Reasoning Service Exemplars for NASA's Data and Reasoning Fabric

Stefan Schuet* Joshua Baculi† Vaishali Hosagrahara‡
Vahram Stepanyan§ Katelyn J. Jarvis¶
Kalmanje S. Krishnakumar‖

NASA Ames Research Center, Moffett Field, CA 94035

Future operations for Urban and Advanced Air Mobility are enabled by a distributed network of reliable and secured data and reasoning services referred to here as a fabric. In the aggregate, such a system must be all encompassing and mission agnostic, but specific use-cases are still needed to improve understanding and drive design paradigms. For this purpose, three reasoning service exemplars for Target Selection and Routing, Trajectory Generation, and Battery Health Management were developed and integrated into a specific NASA proposed data and reasoning fabric. These services were then used to build a mission reasoning application for lightning strike reconnaissance developed in collaboration with the Civil Air Patrol. Autonomous mission execution was then demonstrated using a multi-vehicle simulation platform with a full envelope 6 Degree-of-Freedom dynamics model for a concept electric Vertical Takeoff and Landing aircraft.

Nomenclature

AAM	Advanced Air Mobility
CAP	Civil Air Patrol
DRF	Data and Reasoning Fabric
DMD	Dynamic Mode Decomposition
DOF	Degrees of Freedom
eVTOL	electric Vertical Takeoff and Landing
FlightCODE	Flight dynamics and control modeling tool for COnceptual DEsign
JSON	JavaScript Object Notation
MPC	Model Predictive Control
TOP	Team Orienteering Problem
TSP	Traveling Salesman Problem

I. Introduction

The path to safely realizing Advanced Air Mobility (AAM) is fraught with challenges. These include piloted and autonomous vehicle interaction, steep increases in density of operations and vehicle types, wind disturbances in urban canyons, dynamic obstacle perception and avoidance, multi-vehicle rerouting for emergency services, and complex interactions between the companies, cities, and public that seeks to benefit. A trusted and secured marketplace capable of supporting all AAM stakeholders is needed to catalyze the development of such a system. To this end, NASA has previously proposed a Data and Reasoning Fabric (DRF)

*Computer Engineer, Intelligent Systems Division, AIAA Member, stefan.r.schuet@nasa.gov.
†Systems Engineer, HX5, LLC., Intelligent Systems Division, AIAA Member.
‡Software Engineer, KBR Wyle Services, Intelligent Systems Division.
§ESOF40-Technical Advisor, KBR Wyle Services, Intelligent Systems Division, AIAA Member.
¶Applied Mathematician, Intelligent Systems Division.
‖Associate Division Chief, Intelligent Systems Division, AIAA Associate Fellow.

with a mission agnostic foundation for developing decentralized AAM related services that are safe, resilient, and secure.[1–3] Furthermore, these services need to support both human-to-machine and machine-to-machine interfaces as well as deployment on distributed computing architectures that blend the spectrum of resources from cloud (large data center) to extreme edge (end device) compute nodes.[3,4]

In order to explore the application of NASA's DRF in a simplified but concrete form, a lightning strike reconnaissance use-case was developed in collaboration with the Civil Air Patrol (CAP). In the use-case scenario, a human operator selects a mission geographical region and downloads lightning strike location data from a DRF data service over a time-window of interest. The data is loaded into a graphical interface as shown in Fig. 1. The operator or automated agent, then selects the appropriate vehicles to service the mission, prioritizes the lightning site locations, and (optionally) enters the survey-time needed to complete reconnaissance at each location.

Figure 1: Mission View application displaying launch location and target lightning sites to visit with limited air-vehicle resources. The problem is to autonomously select the highest priority subset of sites, to plan a flyable vehicle trajectory to visit them with limited control authority, and to estimate remaining battery charge.

To build on extensive rotorcraft modeling work for AAM applications, a 1200-lb payload, quad-rotor, electric Vertical Takeoff and Landing (eVTOL) vehicle was selected from a collection of available concept vehicle models[5] for use in this study, and is shown in Fig. 2. For this design, a full-envelope six degree-of-freedom simulation model was established using the FlightCODE toolchain (formerly, SIMPLI-FLYD), jointly developed by NASA and the U.S. Army,[6,7] and augmented with a basic electric power train model.[8] The resulting stitched[9] open-loop model uses four independent voltage inputs to drive a rotor-speed controlled dynamics model with states for position, altitude, Euler angles, rotor-rate, and body-frame translational and rotational rates; a total of 4 inputs and 16 states in all.

The objective of the mission is to survey the highest priority lightning sites, while staying within operational constraints. Reasoning to achieve the mission objective requires (1) determining the subset of high priority points that are reachable with a given number of vehicles, (2) the order or route to take through them, (3) the specific trajectory to fly and the required control inputs that stay within the vehicle constraints,

Figure 2: Selected 1200-lb payload quad eVTOL concept vehicle.

and (4) the predicted battery charge consumed by each vehicle as it moves through its trajectory. Steps (1) and (2) are accomplished by a target selection and routing service, (3) is performed by a flyable trajectory service, and (4) is completed by a battery health management service. Each of these services are developed with the potential to support generic AAM operations, but are assembled here to serve the lightning strike reconnaissance mission in particular. The overall system with distributed service architecture is shown in Fig. 3.

This paper summarizes the development and integration of these services. Autonomous execution is demonstrated on a 6-degree-of-freedom multi-vehicle simulation platform. The primary objective is to show how a distributed reasoning framework can establish a sustainable and faster implementation path for higher levels of autonomy.

II. Services

The services discussed in this section were all developed in Python using existing open-source packages. Web-service capability was implemented using the Flask web framework,[10] deployed on Amazon Web Service cloud infrastructure[11] and registered with NASA's DRF core software.[3] Both human-to-machine and machine-to-machine interactions are needed. Human interaction is enabled through a web-deployable Mission View application that was developed on the Unity platform[12] to collect human-user input and present results for approval. An example display from this application was shown in Fig. 1. In the following, each of the DRF services needed for the lightning reconnaissance use-case are explained with additional implementation details.

A. Target Selection and Routing

Let the fleet consist of n similar air vehicles $v_i,\ i = 1, \ldots, n$, with maximum flight time of $T_{\max}$ and average speed of V_{av} at given atmospheric conditions. The air vehicles are required to survey a set of lightning sites $S = \{s_i,\ i = 1, \ldots, N\}$ in a geographic area of interest, where each site is defined by its centroid c_i and area A_i for $i = 1, \ldots, N$. Each air vehicle starts its flight from a base B and returns to the base within a time window less than $T_{\max}$. We associate a survey time sT_i with each lightning site s_i, which is proportional to site area A_i. Since not all lightning sites may be reachable from the selected base B, we down-select to the set of reachable sites defined as

$$S_r = \{s_i \in S \mid \|c_i - B\|/V_{\text{av}} + sT_i \leq T_{\max}, \text{for } i = 1, \ldots, N\}.$$

A priority score p_i is then assigned to each reachable lightning site s_i based on the severity and fire danger. The base B is concatenated at the beginning and end of the list of reachable points, so that the final list becomes $R = \{B, c_i \text{ for each } s_i \in S_r, B\}$. The objective is to determine a set of sites and a route for each vehicle, limited by $T_{\max}$, such that the total priority score for the entire fleet is maximized. To this end, we compute the pairwise distances

$$d_{ij} = \|r_i - r_j\|, \text{ for all } r_{i,j} \in R,$$

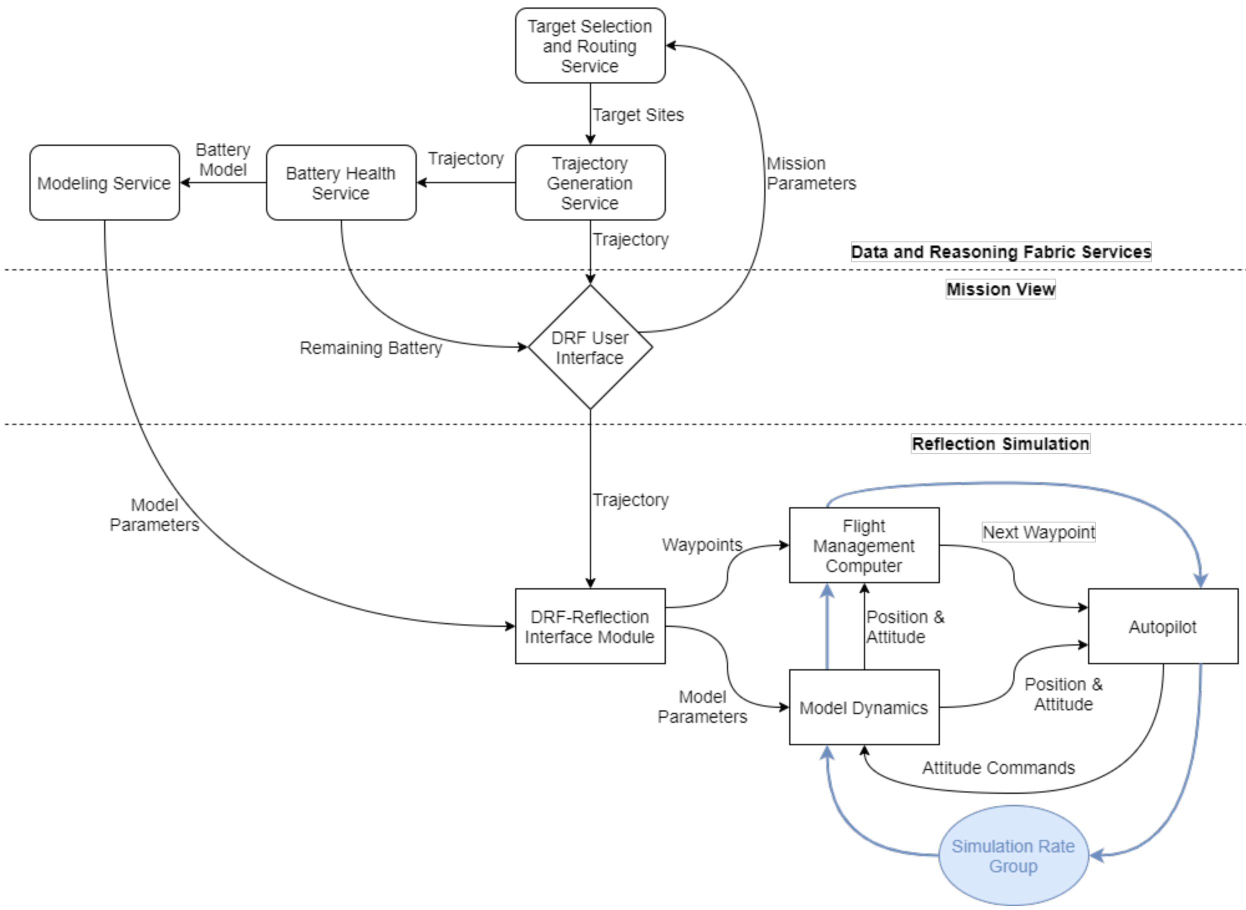


Figure 3: DRF service architecture for the lightning-strike reconnaissance use-case. **Data and Reasoning Fabric** services execute from the cloud or edge-compute nodes, the **Mission View** interface collects human operator mission configuration parameters and passes them to DRF services, and the in-house **Reflection** framework provides nonlinear 6DOF multi-vehicle simulation with autopilot capability for tracking a given trajectory.

between points in the reachable set R and the corresponding times of flight $t_{ij} = d_{ij}/V_{\text{av}}$ based on the average speed.

The required computation is formulated as a team orienteering problem (TOP).[13] It is a mixed-integer linear program for the following decision variables: binary variables $x_{ijk} = 1$ if a visit to site i is followed by a visit to site j for air vehicle k and $x_{ijk} = 0$ otherwise; binary variables $y_{ik} = 1$ if site i visited by air vehicle k and $y_{ik} = 0$ otherwise; integer variables u_{ik} are used to prevent sub-tours; $i, j = 1, \ldots, m$ and $k = 1, \ldots, n$. The optimization problem is expressed as:

$$\text{maximize} \quad f = \sum_{i=2}^{m-1} \sum_{k=1}^{n} p_i y_{ik} \tag{1}$$

$$\text{subject to:} \quad \sum_{j=2}^{m} \sum_{k=1}^{n} x_{1jk} = \sum_{i=1}^{m-1} \sum_{k=1}^{n} x_{imk} = n \tag{2}$$

$$\sum_{k=1}^{n} y_{ik} \leq 1, \;\; i = 2, \ldots, m-1 \tag{3}$$

$$\sum_{j=2}^{m-1} x_{jik} = \sum_{j=2}^{m} x_{ijk} = y_{ik}, \;\; i = 2, \ldots, m-1, k = 1, \ldots, n \tag{4}$$

$$\sum_{i=1}^{m-1} \left(sT_k y_{ik} + \sum_{j=2}^{m} t_{ij} x_{ijk} \right) \leq T_{\max}, \;\; k = 1, \ldots, n \tag{5}$$

$$2 \leq u_{ik} \leq N, \;\; i = 2, \ldots, m, k = 1, \ldots, n \tag{6}$$

$$u_{ik} - u_{jk} + 1 \leq (m-1)(1 - x_{ijk}), \;\; i, j = 1, \ldots, m, k = 1, \ldots, n\,, \tag{7}$$

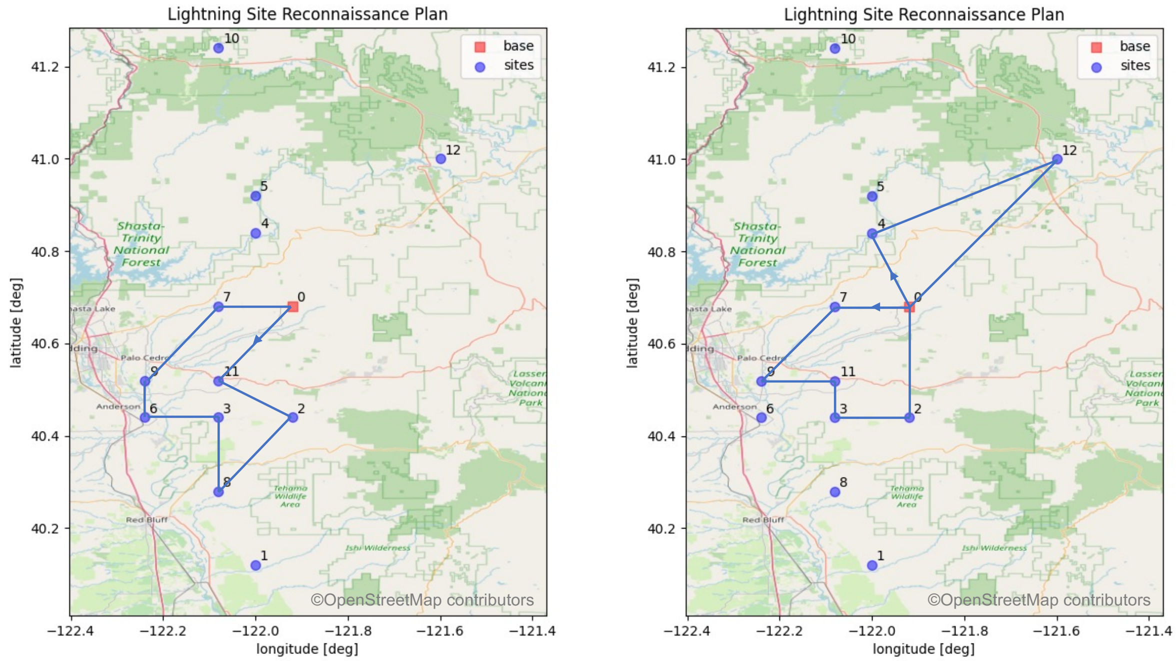
where constraints (2) guarantee that each route starts and ends at the base, constraints (3) ensure that every site is visited at most once, constraints (4) guarantee that, if a site is visited in a given route, it is preceded and followed by exactly one other visit in the same route, constraints (5) ensure the limited time budget for each route, and constraints (6) and (7) are necessary to prevent sub-tours according to Miller–Tucker–Zemlin (MTZ) formulation for the traveling salesman problem (TSP).[14] The solution to the above team orienteering problem maximizes priority given a time constraint. This is different than minimizing the round trip time, as in a standard TSP, and a consequence is that the TOP route may not be the minimum time route to visit the selected sites. The Python Mixed Integer Programming package was used to solve the examples presented in this paper.[15] A couple of solution examples are presented Fig. 4 for the selected quad concept eVTOL rotorcraft operating at 60 knots.

B. Flyable Trajectory Generation

The trajectory generation service is developed around a Model Predictive Control (MPC) architecture that maneuvers a vehicle math model through a series of coarse way-points subject to constraints. This is accomplished by solving a convex optimization problem:

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{i=0}^{N-1} f(u(i)) + g(x(i+1)) \\ \text{subject to} & x(k+1) = A(k)x(k) + B(k)u(k) \\ & |u(k)| \leq 1 \text{ for } k = 0, 1, \ldots, N-1, \\ & |x(k)| \leq 1 \text{ for } k = 1, \ldots, N, \\ & x(t_j) = w(j), \text{ for } j = 1, \ldots, N_{\text{wp}}, \end{array} \tag{8}$$

where $f(u)$ is a convex function of the input that measures control usage, e.g., fuel or battery charge, and $g(x)$ is a convex function that provides regularization of the system state for controlling trajectory smoothness. The problem variables are $x(k) \in \mathbb{R}^n$ for $k = 1, \ldots, N$ and $u(k) \in \mathbb{R}^m$ for $k = 0, \ldots, N-1$. As a matter of good practice, the state and input variables are *normalized* relative to their maximum values. The normalized and discretized vehicle math model $A(k)$, $B(k)$, initial state $x(0)$, and list of target way-points $w(j)$ and waypoint arrival times t_j for $j = 1, \ldots, N_{\text{wp}}$ are given. The vehicle math model is open-loop



(a) One vehicle solution.

(b) Two vehicle solution.

Figure 4: Routing problem solution examples. [Left] One vehicle routing with 75 minute flight-time, and fly-through (no service time) for equal priority sites. [Right] Two vehicle fleet solution with 60 minute flight-time, randomized priorities $p = (10, 9, 10, 1, 7, 10, 8, 2, 7, 12)$, service times $sT = (45, 57, 35, 38, 50, 39, 57, 47, 47, 31)$ seconds, and sites 1 and 10 unreachable.

so that the overall formulation is control system agnostic, i.e., the solution to (8) includes a feasible set of approximated control inputs to the open-loop system. A flyable trajectory solution to this problem for a collection of lightning strikes reachable within a 75 minute operating time is shown in Fig. 5. For this example, a discetized linear vehicle model was used and convex quadratic functions were selected for $f(u)$ and $g(x)$. With these assumptions the problem formulation (8) becomes a convex Quadratic Program.

With a given trajectory solution to (8), trajectory performance robustness due to external disturbance is provided by solving the following robust trajectory optimization problem

$$
\begin{array}{ll}
\text{minimize} & \gamma_{\mathrm{h}} r_{\mathrm{h}} + \gamma_{\mathrm{pos}} r_{\mathrm{pos}} + \gamma_{\mathrm{hdg}} r_{\mathrm{hdg}} \\
\text{subject to} & \displaystyle\sum_{i=0}^{N-1} f(u(i)) \leq \rho \cdot \text{Nominal Input Cost} \\
& r_{\mathrm{pos}} \leq r_{\mathrm{pos}}^{\mathrm{req}}, \quad r_{\mathrm{h}} \leq r_{\mathrm{h}}^{\mathrm{req}}, \quad r_{\mathrm{hdg}} \leq r_{\mathrm{hdg}}^{\mathrm{req}} \\
& \|(x_{\mathrm{xe}}(k), x_{\mathrm{ye}}(k)) - (w_{\mathrm{xe}}(k), w_{\mathrm{ye}}(k))\|_2 \leq r_{\mathrm{pos}} \;\text{ for } k = 1, ..., N, \\
& |x_{\mathrm{h}}(k) - w_{\mathrm{h}}(k)| \leq r_{\mathrm{h}} \;\text{ for } k = 1, ..., N, \\
& |x_{\mathrm{hdg}}(k) - w_{\mathrm{hdg}}(k)| \leq r_{\mathrm{hdg}} \;\text{ for } k = 1, ..., N, \\
& x(k+1) = A(k)x(k) + B(k)u(k) + d(k) \;\text{ for } k = 0, ..., N-1, \\
& |x(k)| \leq 1 \;\text{ for } k = 1, ..., N, \\
& |u(k)| \leq 1 \;\text{ for } k = 0, ..., N-1.
\end{array}
\tag{9}
$$

Here, the problem variables r_{h}, r_{pos}, and r_{hdg} are achievable performance bounds relative to the planned trajectory for height, position, and heading, respectively. The Nominal Input Cost is the total input or control use from the nominal problem (8), and ρ is a positive constant (typically ≥ 1) that limits the net control use to fly the trajectory within performance bounds with disturbances. The points $(x_{\mathrm{xe}}(k), x_{\mathrm{ye}}(k))$

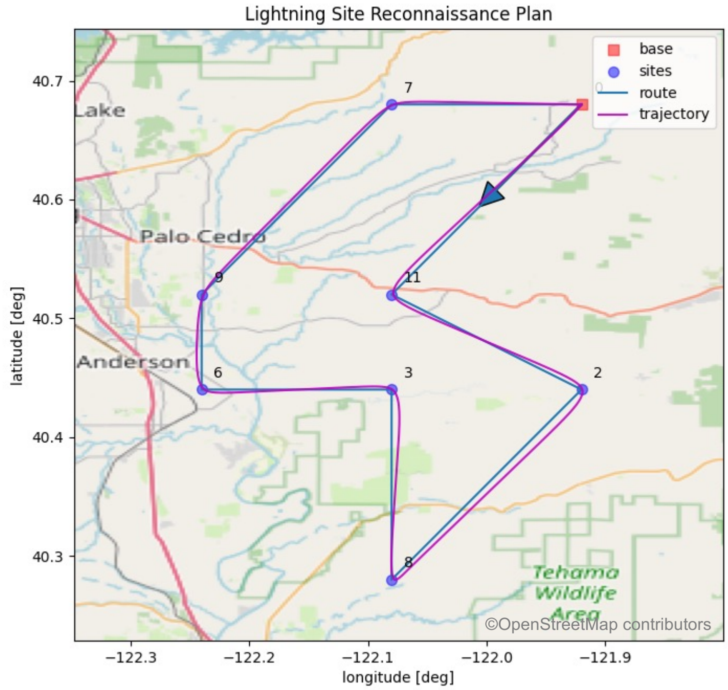


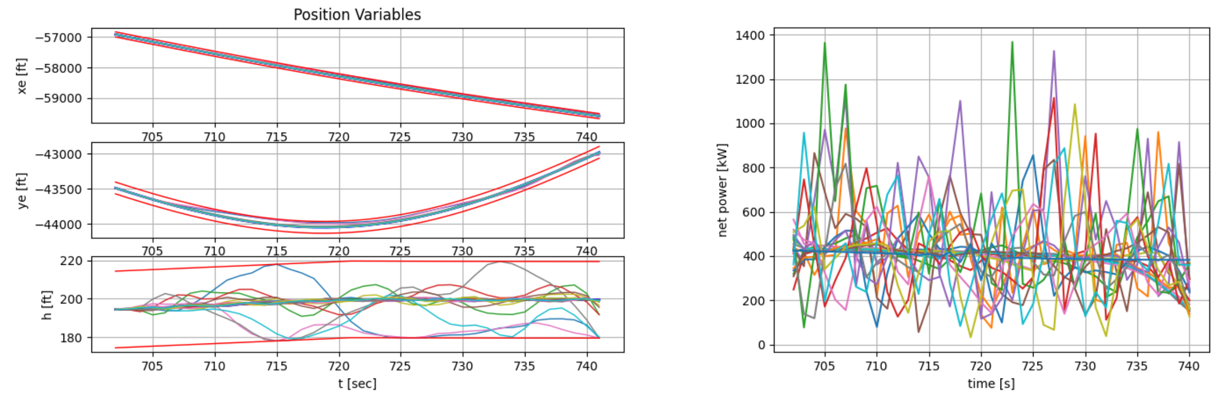
Figure 5: Trajectory solution example for a prioritized list of waypoints reachable within 75 minutes. A discretized approximation to a 6-DOF dynamics model is used to ensure a smooth flyable trajectory with constraints on state and control inputs. For the selected eVTOL vehicle, the trajectory output includes the voltage and current required to fly the trajectory, which is passed to the battery health service for planning battery usage.

are the position coordinates, $x_{\mathrm{h}}(k)$ is the height, and $x_{\mathrm{hdg}}(k)$ is the heading, all in a local earth frame. The given problem data includes the required performance bounds $r_{\mathrm{pos}}^{\mathrm{req}}$, $r_{\mathrm{h}}^{\mathrm{req}}$ $r_{\mathrm{hdg}}^{\mathrm{req}}$, for position, height and heading, respectively, a dense set of waypoint states $w(k)$ loaded from the nominal trajectory and disturbance input $d(k)$ for $k = 0, \ldots, N-1$. The three performance weights γ_{h}, γ_{pos}, and γ_{hdg} provide prioritization between height, position, and heading performance, respectively. The robust problem solution determines a feasible normalized trajectory $x(k)$, control input $u(k)$, and performance bounds r_{h}, r_{pos}, and r_{hdg} subject to the constraint on total control use, i.e., $\rho \cdot$ (Nominal Input Cost). When $f(u)$ is convex quadratic, or convex piecewise linear, the robust optimization problem (9) is a convex Second Order Cone Program due to the position norm constraint.

Example solutions to the robust trajectory planning problem (9) are plotted in Fig. 6. Each of the 20 trajectories plotted is a solution to (9) for a 40 second flight segment through waypoint 11 in Fig. 5, with moderate turbulence simulated from a low-altitude Dryden model. The performance bounds obtained were $r_{\mathrm{pos}} = 83.68$ [ft], $r_{\mathrm{h}} = 20.00$ [ft], and $r_{\mathrm{hdg}} = 4.31$ [deg], with settings $\gamma_{\mathrm{h}} = \gamma_{\mathrm{pos}} = \gamma_{\mathrm{hdg}} = 1$ and $r_{\mathrm{pos}}^{\mathrm{req}} = 100$ [ft], $r_{\mathrm{h}}^{\mathrm{req}} = 20$ [ft] and $r_{\mathrm{hdg}}^{\mathrm{req}} = 10$ [deg]. Fig. 6b shows large amplitude swings in power for disturbance rejection. These are an expected by-product of the rotor-speed control system associated with the chosen concept vehicle.[16]

While there are many interesting variations to both (8) and (9), a good design practice is to keep the cost and constraint functions convex (including linear) if possible. This is because convex optimization problems can be solved reliably and globally, often with existing algorithms and software. Furthermore, it is usually possible to establish polynomial bounds on the compute time, and to numerically prove in-feasibility of particular user requirements.[17] Efficient algorithms for solving MPC problems have been investigated and software is readily available.[18–21] We used the CVXPY Python software package to solve both the nominal (8) and robust (9) problems above.[22,23]

Maintaining the convexity of the optimization problem often requires thoughtful choice of problem variables, and usually strategies for breaking a non-convex formulation down into a series of convex stages.



(a) Trajectory performance.

(b) Power required.

Figure 6: Robust trajectory generation with Monte-Carlo Dryden turbulence.

A common approach is through the use of an iterative series of convex approximations, a process referred to as sequential convex programming. Machine learning based methods are also needed to automate the process of developing optimization friendly surrogate models from higher fidelity ones. One such promising approach seeks to approximate lower dimensional nonlinear dynamics with (typically) higher dimensional linear models through the use of Dynamic Mode Decomposition and Koopman operator theory.[24,25] These techniques are demonstrated on the battery health service.

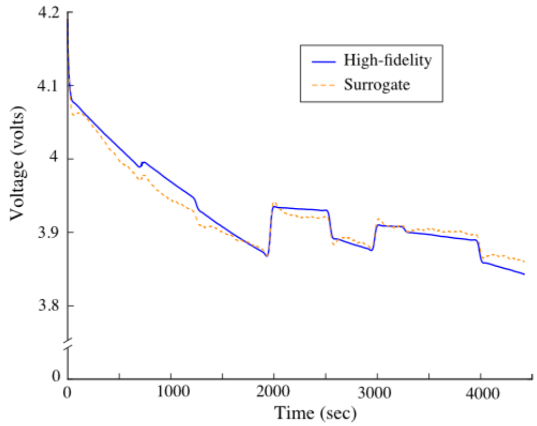
C. Battery Health

The battery health service provides information on the health and potential usage of a collection of Lithium-ion batteries powering a concept eVTOL air-vehicle. The service takes input of the expected power required for a trajectory from the flyable trajectory generation service. The service simulates a battery dynamics model and outputs the predicted battery usage throughout the flight, estimated state-of-charge at flight completion, probability of mission success, and uncertainty bounds on these metrics.

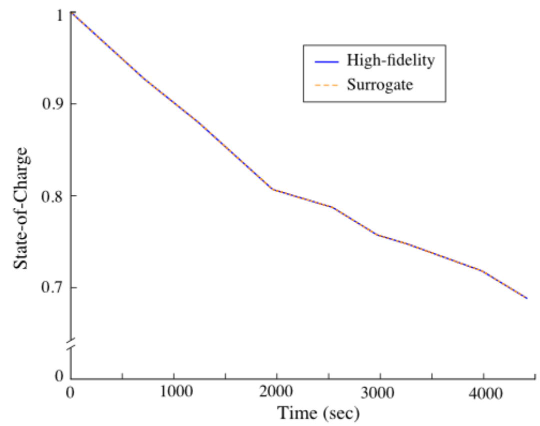
The ProgPy python package is used to simulate internal battery system dynamics. ProgPy is a NASA developed modular framework for building models of engineering systems and performing model-based prognostics, including state estimation, prediction, computation of remaining useful life, and uncertainty propagation.[26,27] Using this framework, the DRF health service simulates battery dynamics with a high-fidelity electrochemistry model. This model is well-established, based in first principles and empirical evidence, and accurately relates a Lithum-ion battery supply current to the resulting voltage discharge.[28]

In addition to the high-fidelity battery model, the battery health service also includes an option to simulate battery dynamics with a faster lower-fidelity surrogate model.[29] The procedure is summarized here. The surrogate model is developed using Dynamic Mode Decomposition (DMD), a powerful data-driven tool for identifying approximate linear dynamics from complex, non-linear systems. In its standard form DMD utilizes observable data from the system to generate a global linear approximation for a nonlinear state-space model. It has been shown to produce accurate look-ahead predictions for fluid dynamics applications for short time horizons on the order of tens of time steps into the future.[30] However, to accurately capture battery voltage discharge during an entire flight, much longer time horizons are required, hundreds to thousands of time steps in the future. We found that a standard DMD implementation was unable to capture battery discharge for these long prognostics horizons. To adjust the technique for this use-case, a physics-enhanced DMD method was developed, where the observable data was augmented with known physical states from the high-fidelity electrochemistry model. By leveraging physics-based information within the data-driven method, our resulting physics-enhanced DMD method was able to approximate voltage discharge throughout entire discharge cycles.

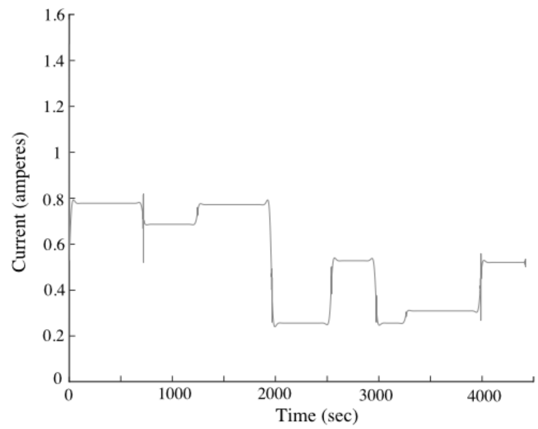
An example battery discharge prediction case, comparing high-fidelity and surrogate model, are shown in Fig. 7. The surrogate model voltage prediction is clearly less accurate than the high-fidelity electrochemistry model. The surrogate state-of-charge estimate, however, is nearly exact and perhaps more relevant in an



(a) Battery voltage.



(b) State of charge.



(c) Required battery supply current.

Figure 7: Prediction of battery voltage (a) and state-of-charge (b) for the battery-cell level current profile shown in (c). The plots compare the surrogate model to the high-fidelity model for the test trajectory from Fig. 5. The surrogate model was trained on randomly generated current profiles meant to be representative of the operational scenario.

operational planning context. The surrogate estimates are achieved with significant computational savings, enabling discharge prediction for a full trajectory in real-time. The powertrain model and control system assumptions used for the trajectory planning example are simplistic and preliminary. The trajectory planning solution uses rotor-speed control to maneuver the vehicle, but in the example shown the rotor-rates were not regulated to equalize when flying a trim velocity. This produced step changes in rotor-torque to maneuver, and hence the battery current shown in Fig. 7c as the vehicle moves through the target waypoints. This step-wise behavior is likely mitigated in a more realistic powertrain model and controller.

III. Multi-Vehicle Simulation

A 6 degree of freedom (6DOF) simulation of multiple quad eVTOL air-vehicles was developed for an in-house simulation framework called Reflection.[31] Reflection is a C++ Visual Studio solution that connects Visual Studio projects as modules for flexible simulation purposes including for studying autonomous air vehicle operations. In our implementation, the aircraft dynamics module parses a JavaScript Object Notation (JSON) file that contains model specification data and then simulates in real-time the inner-loop dynamics and integration. The inner-loop dynamics consist of a nonlinear stitched vehicle model[8,9] with Attitude Command Attitude Hold (ACAH) control that meets proposed additional requirements to ADS-33E-PRF for Level 1 disturbance rejection peak and bandwidth.[32] Outer loop attitude commands are routed to the vehicle module from a separate autopilot module, which also receives way-point information from a flight management computer (FMC) module. A modeling service is also envisioned to provide model fidelity management, for example to generate discretized linear models for trajectory generation, as required in problem (8), or to generate higher-fidelity models for real-time multi-vehicle simulation.

In addition to numerical simulation, Reflection is deployed to visualize trajectory progress during the mission. Modules for map displays and scene environments give the operator real-time visualization of both the aircraft and its surroundings. Maps can be loaded from simple bitmap images that are snipped from online maps and oriented in the simulation environment using anchors. Scene visualization can be populated with static people and buildings as well as other aircraft. For the CAP use-case considered here, two quad concept vehicles were simulated in Reflection with auto-piloted flight through a series of discrete trajectory points produced by the flyable trajectory generation service. Fig. 8 shows two example trajectories through lightning sites that were selected using the target selection and routing service.

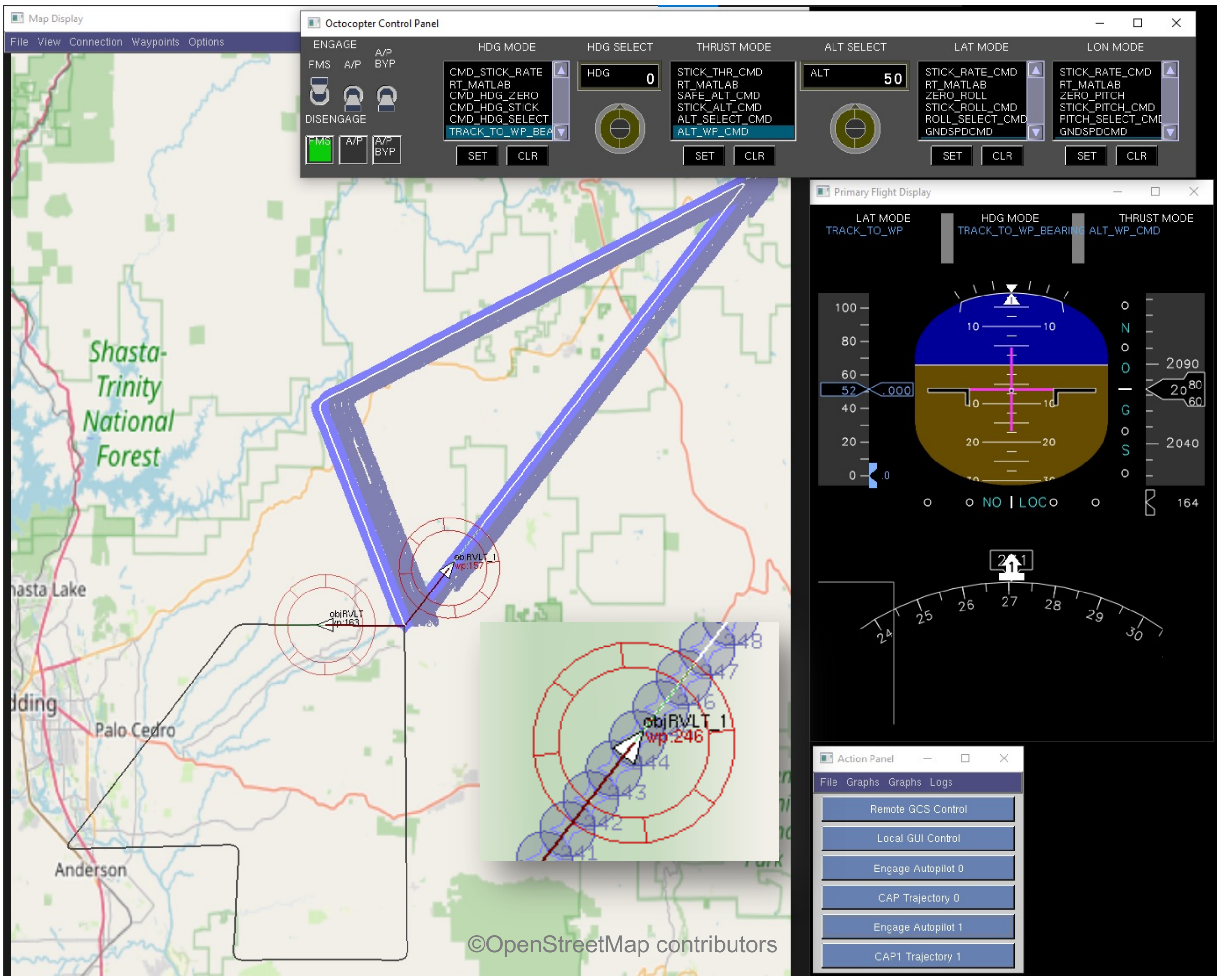


Figure 8: Reflection simulation display showing two air-vehicles simultaneously executing their respective missions. The discrete trajectory points were downloaded from the trajectory service after route optimization using the target selection and routing service. The selected purple trajectory line includes tolerance circles and waypoint IDs discernible in the zoomed view. The white line shows the planned flyable trajectory and the red line shows the tracked vehicle (white triangle) progress on its route.

IV. Conclusion

Three exemplar reasoning services were developed for NASA's Data and Reasoning Fabric, in order to explore implementation possibilities and best practices for developing service oriented intelligent systems. The reasoning capability was readily built on reliable technology stacks for integer programming, convex optimization, VTOL vehicle modeling, as well as algorithm and model packages for diagnostics and prognostics. The web-service capability was implemented using Flask, a commonly used web application framework, and the three services are deployed to the cloud. A graphical mission view user interface was developed on the Unity platform, and the trajectory plans developed by the three services working together were executed in a multi-vehicle simulation framework.

Of course, the three reasoning services presented here are far from constituting the sufficient set needed to ensure a safe autonomy driven ecosystem. Many additional services are needed. For example, unmanned aerial system air traffic control services, community airspace usage policy services, weather and wind information services, autopilot, perception, and collision avoidance services, digital twin and integrated vehicle health management services, and many more. Technologies for many, if not all, of these are currently under development in aerospace programs and projects worldwide. The key outcome here, however, is that the Data and Reasoning Fabric provides a foundation for organizing, securing, and ensuring that any number of services can interact amongst each other in a rapidly expanding fair and competitive marketplace.

Acknowledgements

This work was primarily supported by NASA's Convergent Aeronautics Solutions project, Data and Reasoning Fabric activity. The vehicle model that was used for trajectory planning and real-time simulation was developed under NASA's Revolutionary Vertical Lift Technologies project, and the high fidelity battery health model was developed under NASA's System-Wide Safety project. Where indicated, map images are copyrighted Mapbox (https://www.mapbox.com/about/maps) and OpenStreetMap contributors, available under the Open Database License https://www.openstreetmap.org/copyright.

References

[1] William Van Dalsem, Kenneth Freeman, Sandeep D. Shetye, Aditya N. Das, Kalmanje Krishnakumar, Aaron Swank, Peter Shannon, Luka Tomljenovic, Sandy Lozito, Helen Euler, Supreet Kaur, Channon Wong, and Joseph O'Brien. Data and reasoning fabric minimum viable product. In *AIAA AVIATION 2021 FORUM*, August 2021. DOI: 10.2514/6.2021-2386.

[2] William Van Dalsem, Sandeep Shetye, Aditya N. Das, Kalmanje S. Krishnakumar, Sandy Lozito, Kenneth Freeman, Aaron Swank, Peter Shannon, and Luka Tomljenovic. A data & reasoning fabric to enable advanced air mobility. In *AIAA Scitech Forum*, 2021. DOI: 10.2514/6.2021-2033.

[3] Moustafa Abdelbaky, Jiasi Chen, Alexander Fedin, Kenneth Freeman, Mohana Gurram, Abraham K. Ishihara, Carlee Joe-Wong, Christopher Knight, Kalmanje Krishnakumar, Isaias Reyes, Calvin Robinson, Peter Shannon, Sandeep D. Shetye, Luka Tomljenovic, and William Van Dalsem. Drf: A software architecture for a data marketplace to support advanced air mobility. In *AIAA AVIATION 2021 FORUM*, 2021. DOI: 10.2514/6.2021-2387.

[4] Vahram Stepanyan, Stefan Schuet, and Kalmanje S. Krishnakumar. An approach to reasoning service migration in data and reasoning fabric (drf) implementation. In *AIAA SCITECH 2022 Forum*, January 2022. DOI: 10.2514/6.2022-0787.

[5] Christopher Silva, Wayne R. Johnson, Eduardo Solis, Michael D. Patterson, and Kevin R. Antcliff. VTOL urban air mobility concept vehicles for technology development. In *2018 Aviation Technology, Integration, and Operations Conference*, Dallas, TX, June 2018. AIAA 2018-3847, doi: 10.2514/6.2018-3847.

[6] B. Lawrence, C. R. Theodore, W. Johnson, and T. Berger. A handling qualities analysis tool for rotorcraft conceptual designs. *The Aeronautical Journal*, 122(1252):960–987, June 2018.

[7] Carlos Malpica and Shannah Withrow-Maser. Handling qualities analysis of blade pitch and rotor speed controlled evtol quadrotor concepts for urban air mobility. In *VFS International Powered Lift Conference*. Vertical Flight Society, January 2020.

[8] Stefan Schuet, Carlos Malpica, Thomas Lombaerts, John Kaneshige, Shannah Withrow, Gordon Hardy, and Jeremy Aires. A modeling approach for handling qualities and controls safety analysis of electric air taxi vehicles. In *AIAA AVIATION FORUM*, June 2020. DOI: 10.2514/6.2020-3188.

[9] Eric L. Tobias and Mark B. Tischler. A model stitching architecture for continuous full flight-envelope simulation of fixed-wing aircraft and rotorcraft from discrete-point linear models. Special Report RDMR-AF-16-01, U.S. Army Aviation Development Directorate, April 2016.

[10] Flask, web development one drop at a time. https://flask.palletsprojects.com/en/2.1.x/, last checked June 1, 2022.

[11] Amazon web services. https://aws.amazon.com/, last checked June 1, 2022.

[12] Unity real-time development platform. https://unity.com/, last checked June 1, 2022.

[13] Pieter Vansteenwegen, Wouter Souffriau, Greet Vanden Berghe, and Dirk Van Oudheusden. A guided local search metaheuristic for the team orienteering problem. *European Journal of Operational Research*, 196(1):118–127, 2009.

[14]C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *J. ACM*, 7(4):326–329, oct 1960.

[15]Python mixed integer programming. https://www.python-mip.com, last checked Nov. 1, 2022.

[16]Jeremy Aires, Shannah Withrow-Maser, Allen Ruan, Carlos Malpica, and Stefan Schuet. Analysis of handling qualities and power consumption for urban air mobility (uam) evtol quadrotors with degraded heave disturbance rejection and control response. In *VFS 78th Annual Forum & Technology Display*. Vertical Flight Society, May 2022.

[17]S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.

[18]Inc. CVX Research. Cvx: Matlab software for disciplined convex programming, version 2.0 beta. http://cvxr.com/cvx, September 2012.

[19]CVXPY Convex Optimization for Evereyone. https://www.cvxpy.org, last checked Nov. 1, 2022.

[20]Yang Wang and S. Boyd. Fast model predictive control using online optimization. *Control Systems Technology, IEEE Transactions on*, 18(2):267–278, March 2010.

[21]Thomas J. Stastny, Adyasha Dash, and Roland Siegwart. Nonlinear mpc for fixed-wing uav trajectory tracking: Implementation and flight experiments. In *AIAA Guidance, Navigation, and Control Conference*. American Institute of Aeronautics and Astronautics, January 2017.

[22]Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[23]Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

[24]Milan Korda and Igor Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, 2018.

[25]Steven L. Brunton, Bingni W. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control. *PLOS ONE*, 11(2):e0150171, Feb 2016. DOI: 10.1371/journal.pone.0150171.

[26]Christopher Teubert, Matteo Corbetta, Chetan Kulkarni, Katelyn Jarvis, and Matthew Daigle. Prognostics models python package, May 2022.

[27]Christopher Teubert, Matteo Corbetta, and Chetan Kulkarni. Prognostics algorithm python package, May 2022.

[28]Matthew Daigle and Chetan S. Kulkarni. Electrochemistry-based battery modeling for prognostics. In *Annual Conference of the PHM Society*, 2013. https://doi.org/10.36001/phmconf.2013.v5i1.2252.

[29]Katelyn Jarvis, Matteo Corbetta, Christopher Teubert, and Stefan Schuet. Enabling in-time prognostics with surrogate modeling through physics-enhanced dynamic mode decomposition method. In *Annual Conference of the PHM Society*, 2022.

[30]J Nathan Kutz, Steven L Brunton, Bingni W Brunton, and Joshua L Proctor. *Dynamic mode decomposition: data-driven modeling of complex systems*. SIAM, 2016.

[31]Corey A. Ippolito, Sebastian Hening, Shankar Sankararaman, and Vahram Stepanyan. A modeling, simulation and control framework for small unmanned multicopter platforms in urban environments. In *AIAA Modeling and Simulation Technologies Conference*, 2018. DOI: 10.2514/6.2018-1915.

[32]Chris L. Blanken, Mark B. Tischler, Jeff A. Lusardi, and Tom Berger. Proposed revisions to aeronautical design standard - 33E (ADS-33E-PRF) toward ADS-33F-PRF. Special Report SR-FCDD-AMV-19-01, U.S. Army Aviation Development Directorate, September 2019.