpness (K_PHI)
4	θ_m (rad)	Deviation Amplitude (THETA_M)
5	θ_0 (rad)	Deviation Offset (THETA_0)
6	θ_s (rad)	Deviation Phase-Shift (THETA_S)
7	η_m (rad)	Feather Amplitude (ETA_M)
8	η_0 (rad)	Feather Offset (ETA_0)
9	η_s (rad)	Feather Phase-Shift (ETA_S)
10	K_η (ND)	Feather Sharpness (K_ETA)
11	ω (rad/s)	Flapping Frequency
12	δ (ND)	Wing-Beat Bias

Additional parameters, including the wing geometry parameters, are defined in Table 3.

Table 3. Other M^{3}CT Parameter Descriptions

Parameter (Units)	Description (Name)
ρ (ND)	LQR coefficient
SBR (%)	Span-Break Ratio
B_W (m)	Wing Radius
N_θ (ND; 1 or 2)	Deviation Frequency Coefficient

4.6.1 Pinned MAV Analysis (QSCSD)

In this study example, M^{3}CT is used to perform analysis of a pinned MAV in hover with the QSCSD provider. A flapping frequency (ω) of 125 rad/s is prescribed, and the wing is decomposed into 10 strips. The parameter values of this example are shown in Table 4. Parameter specifications are divided according to the M^{3}CT entry locations, or "M^{3}CT Dialogues".

For convenience, minimum and maximum values of each parameter are also shown in the table, and these values correspond to the side constraints enforced in the pinned MAV optimization (Section 4.3.2). Please note that the values prescribed here are different than the initial conditions used in the Pinned MAV Optimization study example described below.

Table 4. Pinned MAV Analysis Parameters (and Side Constraints for Optimization)

M^3CT Dialogue	Parameter	Value	Min	Max
Kinematics	ϕ_m	1.047	-π/2	π/2
	ϕ_0	0.000	0.0	0.0
	K_ϕ	0.010	0.01	0.5
	θ_m	0.000	-π/4	π/4
	θ_o	0.000	-π/4	π/4
	θ_s	0.000	-π/2	π/2
	η_m	0.785	-π/2	π/2
	η_o	1.571	-π/2	π/2
	η_s	-1.571	-π/2	π/2
	C_η	0.100	0.01	1.0
Wing Geometry	Chord (m), Root to Tip	2.250e-02	0.01	0.06
		3.238e-02	0.01	0.06
		3.978e-02	0.01	0.06
		4.472e-02	0.01	0.06
		4.719e-02	0.01	0.06
		4.719e-02	0.01	0.06
		4.472e-02	0.01	0.06
		3.978e-02	0.01	0.06
		3.238e-02	0.01	0.06
		2.250e-02	0.01	0.06
	Thickness (m), Root to Tip	4.300e-04	0.0002	0.0025
		5.209e-04	0.0002	0.0025
		5.890e-04	0.0002	0.0025
		6.344e-04	0.0002	0.0025
		6.572e-04	0.0002	0.0025
		6.572e-04	0.0002	0.0025
		6.344e-	0.0002	0.0025
			0.0002	0.0025
			0.0002	0.0025
			0.0002	0.0025

		04 5.890e-04 5.209e-04 4.300e-04		

Results are included in Table 5. The reader should note that analysis results are grouped according to objective and constraint, following what is defined in the next study example.

Table 5. Pinned MAV Analysis Results

Parameter (Units)	Value
Cycle-Averaged Power (Normalized, W/N)	15.25
Cycle-Averaged Lift (N)	0.241
Peak Stress Indicator (kS)	1.412
Peak Power During Final Cycle (W)	19.37

4.6.2 Pinned MAV Optimization (QSCSD and CONMIN)

In this study example, M^3CT is used to perform optimization of a pinned MAV in hover with the QSCSD and CONMIN providers. The goal of this study is to minimize the cycle-averaged aerodynamic power while meeting a lift constraint of 0.15 N, assuming a sustained flapping frequency of 125 rad/s. See Table 6.

Table 6. Pinned MAV Optimization Goals

M^3CT Category	Parameter (Units)	Target
Objective	Cycle-Averaged Power (Normalized W/N)	Minimize
Constraint	Cycle Average Lift (N)	0.15

The settings of parameters used by CONMIN are defined in Table 7. The results of the optimization process are shown in Table 8. The reader should note that cycle-averaged power required was reduced by more than 65% of the initial value. This reduction was accomplished by changing the design variables as well as reducing the lift associated with the initial values of the design variables, which was found to be 0.151 N. Thus, power was reduced by over 65% while reducing hover lift by 37% to the constraint value (i.e., lift constraint is active). In other words, the lift corresponding to the initial values of the design variables is more than needed.

Originally, a peak stress constraint was added in the optimization process. The peak stress indicator was found to naturally decay throughout the kinematic optimization. The reduction in stress can be viewed by comparing the initial stress indicator to the final value in Table 8; as a result it was determined that removing the stress constraint better served the optimization process.

Table 7. CONMIN Parameter Settings

M^3CT Dialogue	Parameter Description	Value
RPRM	Constraint Active Value	-0.030
	Constraint Min Violation (CTMIN)	0.003
	Max absolute change (DABOBJ)	0
	Max relative change (DELOBJ)	0.001
	Absolute Change Objective Function (DOBJ1)	0.100
	Relative Change Objective Function (DOBJ2)	0
	Max Relative Change DV (DX1)	0.01
	Max Absolute Change DV (DX2)	0
	Relative finite difference step (FDCH)	0.001
	Min Abs value of finite difference step (FDCHM)	0.0001
	Max relative change in DV (Lin Pgm Meth (RMVLMZ)	0.4
	Max absolute change in DV (Lin Pgm Meth) (DABSTR)	0
	Max relative change in DV (Lin Pgm Meth) (DELSTR)	0.001
	NEWITR change when	0
IPRM	Gradient Calculation Method (IGRAD)	User
	Maximum Optimization Iterations (ITMAX)	200
	Number Consecutive Criteria Iterations (ITRMOP)	3
	File Output Number (JWRITE)	9
	Number of Retained Constraints	0
	Design Variable Scaling	Active and Violated Only
	Max Number Iterations Sequential (Linear|Quad)	50
	Consecutive Iterations Convergence	0
Objective	Inner Loop Evaluators	pMaxNorm=evaluateExertion
	Outer Loop Evaluators	pMaxNorm=evaluateExertion
	Gradient Inner Loop Evaluators	pMaxNorm=evaluateMethod
	Gradient Outer Loop Evaluators	pMaxNorm=evaluateMethod
Constraints	Constraints	ksMaxNorm LavgNorm
Loop Control	Maximum Outer Loop Iterations (ILITMAX)	10
	Maximum Inner Loop Iterations (OLITMAX)	50

Table 8. Pinned MAV Optimization Results

	Parameter (Units)	Initial	Optimized
Objective	Cycle-Averaged Power (Normalized, W/N)	15.25	5.22
Constraints	Lift (N)	0.241	0.151
	Peak Stress (kS)	1.412	3.989e-03
Kinematic Design Variables	Azimuth Amplitude (PHI_M)	1.047	0.776
	Azimuth Sharpness (K_PHI)	0.010	0.010
	Elevation Amplitude (THETA_M)	0.000	0.055
	Elevation Offset (THETA_O)	0.000	-0.055
	Elevation Shift (THETA_SHIFT)	0.000	0.011
	Pitch Amplitude (ETA_M)	0.785	0.945
	Pitch Offset (ETO_O)	1.571	1.571
	Pitch Phase (ETA_SHIFT)	-1.571	1.145
	Pitch Squareness	0.100	0.102
Wing Chord Design Variables	Chord, root to tip (m)	2.250e-02	2.331e-02
		3.238e-02	3.227e-02
		3.978e-02	3.956e-02
		4.472e-02	4.433e-02
		4.719e-02	4.664e-02
		4.719e-02	4.660e-02
		4.472e-02	4.430e-02
		3.978e-02	3.967e-02
		3.238e-02	3.267e-02
		2.250e-02	2.320e-02
Wing Thickness Design Variables	Thickness, root to tip (m)	4.300e-04	5.343e-04
		5.209e-04	3.970e-04
		5.890e-04	5.463e-04
		6.344e-04	5.967e-04
		6.572e-04	5.981e-04
		6.572e-04	5.770e-04
		6.344e-04	5.434e-04
		5.890e-04	4.991e-04
		5.209e-04	4.424e-04
		4.300e-04	3.715e-04

After execution, the raw data stored after each optimization cycle can be found in the data directory under iGrid-08. Each of these folders contains detailed data files of the model parameters. The results tabulated in Table 8 compare the initial values of the design variables to the optimized values for all 29 design variables.

Wing chord and thickness distribution, initial and optimized, are compared in Figure . Instantaneous power and lift at the optimal design point are shown in Figure . To minimize the resultant required power, the optimal kinematic motion exploits the aeroelastic interaction between the wing and the air surrounding the wing. The variation of the three Euler angles through the wing stroke at the optimal design point is shown in Figure 21. Here it can be seen that minimizing power requires angular deviation from the stroke plane (variations of two Euler angles not sufficient). The motion of the wing is shown in Figure , in terms of the stroke path of the wing tip. This motion combines the effects of wing kinematics and wing deformation, and is compared to the motion of the wing tip corresponding to a rigid wing moving with the kinematics corresponding to the initial design point. Structural deformation accounts for the large region enclosed by the motion of the wing tip. Also, it should be noted that the wing tip is biased to positive values of z, owing to the constrained value of the cycle-averaged lift force (also in the direction of positive z), and that the instantaneous lift force never vanishes.

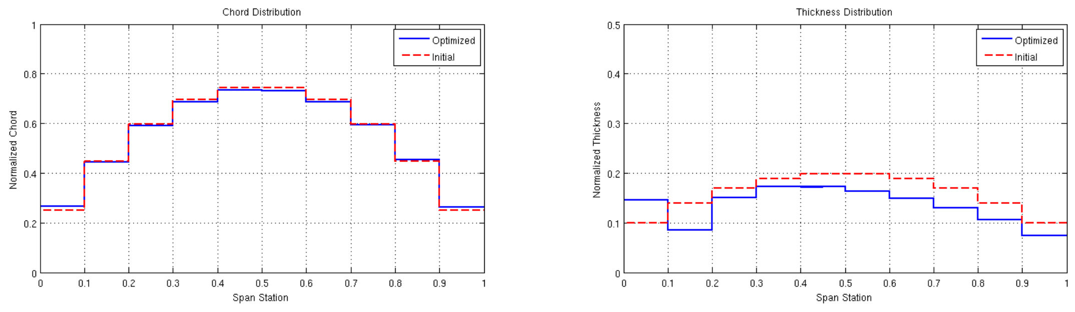


Figure 19. Wing Chord (left) and Thickness (right) Distribution Results

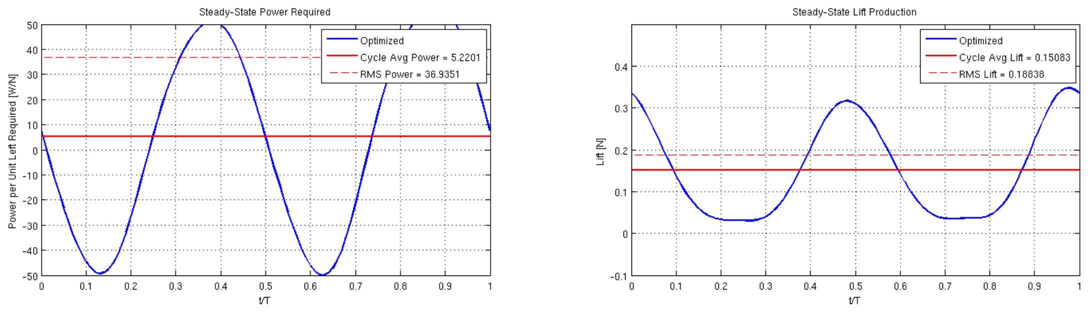


Figure 20. Optimized Instantaneous Power (left) and Left (right) with Cycle-Averaged Values

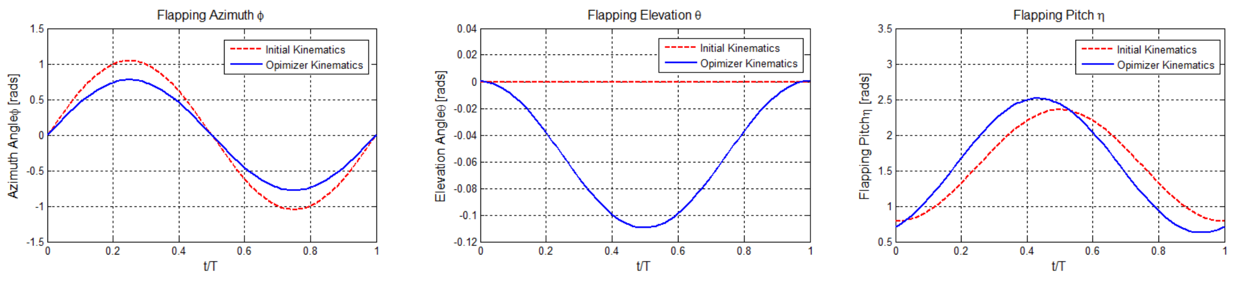


Figure 21. Instantaneous Kinematic Motion at the Optimal and Initial Design Points

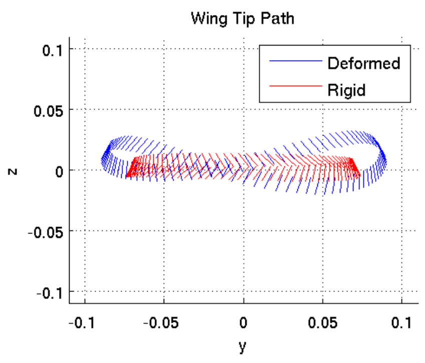


Figure 22. Wing Tip Trace for Aeroelastic Wing at Optimal Design Point and for a Rigid Wing at the Initial Design Point

4.6.3 Free-to-move MAV with Pinned Constraint Analysis (Controls)

The remaining case studies are examples of utilizing M^3CT and the Controls module to perform analysis and optimization of a free-to-move MAV with six degrees of freedom (three in translation and three in rotation). The trim analysis is configured for a hover state.

The Controls module manipulates the body-frame orientation to obtain trim in either hovering or forward flight. An image depicting the changes in orientation with respect to the inertial frame for a given flight mode is shown in Figure . The forces and moments generated in the analysis (and reported below) are presented with respect to the MAV body frame. As a result of the changes in orientation (here, the gravity vector always points downward in the inertial frame; offset between the wing rotation point and the center of mass rotates the body in hover to produce a body-up orientation), for hovering flight, the lift force should be directed along the body frame axis R_Y. In forward trimmed flight, much of the lift should be directed along the R_Z axis.

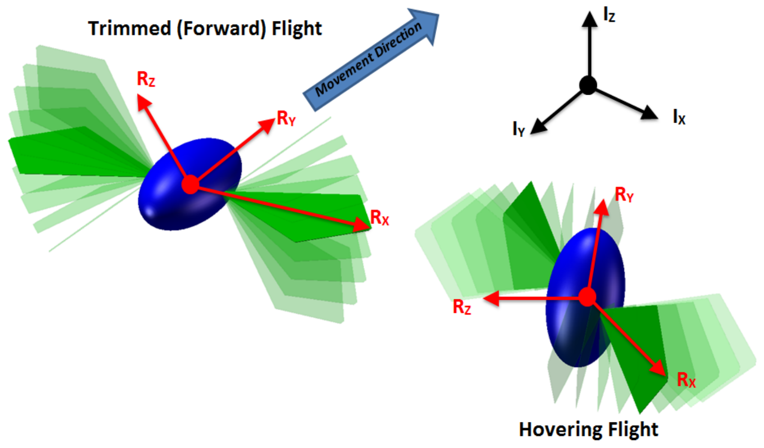


Figure 23. Trim in Either Forward Flight or Hover

For each of the case studies discussed below, the parameters are specified in a manner consistent with the common Hawkmoth (Manduca sexta). Kinematics and wing geometry are approximated, the body mass (now relevant) is specified to be 3.75×10^{-3} kg, the wing span (radius) is specified to be 5.30×10^{-2} m, and the flapping frequency is specified to be 85 rad/s (approximately 13.5 Hz). Other parameter values are given in Tables 10 - 14. Please note that in the Controls module, new dialogues are available to the user. In addition to the Berman and Wang dialogue defined above, “body”, “environment”, “power” dialogues are added to enable the user to define body properties needed for rigid-body analysis in a gusty environment and to enable the user to characterize power losses in the MAV transmission.

In this first case, the Controls module is exercised for a MAV that is pinned, before releasing the physical constraint and allowing the vehicle to achieve trim. This capability is not accessible in the M^3CT version of the Controls module, but is accessed directly via MATLAB, the native language of the Controls module. This capability can be made accessible to M^3CT, following software modification.

Table 9. Parameter Settings for Pinned Analysis of a Free-to-Move MAV (Wing Dialogue)

Parameter (Units)	Value
SBR (ND)	0.7000
Wing Density (kg/m^3)	156.00
Number of Wing Elements	10
Trim Forward V (m/s)	0.0
Trim Climbing V (m/s)	0.0
B_W (m)	0.0530
C_0 (m)	0.0080
C_S (m)	0.0117
C_N (m)	0.0139
T_0 (m)	0.0003
T_S (m)	0.0003
T_N (m)	0.0003
Wing Geometry Mode (ND)	1 (symmetric)
δW_{1x} (m)	0.015
δW_{1y} (m)	0.030
δW_{1z} (m)	0.000
δW_{2x} (m)	-0.015
δW_{2y} (m)	0.030
δW_{2z} (m)	0.000

Table 10. Parameter Settings for Pinned Analysis of a Free-to-Move MAV (Body Dialogue)

Parameter	Value
Body Mass (kg)	0.00375
Body Ellipsoid Major (m)	0.0700
Body Ellipsoid Minor (m)	0.0140
Pure Longitudinal Only	False (deselected)

Table 11. Parameter Settings for Pinned Analysis of a Free-to-Move MAV (Environment Dialogue)

Parameter	Value
Analysis Mode	Calculate cycle-averaged power
Gravity Acceleration (m/s^2)	-9.81
Air Density (kg/m^3)	1.225
RBM Solver Error Tolerance	-10.0
RBM Time Integration Beta	0.50
Floquet Solver Error Tolerance	-10.0
Number Time Intervals / Period	60
Drag Coefficient ($C_{D,0}$)	0.21
Drag Coefficient ($C_{D,\pi/2}$)	3.35

Table 12. Parameter Settings for Pinned Analysis of a Free-to-Move MAV (Kinematics Dialogue)

Parameter	Value
ϕ_m	$\pi/3$
ϕ_o	0
K_ϕ	0.01
θ_m	0.001
θ_o	0
θ_s	0
η_m	$\pi/4$
η_o	$\pi/2$
η_s	0
K_η	0.10
ω	97.98
δ	0
N_θ	1

Table 13. Parameter Settings for Pinned Analysis of a Free-to-Move MAV (Power Dialogue)

Parameter (Units)	Value
Power Elastic Storage Parameter (ND)	0.50
Power Dissipation Cost Parameter (ND)	1.00
Power E Coefficient (ND)	0.10
Power K Coefficient (ND)	20.0

The pinned response is tabulated in Table 14. It should be noted that both the x and y components of the body force vanish (y component approximately so), and that the sum of the z-component of force is the cycle-averaged lift. In this case the stroke plane is oriented such that the body axis lies in the stroke plane.

Table 14. Performance Characteristics of the Free-to-Move MAV when Pinned

Output Parameter	Value
Cycle-Averaged Forces (N)	
Left Wing	[6.464e-04 -1.344e-06 3.426e-03]
Right Wing	[-6.464e-04 -1.344e-06 3.426e-03]
Total Lift	0.0069
Cycle-Averaged Moments (Nm)	
Left Wing	[1.028e-04 1.746e-04 -1.928e-05]
Right Wing	[1.028e-04 -1.746e-04 1.928e-05]
Cycle-Averaged Power (W)	
Left Wing	7.250e-03
Right Wing	7.250e-03
Total	1.450e-02

4.6.4 Free-to-move MAV Trim Analysis (Controls)

The constraint of pinning the MAV is now removed, and a trim state of the vehicle is computed. Initial values for the trim calculation are shown in Table 15, along with side constraints on the different parameters used later in optimization. In Table 16, additional parameters are defined that appear in the M^{3}CT kinematics dialogue. The trim IDs specify which two parameters become variables in the trim computation (i.e., these parameters are varied to meet the trim constraints of cycle-averaged lift balanced by weight and vanishing cycle-averaged pitch moment on the vehicle). In this case trim IDs are 1 and 4, indicating that ϕ_m and θ_m are the two trim variables. Kinematic control IDs correspond to those kinematic parameters that are used for closed-loop control (stroke amplitude and bias, in this case). Finally, the kinematic CAS IDs are the cycle-averaged force sensitivities to the specified kinematic IDs that are reported in the detailed Controls module output.

Trim results are presented in Table 17 in terms of forces and moments. It should be noted that the cycle-averaged lift well matches the vehicle weight.

Not shown in this table is the change in orientation of the MAV. When pinned, the MAV is positioned as shown in Figure (top), and lift is produced in the z-coordinate direction, which is equivalent for body and inertial coordinate systems. However, when trimmed in hover, the body of the MAV points upward (body y-coordinate axis aligned with the inertial x-coordinate axis), and lift is produced in the body y-coordinate direction. This occurs because the wings are located ahead of the body center of gravity, and the gravitational moment pitches the body. The final values of wing stroke and deviation amplitude arising from the trim computation are 0 (-1.2e-05) and 2.456 rad, respectively. The body becomes pitched at an angle of 4.703 rad, or 90 deg (straight up). In forward flight, the aerodynamic forces on the body, along with that of the wings, balances gravity in a manner that changes the orientation of the body with respect to the inertial frame.

Table 15. Parameter Values for the Analysis of a Free-to-Move MAV in Trimmed Hover

Parameter	Initial Value	Minimum Value (Opt. only)	Maximum Value (Opt. only)
Kinematics			
ϕ_m	π/3	-π/2	π/2
ϕ_o	0	0	1.0
K_ϕ	0.01	0.01	0.99
θ_m	0.001	-π/4	π/4
θ_o	0	-π/4	π/4
θ_s	0	-π/2	π/2
η_m	π/4	-π/2	π/2
η_o	π/2	-π/2	π/2
η_s	0	-π/2	π/2
K_η	0.10	0.01	4.0
ω	97.98	62.83	314.2
Controller			
ρ	0.001	$1\text{x}10^{-10}$	10.00
Geometry			
C_0	0.047	0.008	0.100
C_S	0.047	0.005	0.100
C_N	0.047	0.005	0.100
SBR	0.581	0.125	0.875

Table 16. Additional Parameter Settings for Trimmed Analysis of a Free-to-Move MAV

Description	Value
Kinematic Trim IDs	1, 4
Kinematic Control IDs	1, 2
Kinematic CAS IDs	1, 2, 7, 8, 11, 12

Table 17. Performance Characteristics of the Free-to-Move MAV in Trim

Output Parameter	Value
Cycle-Averaged Forces (N)	
Left Wing	[-9.359e-04 1.852e-02 2.484e-08]
Right Wing	[9.359e-04 1.852e-02 2.4837e-08]
Total Lift (N)	3.704e-02 (*0.00378 kg*)
Cycle-Averaged Moments (Nm)	
Left Wing	[2.741e-12 6.6072e-10 -6.582e-04]
Right Wing	[2.741e-12 -6.6072e-10 6.582e-04]
Cycle-Averaged Power (W)	
Left Wing	1.295e-01
Right Wing	1.295e-01
Total	2.591e-01

4.6.5 MUAV Closed Loop Flight in Gust (Controls)

The remaining closed-loop case studies are contained in the STEAP TO-49 Final Report.

5.0 TECHNOLOGICAL CHALLENGES FOR EXTENDING FSI

OBJECTIVE II

There are numerous technological challenges faced in extending flapping sciences integration to a wider range of engineering design problems. This chapter is approached from a computer science perspective, looking at various aspects of the engineering design process as capability requirements. Here, "process" is not meant to be a specific process, such as that employed by some partner in industry, but is meant to represent a dynamic set of engineering operations, linked together in a framework, by a user to meet an engineering goal (e.g., minimize power requirement while meeting a lift constraint).

Of interest are complex systems of equations that need to be solved in an efficient manner, i.e., realistic calculation times. Part of the cost is driven by ensuring that the results have an acceptable error. Another part of the cost comes from the reassembly of parts back into a final result. We desire the design capability to have a cost balance between these two contributing cost factors.

Throughout the design process, there needs to be methods for monitoring divergence during execution, leveraging individual toolsets at various fidelities, and being able to configure the executional order of the systems.

After the processes have been completed, they must be repeatable. This means that the system must be capable of accessing older versions of those toolsets. This is particularly problematic, since method development and tool use generally occur on different time scales. Documentation and understanding of how to leverage these tools must be preserved. Additionally, hardware and software systems available to the network must be capable of running them on a compatible platform in the future.

5.1 Errors from Approximation

The process of evaluating a solution curve as a manageable set of segments, without losing the context of the measurement, is a process called discretization. Discretization introduces an error into the system of differential equations (DEs), an approximation error known as discretization error.

$$\boxed{\widetilde{DE} = DE + Error}$$

Figure 24. Discretization Errors Modify the Governing Equations

The fidelity of a system of equations representative of a physical process is inversely correlated with another approximation error, modeling error (higher fidelity usually, but not always, implying lower error). For example, the blade-element theory used to describe the aerodynamics in FSI is generally thought to be of low-fidelity, since it doesn't capture certain aspects of the physical process related to wing flapping. In comparison, the laminar Navier-Stokes equations are a much higher fidelity set of equations, at least for non-turbulent flows, since they are known to describe all the flow physics produced by wing flapping, including the interference effects explicitly not modeled by blade-element theory.

The nature of the design study has within its essence, a need to push the boundary between computational performance and accuracy, thus generating the tipping point (or maybe thought of as a “sweet spot”) where errors start to become unacceptable: we don’t want to spend to much time in analysis if not warranted. However, it is not known *a priori* where this tipping point occurs.

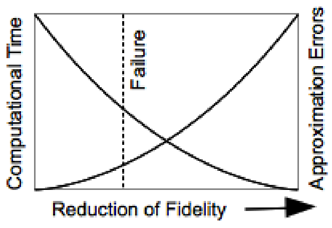


Figure 25. Fidelity Reduction Against Error Relationships (Notional)

The capability requirement set that arises is the need to: change fidelity within the framework; evaluate, or at least score, the accuracy of the resulting analyses, and track computational time consumed in analysis and optimization steps. Only if these requirements are met can the process be made dynamic and extensible for a wide variety of engineering design problems.

5.2 Incompatibility from Reduction

To further reduce the amount of computational time required across a system of equations, Reduced Order Modeling and other methods of model reduction help to generalize well documented phenomena that would otherwise be extremely complex to model. This is an established process, however the implementation of this process can vary from provider to provider.

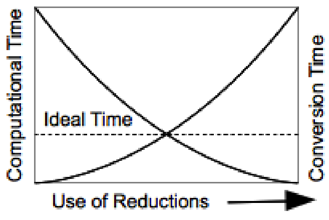


Figure 26. Computational Reductions Against Time Relationships

In order to leverage various methods of reduction, the underlying system of equations must be convertible. That conversion process itself must not introduce any significant errors, and the computational time of that conversion should ideally not require more effort than the reduction alleviates.

Cobbling together disparate reduction methodologies into a single cohesive context reference was weighted upon proprietary interfaces at the individual providers level. This is not the correct placement of function for a conversion. The workflow controller must handle these conversions between the output of the previous providers and the input of the next provider.

5.3 Many to One Context Collaboration

The general workflow of the M^3CT was established as a single provider throughput to another single provider. This encouraged proprietary connectivity between the interfaces of the two. Additionally, the sequential nature of the relationship encouraged 4th wall breaking implementations. Programmatically speaking, the 4th wall is an imaginary boundary between objects. Objects are designed to communicate through interfaces, inheritance, and hierarchal relationships. When you connect to another object directly, not using a prescribed method, you are violating the 4th wall. In theatre, this is a moment when the actor speaks directly to the audience, instead of staying within the boundaries of the stage. This should not be confused with a soliloquy, in which the actor is revealing their internal thoughts to those who would listen.

It is explicit in a multidisciplinary environment that numerous outputs are going to be leveraged by a single input, often recursively, as the system of equations moves toward a solution. It is therefore imperative that a collection of numerous outputs can be effectively mapped to a collection of inputs. Mapping these relationships requires that the outputs and inputs be ordered in a predictable way.

Conversions and integrations of data types will remain to be a large challenge of collaboration. These conversions can somewhat be alleviated through the ability to inject adaptive logic between the outputs and input requirements. With multiple inputs having the probability of contributing to a single input, these adapters must be capable of accepting multiple outputs from multiple contexts.

5.4 Decomposition and Synchronization

Decomposition is the practice of taking a large system and breaking it down into smaller systems. Generally, the interrelationships between these subsystems become more intimate and sequenced as you attempt to break them down. The benefit of calculating these smaller systems can come from different sources; e.g., faster component execution, more atomistic definitions, and sparser matrix representations of discretized equations. Reconstituting these parts back into a solution can be a computational challenge and a source of error. For example, if the problem of flutter is decomposed into a structures problem and an aerodynamic problem, each solved separately, errors may arise from synchronization delays in the analyses.

Where possible, all systems of equations should be broken down into their lowest possible module. The end user needs a way of understanding how to arrange these modules in an effective composition. Furthermore, the interrelationships between modules need to be generic enough in that any comparable module can easily be swapped in place of another. Without this latter definition being applied to the concept, these systems cannot be considered modules, but rather only a collection of proprietary processes in a fixed order of reassembly. Given time constraints and the co-development of methods under FSI-Controls, the latter situation better characterizes M^3CT.

Having a subsystem of equations that ultimately represent one larger system of equations explicitly implies that all of these subsystems must share a common iterative synchronization. This synchronization requirement adds yet another level of complexity on the development of modules due to the disparity between iterative scopes. One module requiring 10 steps, stitched together with another module requiring 20 steps is a simple 1:2 ratio. These special cases are not reliable against the larger case such as 10:11 ratios, where the calculation will require some form of extrapolation or interpolation between micro-iterations of the formula, which introduce yet another element of error.

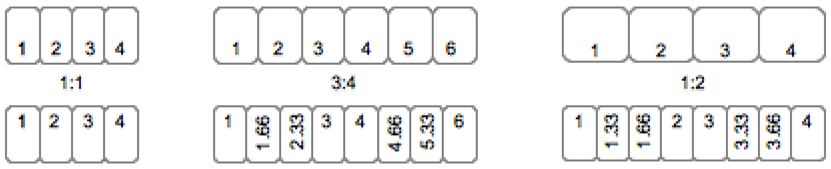


Figure 27. Asymmetrical Synchronization of Iterative Values

Another such disadvantage endured through the process of decomposing a system, is the potential usage of semaphores. This becomes a requirement when two or more systems require access to the same contextual value at the same time. It is easy to place one system after the other when only one semaphore condition is being averted, however duplex thread relationships merit that a race condition can very well exist in the presence of numerous semaphore, and a specific synchronization pattern would need to be enforced. This weighs heavily upon the end user of the providers to configure correctly, and would provide less difficulty as a combined system.

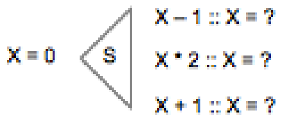


Figure 28. Interpreting the Asynchronously Decomposed Value of X

Ultimately the process of mathematical decomposition and synchronization should follow the same methodologies and constraints of a multithreaded software application. These methodologies must be converted into the realization and scope of the FSOOA operational environment. This means that the multithreading techniques are not limited to the local conditions of a processor unit, but instead are extrapolated over the entire federation. This is an extremely broad and volatile scope.

It will be necessary in moving forward with multidisciplinary development to establish criteria for defining mathematical system modules, their methods of internal synchronization, and how they define their exposed contexts for external runtime manipulation. Further, these modules must be assembled back into an accurate representation of the principle system of equations without further degradation of accuracy.

5.5 Model Matching

The composition of exertion work is ultimately limited by the model format of the providers registered to the FSOOA. Broadly segregating the available list are providers that anticipate two-dimensional (2D) or three-dimensional (3D) calculations. These differences are notable in both rendered objects and algorithm calculations such as for fluid dynamics. False positives and false negatives can lead to failed research.

The models themselves may expect a triangular mesh, quadrilateral (Quad) mesh, or a Non-Uniform Rational B-Spline (NURBS) mesh. Furthermore, each provider may have a unique vertex format anticipated for the vertices of the mesh that are considered incomplete or excessive definitions to other formats.

A common adaptive solution example would be matching a provider using standard units to a provider using metric units. Another would be changing orientation, rotation, and scale.

There are efforts that exist apart from the scope of the M^{3}CT project dedicated toward the discovery of a generic approach to developing an aeroelastic provider module. Such a capability was developed in MDICE[vii], and involves satisfaction of fluid-structure interface conditions. A more robust approach to filtering providers AND satisfying interface conditions, based on model expectations over a broad network, is needed.

5.6 Data Reporting and Aggregation

Providers produce a great deal of context, and one user may want to see different aggregations then another user. The critical failure point comes from requiring M^{3}CT software to do this aggregation without knowing how to interpret the data. In deployed networks, each product is responsible for sending out various channels of information. The network doesn't need to know everything all the time. One channel can simply be a status, and another channel could be a full data stream. The number of channels provided and what those channels hold has been framed out many times over many various implementations.

The Context Object of the SORCER framework has the ability to provide this type of compartmentalized data. The ability to provide aggregated views relies entirely upon the provider of the service correctly. M^{3}CT is fully capable of showing the aggregated data simply by the nature of which context it is accessing.

In the current implementation, only one context is provided from the service per data set, containing all of its information. This makes aggregation impossible, or incorrectly leveraged, upon the M^{3}CT product. In all future implementations, the service provider should be able to produce as many context objects as there are reasonable needs for aggregation. Such accuracy is impossible otherwise, or incorrectly requiring an external tool to perform the aggregation.

One possible interpretation of SORCER's capabilities, and the channel analogy, is for a service provider to use the following context labeling convention:

- **Temp** – The input context currently in use by the provider. This is a local copy of what was pulled from the network and should be disposed after use.
- **Progress** – A context containing select iterations and their current/target values, ideal for looping services and progress indicators.
- **Full** – The full context of the service, possibly never transmitted, retained for historical analysis and archival.
- **All** – The full report includes all values from all iterations.

- **Last** – The full report only includes the values of the last iteration.
- **Report** – The full context of the service reduced programmatically (Using a Query Syntax) by the requestor to a variable set of defined needs. Within a report:
- **Poll** – A context containing only the current data of each value, ideal for iterative observation and error point analysis.
- **Ranged** – A context containing select values over a range of data points, ideal for the display of large data set aggregation (Hi-Lo, Min, Max, etc.)
- **Reduced** – A context containing a reduced set of data that is representative of the full context, ideal for the display of large model aggregation or adapting between fidelities.
- **History** – A context formed similarly to a REPORT as a programmatically reduced set, but from history.

The job id and sequence id can be used to prevent collisions of similar processes, allowing for history to be concise, and for non-concurrent workflow to be aggregated gracefully.

6.0 CONCLUSIONS

OBJECTIVE I

The goals of the research task can be coarsely summarized as the development of sufficiently sophisticated FWMAV models, control design in a 6DOF simulation, and implementations of prototype FWMAV experiments. The theoretical contribution can be seen in our published works. It includes the development of quasi-steady 6DOF FWMAV models, amenable to analytic analysis to produce simplified control models, and for controlled closed-loop simulation. The models were used to guide the selection of potentially viable control inputs in the form of stroke manipulations. The results were stroke parameter suits capable of producing 4-5 DOF control, such as: independent frequency, split-cycle, and stroke-bias control for each of the two wings. Controllers were designed which were shown in simulation to provide guidance between waypoints in 3D space under a 6DOF simulation.

The fabrication and testing of a prototype FWMAV demonstrated the feasibility to control up to four degrees-of-freedom using only two actuators and stroke velocity manipulation, namely frequency and split-cycle stroke modulation. A prototype FWMAV was mounted on a puck and placed on an airtable. The split-cycle modulation was then adjusted using a remote controller, demonstrating the ability to use symmetric and asymmetric split-cycle modulation to control forward/backward translation and clockwise/counter-clockwise rotation. These results are currently being documented and have in part been accepted for publication, further publications pending.

The research successes include contributing theoretical models for analysis and design usage, four to five degree-of-freedom control designs amenable to future implementation, and the proof-of-concept milestone achievement of controlled motion by the prototype FWMAV on the airtable. Future investigations would be advised to consider further robustness analysis, the inclusion of onboard sensors for control feedback, weight and power concerns, the ability to produce more cycle-averaged lift than weight, and ultimately a prototype in controlled untethered flight.

OBJECTIVE II

In order to properly scope the challenges facing development of a design-interface such as M^{3}CT, a number of subject matter specifics must be considered. This section emphasizes some of the critical relationships that need more accommodation in a generalized, engineering design approach.

Here it is clear that the emphasis is on the future of M^{3}CT, instead of on the alpha-version of M^{3}CT developed for the purpose of creating a user experience for design of micro air vehicles. Integration of the FSI-Methods and FSI-Controls components provided a useful engineering backdrop that motivated certain functionality. Results for different use cases are summarized in the Section on Program Results, as well as in the M^{3}CT Users' Guide. Results are found to be in good agreement with stand-alone (i.e., non-network) of FSI methods and tools. Additional applications of FSI components can be found in a companion delivery order report entitled "Computational Prototyping of Micro Air Vehicles," authored by Gary Clayman (Data Science Automation).

The following points of interest emphasize the lessons learned throughout the exploratory implementation of M^{3}CT. Development within the M^{3}CT framework is completed. The lessons learned from these facets of implementation should be used to lay the foundation for the next generation of multidisciplinary, multi-fidelity, model-based (i.e., parametric) computational tools.

6.1 Technological Challenges for Extending FSI

6.1.1 Solution Persistence - After the processes have been completed, they must be repeatable. This means that the system must be capable of accessing older versions of those toolsets. Tools will generally be created at a different pace then the computation of solutions. Documentation and understanding of how to leverage these tools must be preserved. Additionally, hardware and software systems available to the network must be capable of running them on a compatible platform in the future.

6.1.2 Reduced Fidelity - It is entirely possible that fidelity can be reduced while maintaining an acceptable error margin. However, there needs be a capability to determine when error is no longer acceptable so that models of greater fidelity can be employed from some other network resource.

6.1.3 Reduced Modeling - Cobbling together disparate reduction methodologies into a single cohesive context reference was weighted upon proprietary interfaces at the individual providers level. This is not the correct placement of this conversion function. The workflow controller must handle these conversions between the output of the previous providers and the input of the next provider.

6.1.4 Context Composition - It is explicit in a multidisciplinary environment that numerous outputs are going to be leveraged by a single input, often recursively, as the system of equations moves toward a solution. It is therefore imperative that a collection of numerous outputs can be effectively mapped to a collection of inputs. Mapping these relationships requires that the outputs and inputs be ordered in a predictable way.

6.1.5 Many to One Mapping - Conversions and integrations of data types will remain a large challenge of collaboration. These conversions can be mitigated through the ability to inject adaptive logic between the outputs and input requirements. These adapters must be capable of accepting multiple outputs from multiple contexts.

6.1.6 Iterative Synchronization - It will be necessary in moving forward with multidisciplinary development to establish criteria for defining mathematical system modules, their methods of internal synchronization, and how they define their exposed contexts for external runtime manipulation.

6.1.7 Module Matching - A more robust approach to filtering providers in the network based on their module expectations is needed. Further, these modules must be able to be assembled back into an accurate representation of their principle system of equations at the lowest possible margin of error.

6.1.8 Unit Matching - A common adaptive solution example would be matching a provider using standard units to a provider using metric units. Another would be changing orientation, rotation, and scale.

6.1.9 Data Aggregation - In all future implementations, the service provider should be able to produce as many context objects as there are reasonable needs for aggregation. Such accuracy is impossible otherwise, or incorrectly requiring an external tool to perform the aggregation.

6.1.10 Simple Interfaces - Context mappings need to be presented to the end user, and adapted by the end user, to match any two providers together that share a common thread of equations. Pushing the results of one provider into the input of another provider clutters the interfaces of providers. M^{3}CT needs to correctly adapt and submit the context through a single provider interface for the targeted provider.

6.2 MVC in FSOOA

Development of M^{3}CT was approached by an MVC design pattern. Each provider was written twice. The first version of each provider was written to wrap the functional side of the provider service. The second version of each provider was written as an extension of the M^{3}CT exertion class. Proprietary configuration files and view implementations were developed for each provider.

The leading reason for each provider to be implemented uniquely is for the very purpose of exploring which method of implementation would produce the best methodology for future implementations. In this, the M^{3}CT product itself is a conglomeration of numerous attempts at defining this ideal relationship.

General MVC design patterns are insufficient for development within an FSOOA environment. The software architecture must be developed through an adaptive and modular interpretation in order to function agnostically with services that will not exist until after the product is developed. This is something that MVC is incapable of doing at its level of responsibility.

A Content Management System (CMS) is an appropriate paradigm for integration into the FSOOA. CMS is generally considered as a pairing of two applications.

- The Content Management Application (CMA) provides a users interface to managing content relative to their permissions.
- The Content Delivery Application (CDA) provides the infrastructural means of delivering aspects of content throughout the network.

M^{3}CT must also manage the context objects being generated for providers in the network. A Component Content Management System (CCMS) is an appropriate paradigm for that service.

Even though these industry terms are closely coupled with websites, their roles are not dependent upon any web services. Without delving deeply into computer science, the CROW solution provided at the end of this Final Report was inspired by CMS design patterns. *SORCER is essentially an advanced implementation of a CDA. M^{3}CT, in its desired, future* *form, is essentially an implementation of a CMA.* Context Libraries used by CROW Query is a simplification of CCMS assembly. Context Libraries use by the Factory of Factories are generating the components of the CMA. These correlations are not direct, but serve as a generalization of cognitive association.

6.3 Community Created Content

The engineering analysis and design community has existed long before and will exist long after many professional careers. Throughout the tenure of AFRL contributors, there will be contributions based on the best practices of their time. However, best practices are sometimes discovered to no longer be ideal, or have lost their trailing documentations on how to utilize their services effectively.

In order to preserve the usability and documentation of these provider services, context libraries can be generated and versioned. With their content being leveraged as the origin of user interfaces, provider connectivity, and service interface wrapping, they satisfy the complexities presented by community created content.

Beyond the technological implementation, treating providers as community created content is an ideal way of identifying and emphasizing efforts within the network. The implementation could go so far as to allow for other users of the network to rate providers, provide comments and criticisms, and leverage each other's services in creating new solutions. The growth of social networks is exponential, and it is this growth pattern that we desire to harness to make fundamental changes in how engineering design is executed.

6.4 Next Steps (CROW)

The culmination of lessons learned throughout the Developers Guide, Users Guide, and this Final Report have provided insight, clarification, and direction towards the next phase of Multi-disciplined, Multi-fidelity, Model-based Computational tools development. The working title for this next generation project is CROW: Conceptualize, Realize, and Optimize for the Warfighter.

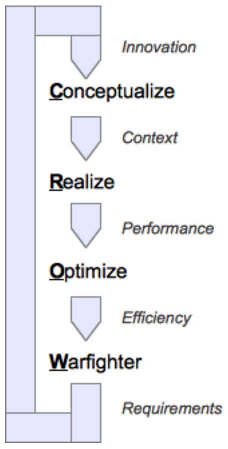


Figure 29. The CROW Workflow

Within the scope of this section, “Provider” represents an independently operating System within a System of Systems (SoS) network environment. Providers are capable of requesting from CROW as well as other Providers, in order to satisfy their input requirements for performing work. In order to become a member of the SoS network environment, a provider must satisfy the providers contract. Once established, the provider is now index-able within the CROW network as a leveraged asset.

6.4.1 CROW First-Meta

The provider’s contract consists mainly of two depths of metadata. The first depth enables the provider to be broadly discovered within the communications network elected for hosting CROW. The second depth enables the dynamic construction of User Interface (UI) Factories and CROW Query (CQ) binding. Second depth metadata is defined in three distinct sections: Input, Progress, and Output.

6.4.1.1 Examples of first-meta: 2D, 3D, Aeroelastic, Rigid, Thermodynamic, Optimization, etc.

Thus, when searching for all Aeroelastic providers in the CROW network, a pool of potential matches can be returned in a relatively fast query process. The CROW workflow will now be able to use all of the selected providers first-meta details when selecting subsequent provider services. Selecting a second parameter, such as 3D, can further refine search results.

6.4.2 CROW Second-Meta

For every provider inserted into the CROW workflow, a simplified GUI allows for one of two behaviors to take place on the Input. Either the user provides a direct resource, of which contains a value. Or the user binds the Input to the Output value of another provider in the workflow. The second-meta provides a range of expectations for each value. When the range is an exact match, and the second-meta is an exact match, the connections can be connected automatically for the user. Otherwise, the user must manually construct connections either directly or through the use of an adapter.

6.4.2.1 Examples of second-meta: worldX, localX, rotationX, chord, thickness, coreTemp, surfaceTemp, ambientTemp, innerLoopCount, outerLoopCount, etc.

6.4.2.2 Examples of data: Strings, Integers, Doubles, Arrays, URL’s, etc.

6.4.2.3 Examples of adapters: Analog to Digital, Kelvin to Celsius to Fahrenheit, 3D to 2D, Standard to Metric, etc.

A known complexity with second-meta comes from the conditional relationship between what is required, and what is required based on a selected option. This is resolved by using appropriate inheritance through the Required, Optional, Conditional (ROC) recursive hierarchy. Second-meta that is marked as conditional, references an optional, and is itself marked as being required or optional with that conditional. These interpretations allow for any number and combination of input and output parameters to be correctly anticipated and initialized by the CROW framework.

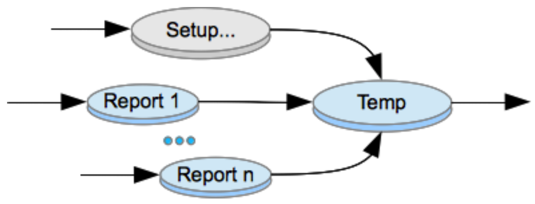


Figure 30. Connecting Outputs to Inputs

6.4.3 CROW Micro-Iteration

Recursion is ultimately what is going to perform the scope of work across a CROW Workflow. This recursion is interpreted in outer looping macro-iteration, and inner looping micro-iteration. Context objects are used to define and control the process of stepping through these iterations.

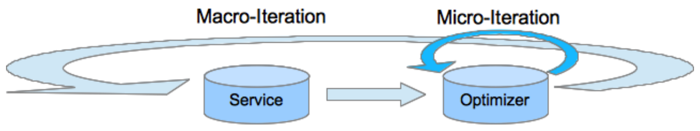


Figure 31. Progress Observation Example

The progress of every provider can be observed through a basic listener implementation. The second-meta of the provider being monitored provides information on how to correctly handle the values being generated. A callback operation is mapped the same way as an input to output operation. This process is triggered at a micro-iteration level.

The progress of a study can likewise be observed through the same implementation as the progress of an individual provider. The scope of concern is raised to the level of which provider in a CROW Workflow is taking action out of a known, or anticipated, sequence of actions. This process is triggered at a macro-iteration level.

6.4.3.1 Examples of progress: Waiting to continue, multi-disciplined combined study-state, percent complete, number of total complete, processing, synchronizing, etc.

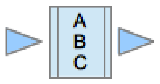


Figure 32. Micro Iteration of Providers A, B, and C - CROW Syntax

The multi-disciplined combined study is a bit tricky. A provider that is capable of working concurrently with one or more providers would use a logic gate such as this to synchronize their inner loop iterations. Progress second-meta can range from non-existent, to extremely complex, dependent upon the interactive nature of the provider. A progress method may go so far as to inject an updated value into the Temp context of the service as an intentional update to its underlying process, and then set its own progress to indicate that it has completed the task. The injected process would listen for this changed state, and use it to continue with its own iteration.

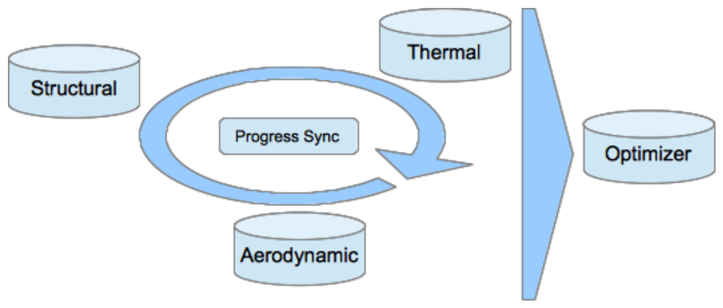


Figure 33: Multi-disciplined Progress Synchronization Example

6.4.3.1.1 Asymmetrical Synchronization

It is noted in the technological challenges for extending FSI that the decomposition and synchronization of complex mathematical systems that the composite determinations could become asymmetrical in how model data is being iterated. This is not necessarily a problem when leveraging components that are both of the original composition. The situation will exist in the case of combining disparate compositions in an attempt to generate new solutions across a multidisciplinary environment. This creates a need to ensure that any future design preempts this plausibility with a built in methodology for handing asymmetrical synchronization.

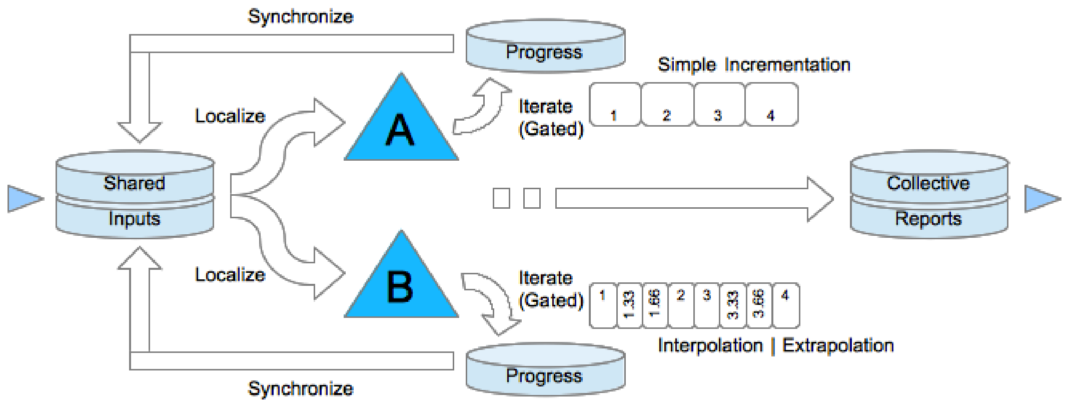


Figure 34. Asymmetrical Synchronization during Micro-Iteration

More details on the approach will come through the actual pursuit and implementation of two micro-iterated systems of equations that are then stitched together. Until that point, this defined method should address these initial concerns.

6.4.4 Chained Optimization

An optimization (Also referred to as Segment-Iteration), is a behavioral analysis of a series of providers. Design variables are adjusted within predefined constraints, as an interpretation of the results of this analysis. These constraints can apply to input values and output values, meaning that the compliance of an optimization may only be able to be determined after completing a step. At the point of non-compliance, an optimizer may consider checking for restored continuity past the edge of failure, or reverting to the previously known validity and adjust the design variables by a different methodology. Gradient-based optimizers will complete as a result of a maxima or minima solution, the result of toggling around a value, the failure of the analysis process, or the result of completing a defined maximum number of attempts on the solution.

Chained optimization (Chained Segment-Iteration), is the behavioral analysis of a series of providers. It is critical that such functionality exists in the CROW architecture. To example the use of chained optimization we can consider the arbitrary case of providers A, B, and C, being optimized by D, where A and B are determined asynchronously, and their results are determined synchronously to C.

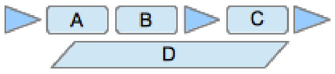
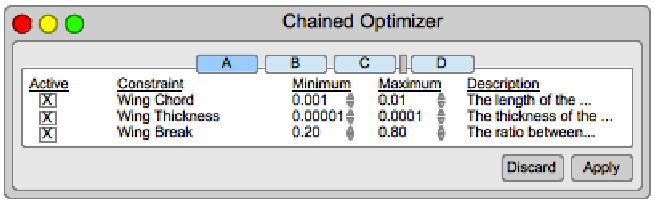


Figure 35. Chained Optimization - CROW Syntax

As a user, the configuration of a chained optimization is relatively simple. Providers A, B, and C are registered to the chained optimization by placing the optimization widget between the left and right WorkerWidget's. When selecting the optimizer for configuration, each provider's library is used to present the user with a UI of design variables and constraints. The Factory of Factories will be leveraged to generate the best possible representation for the varieties of data that can be optimized.



Active	Constraint	Minimum	Maximum	Description
X	Wing Chord	0.001	0.01	The length of the ...
X	Wing Thickness	0.00001	0.0001	The thickness of the ...
X	Wing Break	0.20	0.80	The ratio between...

Figure 36. Chained Optimization - Configuring Constraints

Programmatically, the configuration can be visualized as a series of plugins to the optimization controller object. The agnostic principles of optimization are identical: A series of services are called, evaluated, and the result may be finalized or iterated. This process can be observed through the listener interface, and once the routine is completed, CROW continues with the rest of the workflow.

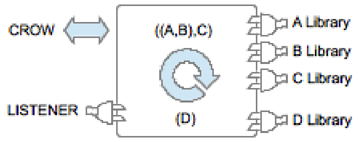


Figure 37. Chained Optimization - Plugged In

6.4.5 CROW Macro-Iteration

Macro-iteration is the process of running the Crow Workflow again, or in a loop. Doing so without introducing any new data, should generate an identical result.

The CROW product can allow for the inputs of any provider to be mapped to the outputs of any other provider from the previous macro-iteration. A primary example of this dimension of analysis could be time step. The results of each loop collectively affect the end result.

Other variable changes, that are guaranteed to be able to be calculated independently, unlike the collective effects of time, would be better served through an implementation of Workflow Concurrency.

6.4.6 Workflow Concurrency

In more advanced examples of use, the user may want to execute multi-dimensional analysis, where a mostly similar processes are conducted across a gradient of a major variable change. In this, multiple crow workflows will be generally similar, however the calculation time, successful completion, and numerous other performances may vary drastically from each other. Of those that are successfully run the user may desire to aggregate their results, make interpretations, and continue with another workflow past that point.

Doing so through a single CROW workflow is impractical, and does not leverage objectification. By treating each CROW workflow as a separate track of logic, relationships can be cloned while data is individually unique. Entirely separate workflow setups can also be run concurrently. In order to bind them all together into one overarching solution, each CROW workflow can be logically placed in series like a tape. However unlike a tape, this is a many to one solution, where a virtually unlimited number of distributed services can be run, and resolved, prior to initiating and evaluating any number of aggregate workflows.

Multi-dimensional analysis introduces a third initialization terminology to the CROW Workflow UI. All three prototypes for use are described next for clarification:

Setup – This information is defined by the user before running the study. This allows for input parameters to be generically satisfied, meaning, they are not the result of a real time calculation, but are instead predetermined by the user creating the study.

Recursive – This information represents the collective outputs of all providers in the Crow Workflow, as their result, from the previous Meta-Loop. This allows for the workflow to iterate over a series of overarching variables, one such example being tau.

Continue – This information is similar to Recursive and Setup, in that this information is used for this Crow Workflow in the capacity of Setup, but the values are the result of other Crow Workflows that have all completed all of their recursions.

Ultimately all three of these UI do the same thing. They provide the starting point for a Crow Workflow Macro-Loop cycle. It is proposed to keep them separately defined, principally in order to preserve a simplified interface and cognitive understanding of functionality.

6.4.7 CROW Mega-Iteration

Mega-Iteration is defined here to satisfy what term to use when the user desires to feed conclusive data back into the initial steps of the CROW Workflow, and perform the entire study again with another overarching level of multidimensional analysis.

Mega-Loop iteration can essentially allow for an evolutionary behavior across the system of systems. Providers at the end of the solution, interpreting the results of the study, can then make educated guesses based off of holistic data. Mutations, alterations, random errors, and other methods of evolution encouragement can be resourced as a way of using artificial intelligence to improve the Warfighter. This level of multi-dimensional analysis is far-reaching and obtainable. Programmatically, it is merely closing the loop between the final results, and the initial setup of the study. Electing how something evolves is by far the bigger question.

6.4.8 CROW Logic Cell

The result of every provider is stored in a full report and retained for historical analysis. Only the second-meta required by the next steps of the CROW workflow are compiled into a sub-report using CROW Query (CQ) syntax. This keeps excess information off of the communications network as well as improving performance. Subsequent workflow will query the provider for a report, specifically, it will only ask for the data that has been mapped as an input to another provider, or is being used by CROW to generate a workflow status report.

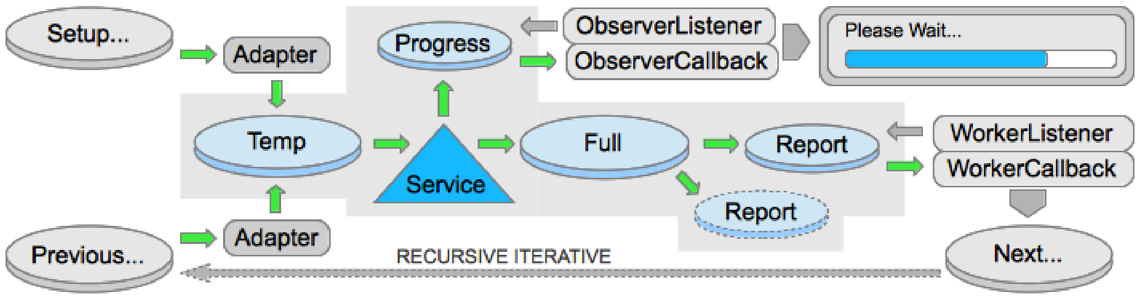


Figure 38. CROW Provider Logic Cell

Every provider in the CROW network is built within the framework of the CROW provider Logic Cell illustrated above. Within that cell, a service may be of any type, concept, or language, so long as the Cell can wrap the service. M^{3}CT development efforts have proven that C++, Fortran, and MatLab, can all be wrapped successfully through methods such as JNA.

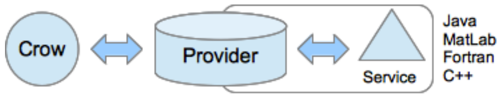


Figure 39. Multiple Source Languages

The number of potential Logic Cells in the CROW network is essentially unlimited. Their design is inspired from an interpretation of a biological neural network; each cell operates independently throughout the SoS, and is capable of communicating in a valid and purposeful manner with other cells based on its individual implementation. Like biological cells, the CROW Provider Logic Cell can be anticipated to grow exponentially. The UI costs into this network are anticipated to reduce synergistically through the use of a Factory of Factories.

6.4.9 CROW Query (CQ)

This functionality is what derives one of two fundamental strengths in the system. CQ will be designed to match Second-Meta outputs with Second-Meta inputs when connecting two or more providers in workflow. By using a simple query language, CQ is nothing more than a patch cable between two or more communication interfaces. The actual values cannot be known during initialization and setup, which makes this abstract method of pairing ideal in this solution and exceedingly easy to learn.

6.5 Key Pairing with CROW Query

If Provider A has an ordered sequence of outputs, and Provider B has an ordered sequence of inputs, then connecting them can be simplified as **A#:B#**

If Provider A has an X, Y, and Z coordinate as separate elements, and Provider B expects an (X,Y,Z) structured input, then this connection can be simplified as **[A#,A#,A#]:B#**

If we reverse the previous condition, we can simply state **A#:[B#,B#,B#]**

If we want to set an absolute value for one of Provider B's inputs, we state **"value":B#**

We can read and write to the workflow configuration by omitting the provider call. Such as **"value":#** or **#:B#**

This simple syntax allows us to work with any complexity of primitive values. Whereas **[A#,B#,C#]:D#** would take the output of three different providers, and assign them to D's input.

CROW Query becomes slightly more complicated when an Adapter is required. Each adapter in the system has a unique name. An adapter that performs Kelvin to Celsius may look like this: **KtoC(A#):B#**

The number of adapters available to the CROW Query syntax is unlimited.

6.5.1 Factory of Factories

This functionality is what derives the second fundamental strength of the system. As new quantifications are discovered, agnostic factories can be generated to handle those quantifications and present them to the user in a constructive way. As with all UIs, having a consistent look and feel to the end user, whether they are using one provider or another, reduces training costs by increasing familiarity and leveraging associative learning patterns.

Using the First-Meta, an unlimited number of providers can be introduced into the network and utilized by CROW without having to specifically program proprietary methods for these new additions. The process of discovery is consistent and intelligent.

Using the Second-Meta, numerous well-defined techniques are available to the solution provider by assigning the appropriate quantification of expected values by range, type, expression, and other methods to be identified. They are then further grouped and related to each other using the ROC recursive hierarchy. Once processed by the Factory, a fully elaborated and cognitive interface is produced to the end user, without the end user having to know anything about the provider beyond context.

6.5.2 Solution Persistence

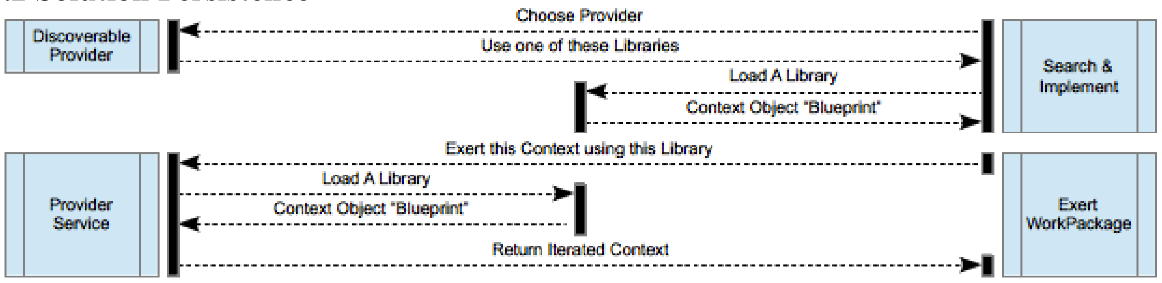


Figure 40. Choosing a Provider Library

Selecting a provider also includes selecting which version of the provider you are going to use. For solution persistence, all providers will maintain a list of libraries to choose from. It is important for users of the CROW network to be able to run their established tests, even after any one of their leveraged providers has updated to a new version. All previous versions are simply marked as deprecated, which will notify the user that an updated solution may be required in order to continue receiving support from said provider.

6.5.3 Providers Contract

Every provider that uses CROW must essentially satisfy the content of this section:

- **First-Meta** allows for the provider to be found in search results, providing relevance to the user.
- **Second-Meta** allows for the interfaces of the provider to be dynamically presented to the user.
- The **Logic Cell** allows for the provider to be implemented consistently throughout the network.
- Solution **Persistence** protects the work efforts of the user by ensuring compatibility stability.

By doing so, they are able to leverage CROW Query (CQ) and the Factory of Factories (FoF).

6.5.4 Implementation Methodology

A multi-disciplined combined study serves as an ideal case study for the implementation methodology of CROW. This particular arbitrary case involves four providers, where three of the providers are progress synchronized prior to their collective result being further evaluated by the fourth provider. For this case study to hold true, it should then be extensible to any number of providers through a fully recursive hierarchy of relationships.

6.5.4.1 Case Study Orientation:

Note: All terms introduced in this section are circumstantial for brevity. All illustrations in this section are rough draft wireframes for concept orientation.

Four CROW Provider Logic Cells (LCs) will represent each of four provider services.

- Three of these LCs will be synchronized through the progress channel.
- Two of these LCs will access the same context within a micro-iteration.
- One of these LC will require an adapter on some of the context values.
- Pseudo Programmatically: JOB(JOB(A, B, C), D);

6.5.5 Populate the Workflow

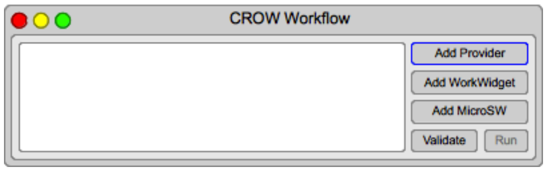


Figure 41. CROW Workflow Controller

Search for and add all providers to the CROW Workflow.

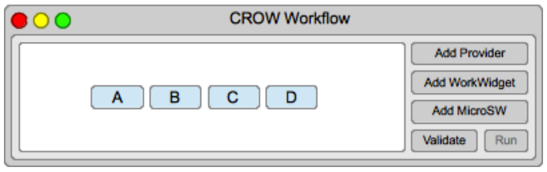


Figure 42. CROW Workflow – Add Providers

Arrange the providers to appear in the order A, B, C, and D.

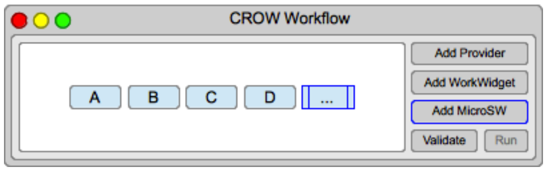


Figure 43. CROW Workflow - Add MicroSW

Add a micro-synchronization widget (MicroSW) to the CROW Workflow.

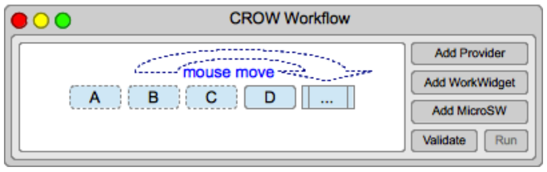


Figure 44. CROW Workflow - Register

Register the MicroSW with A, B, and C by dragging them into the MicroSW. Note that [ABC] is defined in place of A, B, and C.

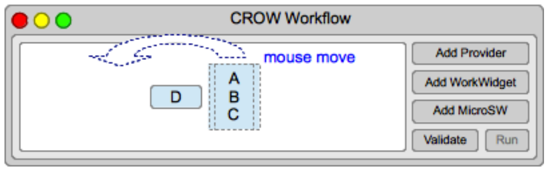


Figure 45. CROW Workflow - Reorganize

You can reorganize elements at any time prior to assigning connections.

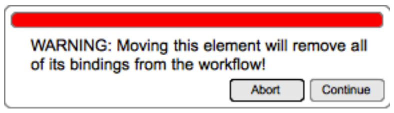

WARNING: Moving this element will remove all of its bindings from the workflow!

Figure 46. CROW Workflow - Moving After Setup Warning

A warning will prompt that changing the assignment will remove the bindings if already set.

Configure the MicroSW

Open the MicroSW context menu and select "Configure".

Specify that A and B are synchronized in parallel.

Specify that [AB] and C are synchronized in serial.

Edit the "onSync" action for each provider in the MicroSW.

Each provider will have specified what its interface is exposing and for which sync gate.

Select which sync gate to active on a specific provider.

Connect the exposed inputs to the exposed outputs of the other providers, or provide a manual value for the sync gate. The ROC hierarchy applies.

Note: Only one provider may set a specific context element during a parallel action.

Note: Providers may have disparate iterations per micro-iteration.

Finish configuring all of the sync gates for all of the providers. Saving often implied.

Select any of the exposed output variables that you want to watch with a Micro-Progress Indicator (MPI) widget. The MPI widget will be updated every sync gate where that variable is exposed.

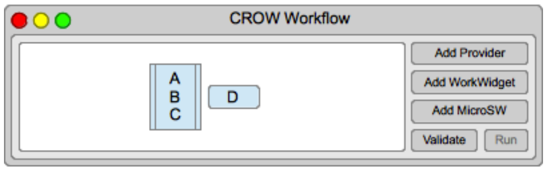


Figure 47. CROW Workflow after MicroSW

Close the MicroSW.

Continue Workflow Setup

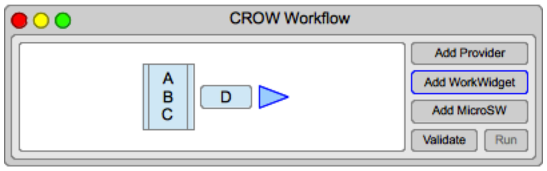


Figure 48. CROW Workflow - Add WorkerWidget

Add a Worker Widget (WW) to the CROW Workflow.

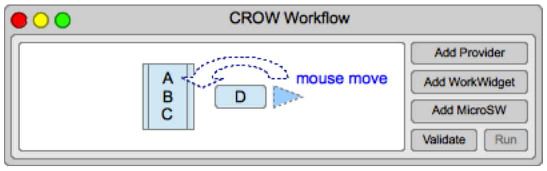


Figure 49. CROW Workflow – Reorganizing WorkerWidgets

Relate the WW with [ABC] and D by dragging the WorkerWidget to the left of D.

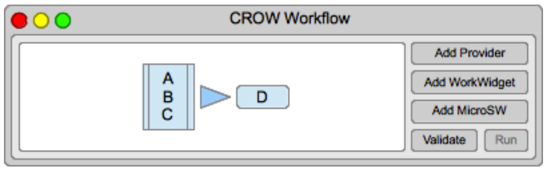
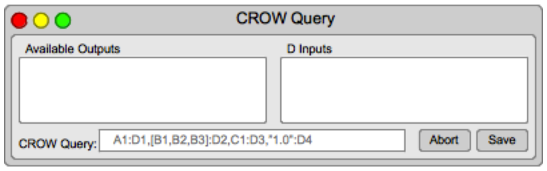


Figure 50. CROW Workflow - WorkerWidget Placed

NOTE: All providers on the left side of the WorkerWidget are automatically identified as eligible output mappings for the right hand provider inputs. Whether or not they are mapped is encouraged by the meta-data of the selected providers, and ultimately the responsibility of the user to define.

Configure the WorkerWidget

Open the WorkerWidget context menu and select “CROW Query”.



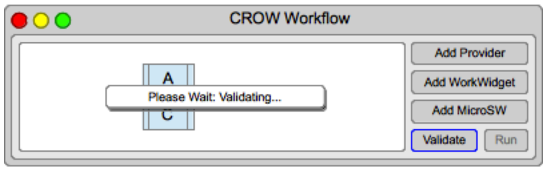
CROW Query: A1:D1,[B1,B2,B3]:D2,C1:D3,"1.0":D4

Figure 51. CROW Query Controller

Connect the exposed inputs of D to the exposed outputs of [ABC], or provide a manual value to the remaining inputs. The ROC hierarchy applies.

Close the WorkerWidget.

Validate the Workflow



Please Wait: Validating...

Figure 52. CROW Workflow - Validating

Validate the CROW Workflow by clicking the Validate button.

Resolve all ROC violations by setting values manually in the Workflow Setup, or editing the MicroSW or WW elements.

The implementation is now ready to execute.

Optional: The user may elect to connect exposed D outputs with exposed [ABC] inputs in succession to there starting values. Setting the macro-iteration count to a positive value will apply macro-iteration. These recursive assignments are ignored when macro-iteration is set to zero.

6.5.6 Trailblazer

As another benefit of participating in CROW, Trailblazer can discover provider connectivity. As the CROW network increases in acceptance and capability, the ability becomes apparent in that two completely incompatible providers can find a way to be linked together through the use of First-Meta and Second-Meta inputs and outputs. A product like Trailblazer would not be possible or realistically feasible to start development until the CROW network has increased in successful participation. Ideally Trailblazer will make the use of Adapters easier and more readily automated while creating CROW workflow.

6.5.7 BAR

The Goal of the BAR approach is to alleviate both the technical and the non-technical challenges of CROW. Provider Buy-in, Awards for contribution, and Recognition, are critical attributes for multidisciplinary action and cultural change amongst analysts accustomed to narrower transition strategies. The long-term benefits of CROW are competing with the short-term benefits of single faceted solutions. This practice also promotes efficient spending across participating sectors, by providing a built in mechanic for observing usage. Leveraging management tools directly into the solution is plausible as described below, bringing large scalability to the software architecture.

6.5.8 CROW Summary

The solution variations in micro-looping, chained optimization (chained segment-looping), macro-looping, workflow concurrency, and mega-looping gives the end user the greatest potential of performance in executing a study along with the greatest measure of flexibility across independently published providers.

While the use cases for CROW can range from simple to extraordinarily complicated, the first-meta and second-meta standards will allow for the exponentially growing provider base to remain flexible, dynamic, and usable, while also encouraging provider participation toward the defined standards.

In short, to address future engineering design challenges, an architecture is required that can adapt to growth driven by complexity. This adaptation has technical and cultural facets.

Developing a comprehensive User Interface that provides a feeling of product familiarity across services already developed, and those to be developed in the future, reduces user error, user frustration, and exponentially reduces the costs of creating a unique interface per provider. Any truly unique needs of an interface can be abstracted, modularized, and added to the library of resources for that provider, and all providers that follow afterward.

It is absolutely critical to the persistence of discovered solutions that versions of providers are properly bound and preserved. Any uncontrolled change within a system of systems across a broad federation of services is a critical failure. Such changes can lead to false positives, false negatives, and potentially lost decades of accumulated full-time equivalents through misdirection or duplication in research and development.

In the pursuit of multi-dimensional, multidisciplinary, multi-fidelity, model-based solutions (“M^4 Solutions” it could be coined); CROW represents a starting point for developing the architecture of that pursuit.

7.0 SUMMARY

OBJECTIVE II

This summary section is part of three such summaries related to the final report. This summary is focused on overarching aspects. Specifically the lessons learned in providing a solution for a multi-disciplined product specialized in Flapping Sciences Integration (FSI) for Micro Air Vehicles (MAV).

It is notable that the underlying methodologies of the M^3CT effort can be extended to all multi-disciplined efforts. The development of an M^3CT beta-quality product would be required to realize that potential. The user interface should be considered alpha-quality and archived for reference.

Lessons Learned

Model implementations across all disciplines need to be defined in a consistent and extensible way. Naming conventions, the order of how common terms are defined, and how common terms are stored in a context object are all critical requirements. Establishing these conventions will help alleviate the costs and user errors associated with establishing modularized FSI solutions. These model implementations play a major role in calculation time lost resolving interface differences.

There are numerous technological challenges in implementing FSI solutions. The tradeoffs between calculation time, accuracy, and the dimensions of analysis all contribute to the overall productivity of the study. False positives, false negatives, unacceptable error margins, and the cost of resources all play a role in deciding how to break processes down, and whether or not such a task is ultimately possible.

The implementation architecture approached for this solution is insufficient. A community created content paradigm can be leveraged for methods on handling such a diverse background of providers. CROW has been proposed as a step towards this paradigm. The principle architecture selected for the underlying use of M^3CT has evolved to iGrid12 during the course of this study. It remains as a plausible solution at the middle tier of implementation.

8.0 PUBLICATIONS AND PRESENTATIONS

Journal Papers:

Oppenheimer, M.W., Doman, D.B., Sigthorsson, D.O., "Dynamics and Control of a Biomimetic Vehicle Using Biased Wingbeat Forcing Functions," AIAA Journal of Guidance, Control, and Dynamics, 2011, Vol.34: 204-217

Doman, D., Oppenheimer, M.W., Sigthorsson, D.O., "Wingbeat Shape Modulation for Flapping-Wing Micro-Air-Vehicle Control During Hover," AIAA Journal of Guidance Control And Dynamics, 2010, AIAA-0731-5090 vol.33 no.3 (724-739).

Book Chapters:

Doman, D.B., Oppenheimer, M.W., Sigthorsson, D.O., "Dynamics and Control of Flapping Wing MAVs," in "Handbook of Unmanned Aerial Vehicles," Edited by K. P. Valavanis and G. J. Vachtsevanos, Springer, 2013.

Technical Reports:

Hall, A.J., Roberts, A., Weintraub, I., Riddick, J.C., "Flapping Wing Technology for Micro Air Vehicles Incorporating a Lead Zirconate Titanate (PZT) Bimorph Actuator", Army Research Laboratory ARL-TR-6040, June 2012.

Conference Papers:

Sigthorsson, D.O., Oppenheimer, M.W., Doman, D.B., "Wingbeat Synchronization And Implementation," AIAA Guidance, Navigation, and Control Conference, 2012.

Oppenheimer, D.O., Awtar, S., Sigthorsson, D.O., Weintraub, I., Doman, D.B., "Computation of Inertial Forces and Torques Associated with Flapping Wings," AIAA Guidance, Navigation, and Control Conference, 2012.

Sigthorsson, D.O., Oppenheimer, M.W., Doman, D.B., "Insect Sized Flapping Wing Vehicles Versus Rotorcrafts, a Comparative Study," AIAA Aerospace Sciences Meeting, 2012.

Sigthorsson, D.O., Oppenheimer, M.W., Doman, D.B., "Flapping Wing Micro Air Vehicle Aerodynamic Modeling Including Flapping and Rigid Body Velocity," AIAA Aerospace Sciences Meeting, 2012.

Oppenheimer, M.W., Sigthorsson, D.O., Doman, D.B., “A Revised Blade Element Model for Vehicles with Flapping Wings,” AIAA Aerospace Sciences Meeting, 2012.

Sigthorsson, D.O., Oppenheimer, M.W., Doman, D.B., “Body Torque Generation for a Flapping Wing Micro Air Vehicle by Angle of Attack Change,” AIAA Aerospace Sciences Meeting, 2011.

Oppenheimer, M.W., Sigthorsson, D.O., Doman,D.B., “Controllability Analysis For A Flapping-Wing MAV With Power And Control Actuators,” The American Control Conference, 2010.

Sigthorsson, D.O., Oppenheimer, M.W., Doman,D.B., “Flapping Wing Micro-Air-Vehicle Control, Employing Triangular Wave Strokes and Cycle-Averaging,” AIAA Guidance, Navigation, and Control Conference, 2010.

Sigthorsson, D.O., Oppenheimer, M.W., Doman, D.B., “Flapping Wing Micro-Air-Vehicle 4-DOF Controller Applied to a 6-DOF Model,” AIAA Guidance, Navigation, and Control Conference, 2010.

Oppenheimer, M.W., Doman, D.B., Sigthorsson, D.O., “Dynamics and Control of a Biomimetic Vehicle: Part I - Aerodynamic Model with Wing Bias,” AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace, AIAA-2010- 1023, 2010.

Doman, D.B., Oppenheimer, M.W., Sigthorsson, D.O., “Dynamics and Control of a Biomimetic Vehicle: Part II - Control with Wing Bias,” AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace, AIAA-2010- 1024, 2010.

Other 2010 GNC References:

Oppenheimer, M.W., Bolender, M., Sigthorsson, D.O., Doman, D.B., “Analysis of the Translational Motion of a Flapping Wing Micro Air Vehicle”, AIAA-Paper-2010-7708, Guidance, Navigation, and Control Conference, 2010.

Oppenheimer, M.W., Sigthorsson, D.O., Doman, D.B., “Multiple Degree-of-Freedom Control of a Flapping-Wing Micro Air Vehicle with Power and Control Actuators”, AIAA-Paper-2010-7557, Guidance, Navigation, and Control Conference, 2010.

All conference papers were presented at the respective conference.

REFERENCES

[i] Stanford, B., Beran, P. S., Snyder, R., and Patil, M., “Stability and Power Optimality in Time-Periodic Flapping Wing Structures,” 53rd Structures, Structural Dynamics, and Materials Conference, 23 - 26 April 2012, Honolulu, Hawaii, Aug. 2011, pp. 1–13.
Figure 1
[i]i Bhatia, M., Patil, M., Woolsey, C., Stanford, B., and Beran, P., “LQR Controller for Stabilization of Flapping Wing MAV in Gust Environments,” *AIAA: submitted for publication.”*

[i]ii Berman, G., Wang, Z., “Energy-Minimizing Kinematics in Hovering Insect Flight,” *Journal of Fluid Mechanics*, Vol. 582, pp. 153-168, 2007.

[iv] Bryson, D.E., Stanford, B.K., McClung, A.M., Sims, T.W., Miranda, J.L., Beran, P.S., and Parker, G.H., “Multidisciplinary Optimization of a Hovering Wing with a Service-Oriented Framework and Experimental Validation,” AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition Conference, Orlando, FL, January 4-7, 2011.

[v]Clayman, G., Beran, P., Bhatia, M., and Stanford, B., “Service-Oriented Multidisciplinary Optimization of a Closed Loop Flapping Wing Subject to Gust,” AIAA Journal (in review), 2012.

[vi]Svanberg, K., “The Method of Moving Asymptotes – A New Method for Structural Optimization,” *International Journal for Numerical Methods in Engineering*, Vol. 24, No. 2, pp. 359-373, 1987.

[vii] Snyder, Richard D., Beran, Philip S., Huttsell, Lawrence J., and Pettit, Chris L. “Aeroelasticity Research in the Multidisciplinary Technologies Center of the United States Air Force Research Laboratory” International Forum on Aeroelasticity and Structural Dynamics, Madrid, Spain, Jun 200[1].

LIST OF SYMBOLS, ABBREVIATIONS AND ACRONYMS

Symbol	Definition
CAS	Cyclic Analytical Sensitivity
ROM	Reduced Order Model
CSD	Computational Structural Dynamics
FSOOA	Federated Services Object Oriented Architecture
SORCER	Service ORiented Computing EnviRonment
M^3CT	Multidisciplinary Multifidelity Model-based Computational Tool
QS	Quasi-Steady
QSCSD	QS + CSD, A provider in the network.
ANT	Another Neat Tool
IDE	Interactive Development Environment
NURBS	Non-Uniform Rational B-Spline
MMA	Method of Moving Asymptotes
DOT	Design Optimization Tools
MUAV	Micro Unmanned Air Vehicle
AE	Aero-Elastics
FSI	Flapping Sciences integration
MAV	Micro Air Vehicle
SOA	Service Oriented Architecture
EO	Exertion Oriented
UUID	Universally Unique IDentifier