Data Augmentation for Intelligent Contingency Management Using Generative Adversarial Neural Networks

Newton H. Campbell Jr.*
NASA Goddard Space Flight Center, Greenbelt, MD, 20771, USA

Hari S. Ilangovan† and Irene Gregory‡
NASA Langley Research Center, Hampton, Virginia, 23681, USA

Sarkis S. Mikaelian§
NASA Armstrong Flight Research Center, Edwards AFB, CA 93524, USA

Artificial intelligence (AI)-based techniques for intelligent contingency management (ICM) require that intelligent agents learn various aspects of system dynamics to create and execute contingencies. For high assurance contingency management, agents achieve the most compelling results through supervised or semi-supervised machine learning, for which agents require large datasets to learn the dynamics of the system. Unfortunately, data collection in aerospace applications can be costly, due to both time and resources. Presented work describes a framework for data augmentation of ICM databases containing training data for machine learning models. This framework populates the database with the outputs of generative adversarial network (GAN) models that were trained on flight data. Methods for evaluating the suitability of these models based on the equations of motion, as well as other physical constraints, are discussed. The paper demonstrates the utility of this database for training intelligent agents on the NASA T2 generic transport aircraft model and experimental vertical takeoff and landing (VTOL) simulation model.

I. Nomenclature

G	=	Generator Neural Network (Function)
D	=	Discriminator Neural Network (Function)
n	=	Baseline neural network layer size
x	=	Training data
$\hat{x}$	=	Generator Output
w	=	Neural Network Weights
z	=	Generator Input
β_1, β_2	=	Decay rates
β_{I}	=	Initial bias vector
ϵ	=	Fuzz factor
η	=	Learning rate parameter
μ, ν	=	Exponential moving averages (training momentum)
θ_g, θ_d	=	Generator, Discriminator Parameters (weights)
L_{GAN}	=	Vanilla GAN Loss Function
$P(z)$	=	Random Noise Vector Distribution (Standard Normal)
$P(x)$	=	Real Data Distribution
$P_{\theta_g}(\hat{x})$	=	Generated Data Distribution
$\dot{}$	=	Time Derivative
p, q, r	=	Angular Velocity - Roll, Pitch, Yaw ($\frac{rad}{s}$)
u, v, w	=	Translational Velocity ($\frac{m}{s}$)
V	=	Magnitude of Velocity ($\frac{m}{s}$)
α, β, γ	=	Airflow: Attack, Sideslip, Flight Path Angles (rad)
ψ, θ, ϕ	=	Attitude: Yaw, Pitch, Roll Angles (rad)
$\delta_{\mathrm{e}}, \delta_{\mathrm{a}}, \delta_{\mathrm{r}}$	=	Controls - Elevator, Aileron, Rudder ($\frac{N}{m^2}$)

*Senior Principal Solutions Architect, Science Applications International Corporation (SAIC), NASA Enterprise Applications Service Technologies.
†Senior Data Scientist, Science Applications International Corporation (SAIC), NASA Enterprise Applications Service Technologies.
‡NASA Senior Technologist for Advanced Control Theory and Application, NASA, Fellow AIAA.
§Pathways Engineering Co-Op, NASA Aeronautics Research Mission Directorate.

II. Introduction

Robust contingency management requires being prepared to deal with unforeseen circumstances, and recognizing and responding to unpredicted events. Unforeseen and unpredicted circumstances happen routinely. While the likelihood of a specific unplanned circumstance may be low, given the extremely large number of unplanned circumstances, the likelihood of some unplanned events occurring is high. In today's aviation operations, contingency management is performed by highly trained human operators, utilizing a range of behavioral strategies[1]. If autonomous flight is to become a reality, then robust contingency management must be able to make intelligent decisions which would require inclusion of learning elements. This paper discusses some of the approaches utilized to develop these capabilities.

Machine learning models often require significant data to produce reliable inferences. Under the NASA Intelligent Contingency Management program[2], the machine learning scenarios are focused on the aerospace domain. These scenarios **scope** the problem of understanding the necessary data augmentations to a high-fidelity simulation environment, in which operators assign random trajectories to experimental vehicle models. Given the speed and computational constraints of our (or any) simulation environment, acquiring data by manually (or in batch) running large numbers of human-parameterized scenarios is insufficient for many use cases.

For an arbitrary use case, the diversity, fidelity, granularity, cleanliness, and size of the necessary data to execute a specific machine learning Concept of Operations (CONOPS) can vary. In this paper, data diversity refers to properties of the data that describe its richness, divergence, and evenness. Fidelity refers to the capacity of an aircraft model to represent vehicle flight dynamics in the real world. Granularity describes the level of detail with which temporal and spatial properties of the model can be observed through the data. Cleanliness refers to the lack of incorrect, corrupted, or incomplete data relative to the phenomena under observation. Finally, data size refers to the number of scenarios (trajectories) that are incorporated into the training/validation dataset.

The data's variance and homogeneity are controllable in the ICM simulation environment, where researchers can produce and characterize trajectories and aircraft failures based on desired features. Consequently, this allows researchers to create a reasonably diverse dataset. Furthermore, the simulation environment permits parameterization of both fidelity and granularity for various scenarios. Finally, if necessary, particularly when experimenting with new algorithms, the environment can be parameterized only to produce clean, accurate flight data (no simulation of noise or error). Thus, this paper concentrates on a scenario in which the remaining factors for creating an ideal machine learning experimentation environment are the size and richness of the data.

Numerous high-performance computing studies exist that research the problem of running large numbers of aircraft model simulations to address problems in the aerospace domain[3–7]. In other domains, the recognized alternative to running massive parallel simulations is *variational data augmentation*, in which a training dataset is enriched by synthetic data derived from a smaller series of simulations or experiments[8–12]. The study in this paper explores generative machine learning[13] to address the problem of creating rich, sizable datasets to train more expressive machine learning mechanisms for desired CONOPS. Specifically, this paper contributes the following:

- The application of generative adversarial networks (GAN)[14] to high-fidelity aircraft simulation data to provide granular methods for controlling the richness of simulation output data in desired scenarios
- A technique for maintaining the fidelity of simulation output data by training generative models to restrict their inferences to known physical constraints, such as the canonical aircraft equations of motion (6DOF EOM)
- A data pipeline design and evaluation methods for variational data augmentation of commercial aircraft and urban air mobility model simulations

Section II contains a brief review the aerospace and machine learning concepts that justify the methodology for this study. Section III fully details the methodology for storing experimental aircraft simulation data and training constrained generative models, as well as the methodology to measure the performance of these generative models. Section V contains details of experiments and corresponding results using these methodologies on the NASA AirSTAR T2[15] and Lift-Plus-Cruise[16] aircraft configurations. Section VI describes the impact of these results and Section VII explicitly states the conclusions of the study.

III. Background

A. Reduced-Order Modeling for Aerospace

The algorithms behind machine learning have matured significantly over the last two decades. However, for practical, real-world applications, *data acquisition*, the process of sampling signals that reflect real-world conditions, is still a limiting factor. Reduced-order modeling is a common approach for creating a computationally tractable model that

(a) T-2 aircraft in flight (credit: NASA Langley Research Center)

(b) NASA Lift-Plus-Cruise Vehicle Schematic (credit: NASA Langley Research Center)

Fig. 1 Images of aircraft under experimentation on the NASA Intelligent Contingency Management program.

address the problem of augmenting physics data with its own variations, produced by a lower-fidelity model[17–20]. High-performance computing (HPC) is an alternative solution, where the necessary parameters for a high-fidelity simulation are chosen and large numbers of key scenarios are gathered[21, 22]. The fields of fluid dynamics, jet noise, and turbulence modeling have each benefited significantly from an amalgamation of these two approaches[23–26]. One such amalgamation is the use of non-intrusive machine learning methods for inducing non-linear models that approximate physical aerospace processes of interest[27, 28]. In these cases, results demonstrated accurate variations of high-fidelity physical phenomena. However, these models' efficiency (during training) and stability require significant improvement to scale the approaches. Furthermore, integrating the known physical equations that govern the modeled phenomena makes integrating additional parameters and physical constraints difficult. Therefore, it may be challenging to adapt the methods to similar models or different phenomena. The field of reduced-order modeling for aerospace applications requires flexible models that can generally learn and express dynamic, non-linear, high-dimensional features while maintaining physical consistency.

B. Vehicle Models of Interest

In previous works, the ability for a surrogate model to accurately produce variations of an original physical phenomena was applied to the NASA T2 aircraft[15, 29], shown in Figure 1a. The study used T2 flight data, provided by the NASA Airborne Subscale Transport Aircraft Research (AirSTAR) program during a previous study on simulating loss of control(LOC)[30], to train a conditional variational autoencoder (CVAE)[31]. The probability of a trained CVAE to reconstruct the T2 flight data was used as the threshold for a detector. The research aligned detections with periods of flight in which the aircraft experienced LOC. These results demonstrated a 97% true-positive rate for detecting LOC. While the detector requires some modification to reduce its false-positive rate, this study (and the subsequent DOE for neural network architectures[32]) demonstrated a capacity for generative networks to be used as a reduced-order model to synthesize flight dynamics data for standard and abnormal flight maneuvers. This capacity is predicated on how many examples of these maneuvers or events are in the training data. Therefore, the training data for synthesizing a model should be labeled such that a model can learn the physics of specific flight phenomena. Furthermore, the study highlights that the T2 aircraft data provided by AirSTAR can be provided to machine learning algorithms to make inferences about a commercial aircraft in flight.

In recent years, electric vertical takeoff and landing (eVTOL) vehicles have become a central focus of the Urban Air Mobility (UAM) domain[33–35]. NASA Langley Research Center has focused on this class of aircraft under its Autonomous Systems - Intelligent Contingency Management (ICM) program[36]. The project is developing a platform in which novel artificial intelligence algorithms are applied to experimental aircraft models in a nonlinear simulation environment, to support ICM mission goals[2]. The platform allows a user to easily specify a type of experimental vehicle, vehicle and environmental parameters, and trajectories. The first aircraft model of focus when developing this platform is the Lift-Plus-Cruise (L+C) aircraft[16], shown in Figure 1b. The ICM implementation of the Lift-Plus-Cruise simulation includes six degree-of-freedom dynamics, as well as aerodynamic, control design, propulsion, actuator, and atmosphere models[37–39]. The simulation is developed in MathWorks Matlab/Simulink®[40]. The vehicle inputs and environmental parameters can be provided to the batch simulations, with output specified as MATLAB binary data files that can be parsed and used by machine learning algorithms.

C. Generative Modeling

While both the T2 and L+C aircraft models have demonstrated their utility to be used in machine learning algorithms, much like in other cases, the size and richness of these datasets does not allow for artificial intelligence training beyond proof-of-concepts. This is due to the necessity of manual parameterization of the vehicle models and the inherent slowness of running their high-fidelity simulations. Augmentation of this type of datasets can be done using generative adversarial networks (GANs), a flexible neural network model for synthesizing similar data based on examples[14]. For variational data augmentation, the model can serve as a reduced-order model for providing specific variables necessary for data-rich experimentation.

Much like the variational autoencoder model, GANs attempt to understand the latent process that produced the provided training data. A GAN is comprised of two neural networks, learning in competition with each other: a *generator* G and a *discriminator* D. A GAN works as follows: Let $\hat{x}$ be a dataset that is output by G. At each training step, the D makes a prediction as to whether $\hat{x}$ was provided by G or the original data source. Once the D cannot reliably distinguish between the two (i.e. a prediction that $\hat{x}$ was part of the original data with probability of 50%), then it is confirmed that G is producing data that is an accurate (statistical) variation of the original data source.

A variety of GANs exist that expand on this loss function and the standard architecture, and can serve the data augmentation task[41–44]. This study focuses on characterizing the traditional GAN's capacity for doing this, along with a small augmentation to this loss function. Once characterized, the performance of these base case models on the vehicle dynamics of the vehicle models of interest can be scaled to the GAN alternatives.

Most studies use GANs for their ability to generate images, video, or text. These use cases have become popular because a human can trivially verify that the model training has resulted in the GAN generating compelling data. However, this kind of evaluation is not scalable and difficult to apply to the synthesis of time series data, even when observing plots. In recent years, several studies have developed GANs for synthesizing time series that solve stochastic differential equations[45] or include discrete properties of the original phenomena under observation[46]. However, few studies on GANs include robust evaluation methods that analytically evaluate that data similar to the training data is being produced, particularly when synthesizing physics data. In practice, GAN outputs are evaluated both qualitatively and quantitatively. Qualitative evaluation usually requires human visual assessment of results by inspecting the quality of the generated samples. Some quantitative evaluation metrics, including Inception Score (IS), maximum mean discrepancy (MMD), and Fréchet Inception Distance (FID) have been previously proposed to evaluate GAN performance[43]. However, it is complicated to evaluate domain-specific (i.e. flight vehicle dynamics) time-series data without specific application of constraints that are inherent to the domain.

IV. Methodology

The following section describes the methods that enable the CONOPS illustrated in Figure 2. The CONOPS proceeds as follows: First, after parameterizing and running an ICM simulation (in a specific flight regime), data is stored in a flight dynamics database to query training data. Next, automated software continuously samples a noise distribution $p(z)$, providing a sample z as input to a trained generative network. The generative network produces an output that a signal filter then processes for smoothing. Next, the automated system validates the output against an evaluator, which validates the kinematic constistency and other physical or flight vehicle-specific constraints. Finally, if the output passes the evaluation, the synthetic data is inserted into the database, labeled as synthetic.

A. Data Storage

The methodology for this study focuses on the storage of flight dynamics data that will be subject to variational data augmentation while having both simulated and synthesized data readily available to machine learning algorithms. The following storage goals were the focus:

- Minimize the number of queries and input & output (I/O) operations necessary to load a set of flight dynamics time-series data
- Minimize storage requirements for high-fidelity simulation data
- Query flights and subsets of flights based on vehicle, environmental factors, failures, maneuvers, changing flight regimes, and specific experiments
- Maintain a mapping of flight dynamics observations provided by the simulator or real-world experiments to data provided by generative models

Tables 2 and 3, listed in the Appendix, describe the flight dynamics data that need to be stored by the ICM program. The input data encompass flight parameters, input file specifications, environmental parameters, and software inputs to a

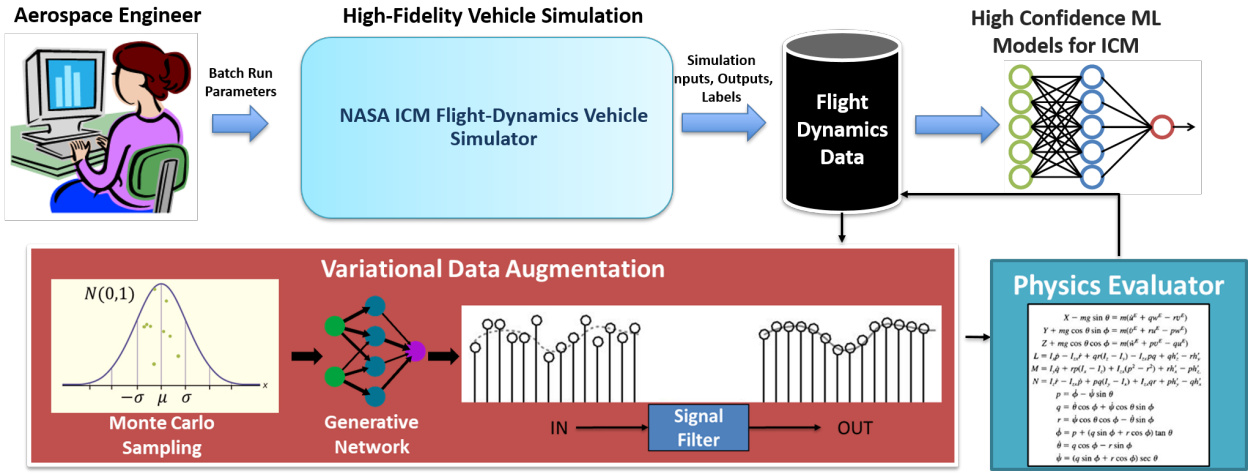


Fig. 2 Variational Data Augmentation Concept of Operations for ICM

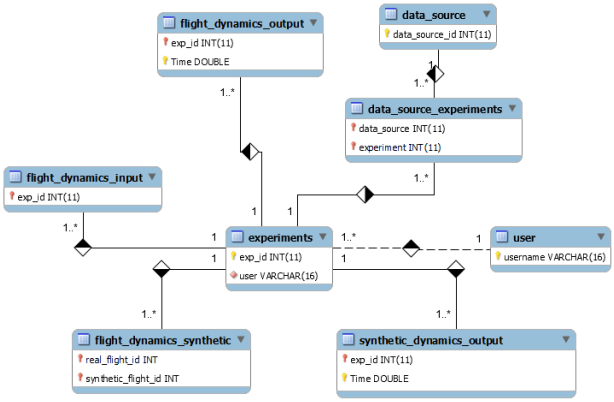


Fig. 3 High-Level Enhanced Entity-Relationship Diagram for ICM Flight Dynamics Data Storage

simulation or a set of batch simulations. The output data include environmental factors, control inputs and outputs of the simulation in multiple frames, and intentional failure parameters. With 368 inputs and 468 outputs for each flight, flat-file schemas would make querying flights difficult when nuanced applications arise. Furthermore, neither popular time-series databases nor relational databases inherently have all the features necessary to store data efficiently.

The high-level diagram for the ICM data storage solution is shown in Figure 3. The solution needs to capture metadata and time-series data for each flight experiment. Users need to adequately categorize subsets of data that they can later query to train machine learning models for specific tasks. Moreover, data sources must be mapped to experiments (flights) such that other organizations and entities have full traceability for experimental results.

In the data storage solution design, complications arise concerning the storage of highly granular time-series containing 468 measured variables. As Table 3 shows, the outputs can be nicely categorized. In a normal relational database paradigm, normalization rules dictate that one should create multiple tables based on these categories to support normalization beyond the third-normal form[47]. Recall that this study aims to understand the feasibility of storing large variations of experimental flight dynamics simulations, to replace the need of designing parallel or HPC schemes that run a high-fidelity simulation billions of times. Given the significant number of rows that can be stored, creating tables based on all the categories and subcategories in Table 3 using a standard foreign key structure would result in significant computation for joins, high data insertion costs, and additional caching and storage. Therefore, a large table containing all flight dynamics output variables was designed, with a focus on proper indexing to optimize queries. Proper indexing requires specifying the appropriate order of columns when specifying the index, such that rows are sorted and grouped in ways that benefit queries.

This study chose to leverage a relational database, MySQL InnoDB[48], because of its B-tree and B+tree models

for index-based storage, which allow fast access, querying and insertion for big data[49, 50]. Storing flight vehicle dynamics simulation data requires insertion queries that capture observations that range from 10^3 to 10^6 in orders of magnitude. Because we use one large table to store the simulation flight dynamics outputs, INSERT queries do not require significant modification or indexing. The queries are formatted using the MySQL *executemany()* function, which leverages an Extended Inserts feature that dramatically improves the performance of bulk inserts compared to a series of single inserts[51]. This feature produces a single statement that can insert all required rows atomically and is optimized for the database. In addition, partitioning allows the table to be split across multiple disks based on indexes, allowing even faster lookup times. After each simulation, software adapters run the following steps to insert the MATLAB inputs and outputs into the DB:

1) A user-defined description, experiment keywords, and user/platform traceability information for the simulation experiment are inserted into the database using a single INSERT query.
2) The simulation insertion parameters (*flight_dynamics_inputs*) are converted into a JSON dictionary and stored as a JSON data type, using a single INSERT query.
3) The granularity of the data is downsampled (to user-specified level of granularity) using the average across the desired sample rate.
4) MySQL INSERT queries are concatenated and executed using a single network bind, query, and close operation.

The most common queries under the Flight Dynamics database solution will contain ORDER BY, GROUP BY, and WHERE (equals) clauses. In most cases, query results involving this table would be grouped by or equal to a particular experiment ID. The data will also need to be filtered to capture specific flight lengths or properties (i.e., failures to specific actuators, velocities/rates above a certain threshold). For example, the project has particular interest in flight regimes under which a failure has occurred. Therefore, the composite observation time and experiment ID were chosen as the primary keys of this table. An additional composite index was assigned to optimize search, using the following:

<*observation_time*,*experiment_id*,*surface_failure_initiation*,*propeller_failure_initiation*>

This is one example of a composite index (or key) that permits fast search queries for pulling specifically the parts of flight regimes during which failure has (or has not) occurred, and to do so over multiple flights. Because time and experiment will be the most likely criteria for queries, they are first in the composite index. This is useful for training a machine learning algorithm on segments of a flight where a specific failure has occurred. A query like the following would produce the necessary (concatenated) time-series:

```
SELECT exp_id, observation_time, u, v, w, SQRT(POWER(u,2)+POWER(v,2)+POWER(w,2) AS V
FROM flight_dynamics_output
WHERE surface_failure_3_initiate != 0 OR propeller_failure_3_initiate != 0
ORDER By exp_id ASC, Time ASC;
```

When executing this query, MySQL attempts to eliminate as many rows from consideration as possible. To consider all options, it executes a full index scan of the single $flight\_dynamics\_output$ table. The search starts with observation time and experiment ID, because they are both primary keys, as well as the first keys in the composite index described above. The search then parses branches of the tree where the indices of the columns in the WHERE clause are not equal to 0. When searching for records that contain surface or propeller initiated failures, an optimal indexing specification should maximize the ratio of branches/rows that are examined to the number of rows (B-Tree leaves) that should be returned.

The optimal way to prepare a dataset for model training is to first query a list of flights and then to separately query each time-series. For example, to acquire a list of flights in which the L+C aircraft execute the optimal response collision avoidance (ORCA) algorithm[52] without turbulence, a typical query to acquire flight metadata would be:

```
SELECT e.exp_id as exp_id, e.short_name as short_name, e.description, (MAX(Time)-MIN(Time
    ↪ )) AS num_seconds, COUNT(Time) AS num_observations, i.flight_dynamics_inputs->'$.
    ↪ case' as flight_regime, i.flight_dynamics_inputs->'$.trajFile' AS trajectory_file,
    ↪  i.flight_dynamics_inputs->'$.vehicleType' AS vehicle, i.flight_dynamics_inputs->'
    ↪ $.SwitcheTurbulenceOn' AS turbulence, i.flight_dynamics_inputs->'$.fmType' AS
    ↪ fmType
FROM experiments e INNER JOIN flight_dynamics_outputs o ON e.exp_id=o.exp_id INNER JOIN
    ↪ flight_dynamics_inputs i ON e.exp_id=i.exp_id
WHERE short_name='ORCA' GROUP BY exp_id HAVING num_observations < 900 AND turbulence = 0
```

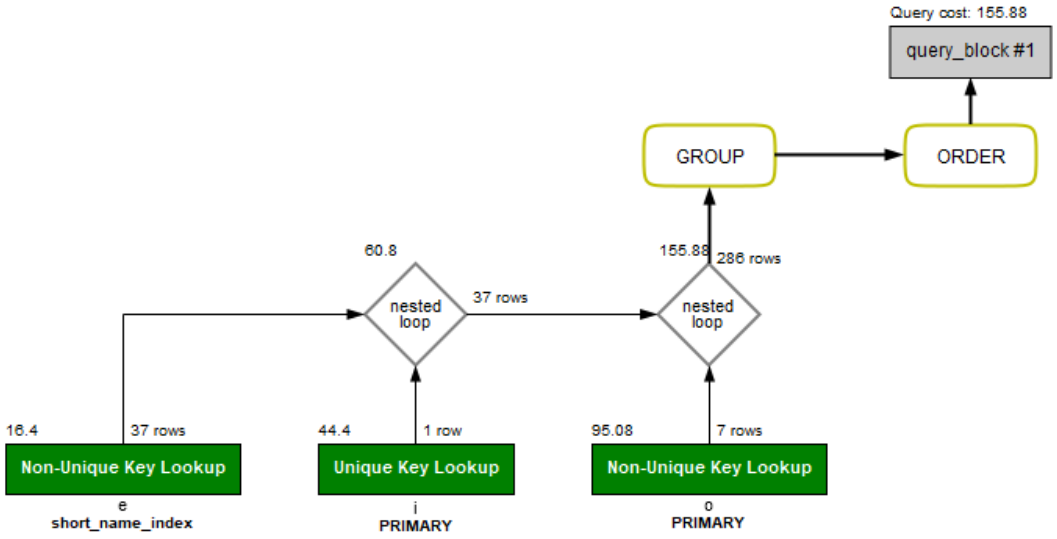


Fig. 4 Visualization of query execution for a query on L+C flights that executed collision avoidance maneuvers without turbulence.

```
ORDER BY exp_id
```

Figure 4 illustrates a tree of events that MySQL uses to process this query. First, the query executes a set of inner joins on 3 tables based on text criteria from a column in one of those tables. Next, it groups the data (aggregating values such as *Time* based on a known key). Finally, it executes two comparison clauses based on information in the larger *flight_dynamics_output* table. Because the experiments table has an index for the column `short_name`, the server saved time by filtering the number of experiment rows that needed to be analyzed and the ratio of rows examined to rows produced was 100%. Once the experiment IDs with the appropriate criteria are acquired, an additional query to retrieve the actual time-series observations based on the experiment IDs from this query also retrieves the necessary rows with 100% efficiency.

B. Generative Adversarial Networks for Augmenting Flight Data

To serve as a base case for utility in this domain, this study begins with the original architecture for GANs [14], referred to as a vanilla or *baseline* GAN, and then compares this performance against an architecture where kinematic consistency is verified during training, commonly referred to as a *physics-constrained* GAN.

1. Preliminaries

The architecture and methods for training and generating data for the ICM Flight Dynamics database using the original GAN design follows the goal of training G such that it can be provided a random noise vector z as input, and output synthetic time-series data $\hat{x}$ that aligns with the original data distribution $p(x)$.

To do so, the goal of the training step is to train parameters θ_g and θ_d to find the Nash-Equilibrium [53] of the two-player game with players G and D. After sufficient training, the GAN is able to achieve this equilibrium and G is typically able to reproduce a desired form of synthetic data that closely resembles x. The loss function is then determined using $G(z)$ and $D(x)$ as:

$$L_{\text{GAN}}(G(z), D(x)) = \mathbb{E}_{x\sim P(x)}\left[\log D(x)\right] + \mathbb{E}_{z\sim P_{(z)}}\left[\log(1 - D(G(z))\right]^2 \tag{1}$$

During training, θ_g and θ_d are determined by manipulating the weights w of G and D based on this function, using the Adam optimizer[54], a modified version of Gradient Descent. This optimizer showed comparatively stable learning

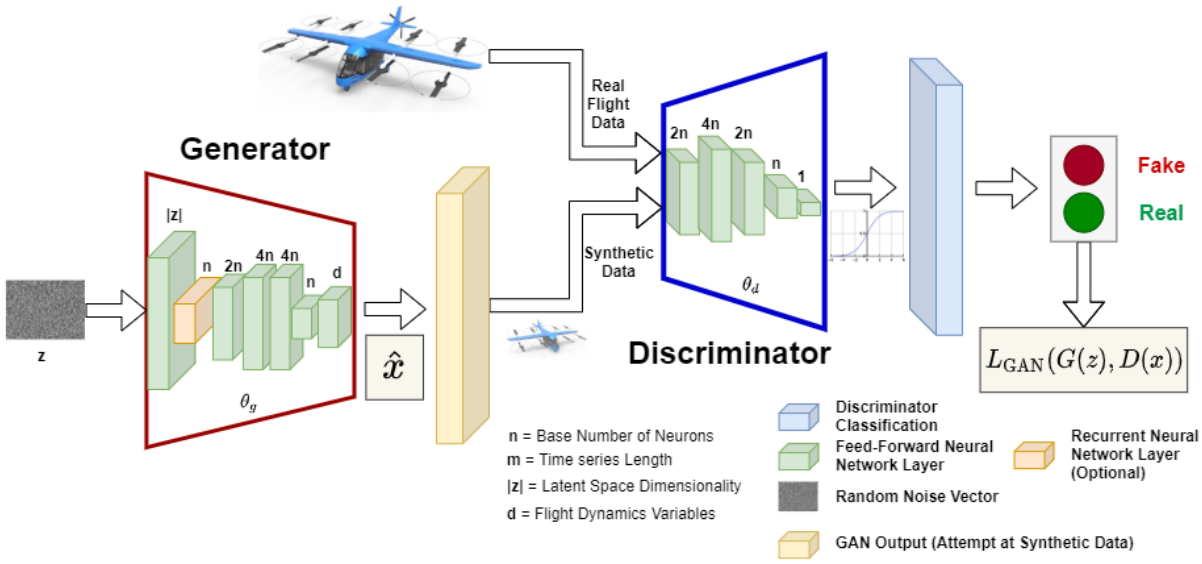


Fig. 5 Baseline architecture for a generative adversarial network used during experimentation.

in previous experiments on optimization techniques and latent space learning in the aerospace domain[32]. The goal of the optimizer is to update the weights w at each timestep t using the equation:

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}} + \epsilon} \cdot \hat{\mu} \tag{2}$$

with bias corrections

$$\begin{aligned} \hat{\mu}_t &= \frac{\mu_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \end{aligned} \tag{3}$$

and momenta

$$\begin{aligned} \mu_t &= \beta_1 \mu_{t-1} + (1-\beta_1)\frac{\partial L}{\partial w_t} \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2)\left[\frac{\partial L}{\partial w_t}\right]^2 \end{aligned} \tag{4}$$

with $m_t, v_t = 0$.

2. Architecture

Figure 5 shows the architecture for the baseline GAN. This neural network architecture is constructed using only fully-connected layers. The sizes of the hidden layers of G and D are parameterized to be multiples of a baseline size n. G is comprised of 6 fully-connected hidden layers that increase and then decrease as it approaches its output ($\hat{x}$) ($|z|$ -> n -> $2n$ -> $4n$ -> $4n$ -> n). The second layer of G is an optional recurrent neural network (RNN) layer, to consider short-term dependencies when synthesizing outputs. All hidden layer activation functions are Exponential Linear Units (ELU)s [55], based on previous results regarding their ability to encode and decode probability distribution samples of a latent space for aerospace applications[32]. D is comprised of 4 feed-forward layers, also with ELU activation functions ($2n$ -> $4n$ -> $2n$ -> n (Dropout) -> 1). The final layer of D is a single neural network node with a sigmoid function, for binary classification.

A key feature of this research is the ability to learn the statistical distributions for **unscaled** data. To successfully have G learn the physical scales on which each flight measurement exists in shorter training times, the final layer of G is provided bias vector β_I. For t timesteps and d flight variables, let x be defined as follows:

$$x = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,d} \\ x_{2,1} & x_{2,2} & \dots & x_{2,d} \\ x_{3,1} & x_{3,2} & \dots & x_{3,d} \\ \dots x_{t,1} & x_{t,2} & \dots & x_{t,d} \end{bmatrix} \tag{5}$$

Let $\mathcal{N}(\mu, \sigma^2)_i$ be a normal distribution parameterized with the mean and variance of the i^{th} column (flight variable) of x. When the network is constructed, β_{I} is a sampling of the normal distribution for each flight measurement in the training dataset:

$$\beta_{\text{I}} = \begin{bmatrix} \mathcal{N}(\mu, \sigma^2)_1 & \mathcal{N}(\mu, \sigma^2)_2 & \dots & \mathcal{N}(\mu, \sigma^2)_d \end{bmatrix} \tag{6}$$

C. Evaluation

This section describes the methodology to evaluate the GAN's output against the equations of motion (6DOF-EOM) and emphasizes how this can be extended to other physical constraints. The method measures properties of time-series produced by the GAN generator on the basis of training stability and kinematic consistency.

1. Aircraft Equations of Motion

This GAN evaluation methodology is based on algorithms for Kinematic Consistency Checks(KCC). In KCC, an algorithm feeds flight dynamics measurements (or in this case, synthesized data) to a 6DOF aircraft model and compares the resulting translational and angular velocities with their measured counterparts. Let flight states X and the control surface deflections δ be defined as:

$$X = \begin{bmatrix} u \\ v \\ w \\ \phi \\ \theta \\ \psi \\ p \\ q \\ r \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \text{Translational Velocities} \\ \\ \\ \text{Euler Angles} \\ \\ \\ \\ \text{Angular Rates} \\ \\ \end{matrix} \quad \text{and} \quad \delta = \begin{bmatrix} \delta_{\text{e}} \\ \delta_{\text{a}} \\ \delta_{\text{r}} \end{bmatrix} \begin{matrix} \text{Elevator} \\ \text{Aileron} \\ \text{Rudder} \end{matrix}$$

Commonly, for KCC, the aerodynamic force and moments are also assessed as the response or dependent variables. They are excluded from this study. Instead, this approach analyzes the flight dynamics states and control surface deflections alone. The EOM relationships between Euler angles, angular velocities, and angular accelerations $(\dot{\phi}, \dot{\theta}, \dot{\psi})$ are described by the following system of equations:

$$\begin{aligned} \dot{\phi} &= p + (q \sin(\phi) + r \cos(\phi)) \tan(\theta) \\ \dot{\theta} &= q \cos(\phi) - r \sin(\phi) \\ \dot{\psi} &= q \tfrac{\sin(\phi)}{\cos\theta} + r \tfrac{\cos(\phi)}{\cos(\theta)} \end{aligned} \tag{7}$$

To evaluate the kinematic consistency of the GAN-synthesized p, q, r against the GAN-synthesized ϕ, θ, ψ, a cubic interpolation algorithm for smooth attitude interpolation[56, 57] computes angular velocities and accelerations along a spline, parameterized by ϕ, θ, ψ. Let the computed rates from this spline be $\hat{\dot{\phi}}, \hat{\dot{\theta}}, \hat{\dot{\psi}}$. The mean-squared error calculation provides a metric for the physical consistency between the GAN-synthesized rates and the rates computed from the GAN-synthesized Euler angles as:

$$\begin{aligned} \epsilon_p &= \left(\dot{\phi} - \hat{\dot{\phi}}\right)^2 \\ \epsilon_q &= \left(\dot{\theta} - \hat{\dot{\theta}}\right)^2 \\ \epsilon_r &= \left(\dot{\psi} - \hat{\dot{\psi}}\right)^2 \end{aligned} \tag{8}$$

The KCC verifies the body-axis translational velocity components and relative airflow angles against the following system of equations:

$$\begin{aligned} V &= \sqrt{u^2 + v^2 + w^2} \\ \alpha &= \tan^{-1} \tfrac{w}{u} \\ \beta &= \sin^{-1} \tfrac{v}{V} \\ \gamma &= \theta - \alpha \end{aligned} \tag{9}$$

The translational velocity components and airflow angles are compared against each other for kinematic consistency using the following mean-squared errors:

$$\begin{aligned} \epsilon_u &= \left(u - \tfrac{w}{\tan\alpha}\right)^2 & \epsilon_\alpha &= \left(\alpha - \tan^{-1}\tfrac{w}{u}\right)^2 \\ \epsilon_v &= \left(v - \sqrt{V^2 - u^2 - w^2}\right)^2 & \epsilon_\beta &= \left(\beta - \sin^{-1}\tfrac{v}{V}\right)^2 \\ \epsilon_w &= (w - u\tan\alpha) & \epsilon_\gamma &= (\gamma - (\theta - \alpha))^2 \end{aligned} \tag{10}$$

The metrics for evaluation are defined by the error vector ε:

$$\varepsilon = \begin{bmatrix} \epsilon_p & \epsilon_q & \epsilon_r & \epsilon_u & \epsilon_v & \epsilon_w & \epsilon_\alpha & \epsilon_\beta & \epsilon_\gamma \end{bmatrix} \tag{11}$$

2. Physics-Constrained Training

For flight dynamics applications, a trained GAN should reliably produce time series with dependencies that align with the equations in the previous section. For clarity, let f, g, h be evaluation functions. The evaluation dependencies can be re-written as:

$$\begin{aligned} \begin{bmatrix} \phi & \theta & \psi \end{bmatrix} &= f(p, q, r) \\ \begin{bmatrix} u & v & w \end{bmatrix} &= g(\alpha, \beta, V) \\ \begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix} &= h(u, v, w, V, \theta) \end{aligned} \tag{12}$$

For neural network applications, these dependencies can trivially form a set of loss functions, $L_{\text{ang}}, L_{\text{vel}}, L_{\text{flow}}$, respectively, by taking the squared error of both sides:

$$\begin{aligned} L_{\text{ang}}\left(G(z)\right) &= \tfrac{1}{N}\left[f(p, q, r) - \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}\right]^2 \\ L_{\text{vel}}\left(G(z)\right) &= \tfrac{1}{N}\left[g(\alpha, \beta, V) - \begin{bmatrix} u & v & w \end{bmatrix}\right]^2 \\ L_{\text{flow}}\left(G(z)\right) &= \tfrac{1}{N}\left[h(u, v, w, V, \theta) - \begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix}\right]^2 \end{aligned} \tag{13}$$

for N training samples.

When training using Adam optimization (and most other optimization techniques), the weight update described in Equation 2 depends on $\frac{\partial L}{\partial w_t}$, the gradient of the loss function with respect to each of the weights. To ensure that the loss functions are trained collectively, the following loss function, derived from KCC, is proposed:

$$L\left(G(z), D(x)\right) = \begin{bmatrix} L_{\text{GAN}}\left(G(z), D(x)\right) \\ L_{\text{ang}}\left(G(z)\right) \\ L_{\text{vel}}\left(G(z)\right) \\ L_{\text{flow}}\left(G(z)\right) \end{bmatrix} \tag{14}$$

with the gradient defined as

$$\frac{\partial L}{\partial w_t} = \left(\frac{\partial L_{\text{GAN}}}{\partial w_t}, \frac{\partial L_{\text{ang}}}{\partial w_t}, \frac{\partial L_{\text{vel}}}{\partial w_t}, \frac{\partial L_{\text{flow}}}{\partial w_t}\right) \tag{15}$$

for Adam optimization.

V. Experimental Results

A. Data and Neural Network Setup

The two sources of data include the AirSTAR testbed results for the T2 research airframe and the NASA ICM program's UAM Flight Dynamics Vehicle-simulation environment, applied to the Lift-Plus-Cruise aircraft model.

The AirSTAR testbed provided results from an early loss-of-control study, in which an aircraft repeatedly entered and exited multiple flight envelopes[58]. From these experiments, a subset of 92 flight experiments and maneuvers over 47 test flights were stored on a dedicated MySQL server, provided by the NASA Langley Research Center (LaRC) Office of the Chief Information Officer (OCIO) MySQL server, using the schema described in Figure 3. The measurements stored for each of these flight experiments include a subset of 133 flight dynamics and control variables from the full suite of sensors, actuators, instrumentation, and controls onboard the T2 airframe. These measurements were recorded over the entire time span of the flight at consistent time intervals in the range of 50 Hz. The following flight state vector was used to train the AirSTAR GANs:

$$X_{\text{AirSTAR}} = \begin{bmatrix} V & \alpha & \beta & \gamma & p & q & r & \phi & \theta & \psi \end{bmatrix} \tag{16}$$

The moments, $\{I_{xx}, I_{yy}, I_{zz}, I_{xz}\}$, and vehicle mass, m_{T2} are available and recorded for each flight, but not included for training the network.

The Lift-Plus-Cruise flight dynamics-vehicle model captures data at 200 Hz. For these experiments, the simulation data was also resampled to 1 Hz and stored on the NASA LaRC OCIO database servers. The 6DOF-EOM variables were stored in the inertial, wind, and air relative frame of reference. For GAN experiments with this vehicle, flight states include the addition of the body-axis translational velocity components:

$$X_{\text{L+C}} = \begin{bmatrix} u & v & w & V & \alpha & \beta & \gamma & p & q & r & \phi & \theta & \psi \end{bmatrix} \tag{17}$$

As the vehicle commands are available from the simulation environment and have an expressive relationship with the variables of $X_{\text{L+C}}$, they were also included as optional elements of the GAN inputs:

$$X_{\text{L+C}_\delta} = \begin{bmatrix} u & v & w & V & \alpha & \beta & \gamma & p & q & r & \phi & \theta & \psi & \delta_{\text{e}} & \delta_{\text{a}} & \delta_{\text{r}} \end{bmatrix} \tag{18}$$

GANs for both aircraft were created as specified in IV.B, using the Python 3.7 Tensorflow 2.7 library[59]. To generate new variations of X_{AirSTAR}, two GANs were constructed: one where G was comprised entirely of feed-forward layers and one in which a Gated-Recurrent Unit Layer (GRU)[60] was introduced as the optional RNN layer from Figure 5. X_{AirSTAR} variables were extracted from 44 flights of the vehicle for training, all in which the vehicle shifted in and out of multiple flight envelopes. The base number of layers (n from Figure 5) for both networks was set to 64. For AirSTAR, the capacity to generate data using each neural network architecture was examined. For Lift-Plus-Cruise, 16 GAN variants were created and evaluated, based on optional configuration setups described throughout this paper. These configurations are summarized in Table 1.

The training data for AirSTAR was produced by a research pilot with vision displays and ground track array awareness executing a specific set of slow stall maneuvers, slowly providing a nose-up elevator command and multi-axis excitation to the control surfaces. Following each stall maneuver, the aircraft nose dropped. Following a roll maneuver, the aircraft regained speed and leveled the wings using recovery controls. These flights were roughly 20 minutes on average. In some Lift+Cruise flights labeled *Linear*, the aircraft executes a vertical takeoff, briefly hovers, and then increases velocity along the X-axis. Other flight data records show the aircraft already in flight, steadily moving along the X-axis in a trim state. The dataset for the Linear flight regime includes 27 flights that average 45 seconds per flight, in which the aircraft flies straight and there is no significant variance in its rotation. In the second flight regime, the aircraft flies with another aircraft in the simulation and is coded to automatically execute the optimal reciprocal collision avoidance algorithm (ORCA)[52]. The other aircraft approaches the Lift+Cruise on a flat plane and a maneuver is taken at a specific lookahead time. The dataset includes 37 flights, averaging 100 seconds per flight, with varied lookahead times and angles of approach, as well as a flight in which ORCA is not executed.

B. Performance: AirSTAR T2

The training for both the Feed-Forward and GRU-based networks was limited to 3000 training cycles, or *epochs*. For each epoch, each of the 44 AirSTAR T2 served as an individual batch for training. The training losses for GANs

Flight Regime	Loss Function	RNN Layer	Include Commands
Linear	Baseline (Vanilla)	None	No
ORCA	Physics-Constrained	GRU	Yes

Table 1 GAN Variants for the Lift-Plus-Cruise Vehicle Model

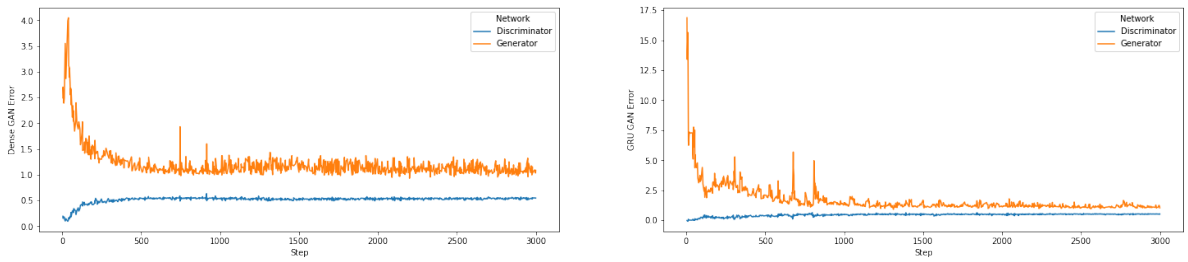


Fig. 6 AirSTAR T2 GAN Training Loss

Feed-Forward GAN (left) and GRU GAN (right) trained on 44 AirSTAR T2 flights for 3000 epochs

trained on the AirSTAR T2 model data are plotted in Figure 6. For the Feed-Forward GAN, the binary cross-entropy loss converges after 500 epochs of training. The adversarial nature of the networks is consistent with oscilating errors around their convergence for both G and D at beyond epoch 500. For the GRU GAN, the binary cross-entropy loss converges after 1500 epochs of training. The GRU GAN requires longer training cycles for convergence and returns a higher relative loss when compared to the Feed-Forward GAN. The training loss variance of the GRU GAN prior to convergence before epoch 1500 is also higher relative to the Feed-Forward GAN.

An example of the flights generated by both GANs compared to one of the original 44 flights is shown in Figure 8. A key characteristic of most of the original flights were large spikes in the values of V and ψ, which are captured in variants generated by both GANs. For 10,000 sample flights generated from the trained GANs, the 6DOF-EOM evaluation methods from Section IV.C were applied and are visualized in Figure 9. The KCC for velocity components u, v, w were not available because only the magnitude of the velocities was provided in the training dataset. For the GRU GAN, the angular velocity and attitude losses are normally distributed for all parameters. The same is true for all parameters of the Feed-Forward GAN, except ϵ_α and ϵ_β, which exhibit the smallest mean squared errors. All airflow angles exhibit the smallest errors. For the GRU, the rate errors are smaller than those of the Feed-Forward, on average. However, the airflow errors are larger.

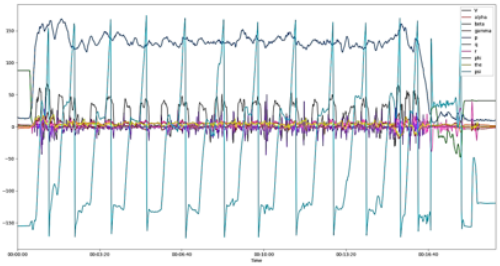
C. Performance: Lift+Cruise

Unconstrained GANs with and without vehicle commands

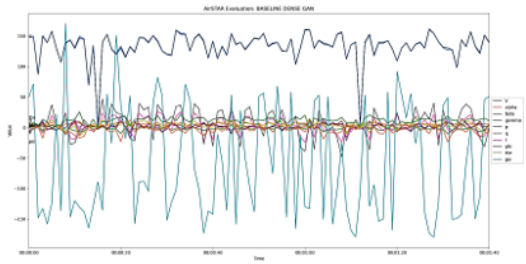
The training step for all GAN configurations that learned Lift+Cruise flight dynamics was limited to 3000 epochs. For the Linear flight cases, the average flight time series lasted for 45-seconds. So the GAN generated 45-second flights for analysis. For the ORCA flight cases, flight times averaged approximately 100 seconds. GAN experiments for ORCA thus generated 100-second flights. The training loss for GANs trained on Lift-Plus-Cruise GAN training loss from the GVS Lift-Plus-Cruise model is included in Figure 10.

The training loss pattern for the baseline Lift-Plus-Cruise GANs that were trained on Linear flights are included in Figure 10. For the Feed-Forward GANs, the training losses are relatively stable for epochs < 2000. However, for both command cases, the G loss oscillates at a higher amplitude for epochs > 2000. The command included case for the Feed-Forward G spikes with a corresponding fluctuation in D at the $[1800, 2000]$ epoch interval. For the GRU GANs, the training losses are relatively stable for the command omitted case at epochs > 500. The command included case, however, shows similar behavior to the Feed-Forward network with a divergence in the G and D training loss at epochs > 2000. The variance in the amplitude of the training loss for the G is higher for both GRU GANs when compared to their respective Feed-Forward cases.

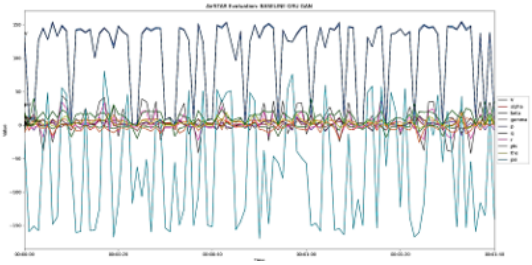
For 10,000 sample flights generated for Linear cases from fully trained GANs, the 6DOF-EOM evaluation methods were applied and visualized in Figure 12. The inclusion of commands does not consistently decrease the average loss for



(a) *Plot of $X_{AirSTAR}$ from original simulator*



(b) *Plot of $X_{AirSTAR}$ from trained Feed-Forward GAN*



(c) *Plot of $X_{AirSTAR}$ from trained GRU GAN*

Fig. 8 Original plot of AirSTAR flight states and GAN-Generated variants

Error	Distribution	Mean	Median	Dispersion
ϵ_p		0.0123161	0.0122825	0.0696694
ϵ_q		0.0150976	0.0150787	0.0539245
ϵ_r		0.0484835	0.048486	0.0370002
ϵ_α		0.00163506	4.23197e-17	2.58603
ϵ_β		2.33977e-18	2.22836e-18	∞
ϵ_γ		0.0117215	0.0116974	0.0590154

(a) *AirSTAR Feed-Forward GAN mean physics loss for 10,000 generated flights*

Error	Distribution	Mean	Median	Dispersion
ϵ_p		0.00894393	0.00891789	0.0779473
ϵ_q		0.0119946	0.0117747	0.130692
ϵ_r		0.0353929	0.0353618	0.053908
ϵ_α		0.109681	0.109662	0.311965
ϵ_β		4.73393e-17	4.65719e-17	∞
ϵ_γ		0.0165874	0.0165638	0.0612807

(b) *AirSTAR GRU GAN mean physics loss for 10,000 generated flights*

Fig. 9 AirSTAR GAN Training Loss and Physics Loss for full flights simulated using AirSTAR T2 GAN Model

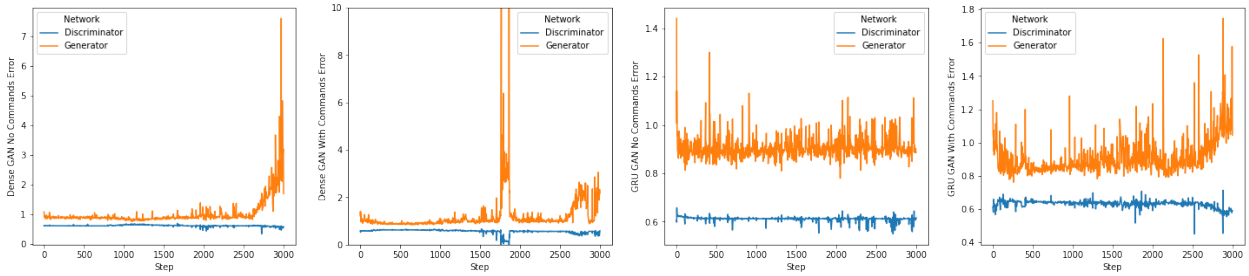


Fig. 10 L+C training loss for baseline linear flight cases

Feed-Forward GAN with (Left-Center) and without (Left) commands and GRU GAN with (Right) and without (Right-Center) commands for the baseline GAN architectures

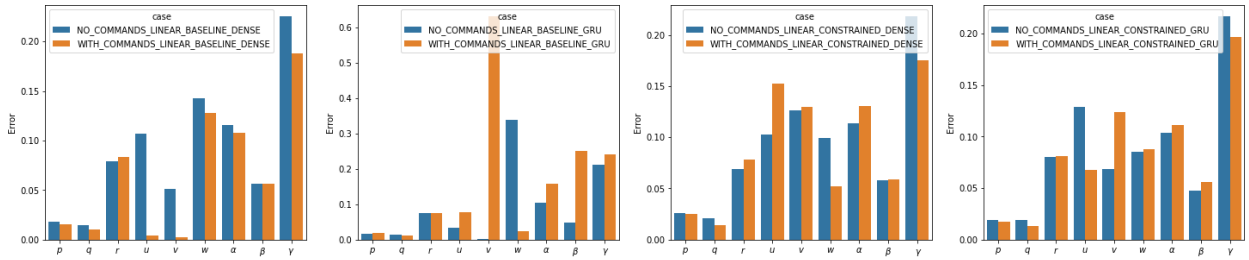


Fig. 12 L+C mean physics loss for 10,000 generated 45 second linear flights

each physics evaluation metric.

Constrained GANs without vehicle commands

The training loss for the constrained Lift-Plus-Cruise Linear and ORCA models all converged quicker than their respective baselines as depicted in Figure 13. The G loss that was influenced by both the binary cross-entropy and physics loss converged during training within the first 40 epochs for all constrained variants without commands.

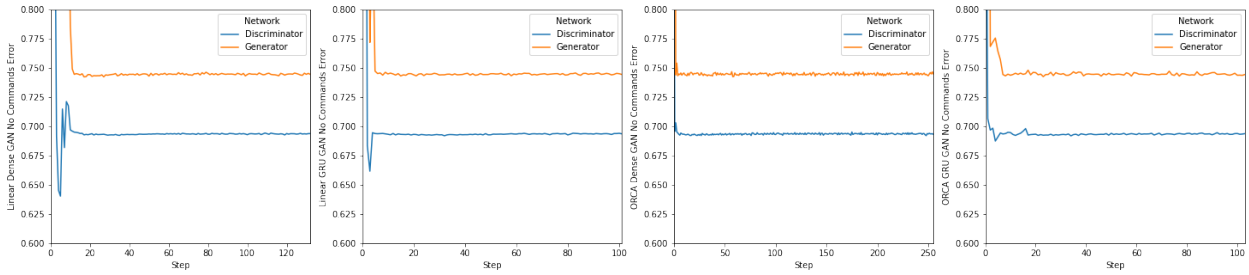
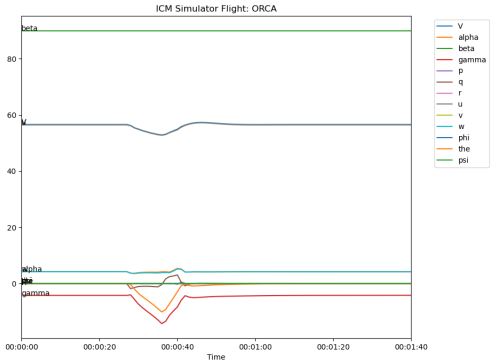


Fig. 13 L+C training loss for constrained linear and ORCA flight cases

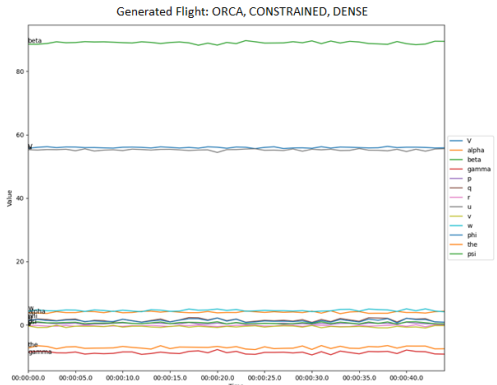
Linear Feed-Forward GAN with (Left-Center) and without (Left) GRU layers and ORCA Feed-Forward GAN with (Right) and without (Right-Center) GRU layers for the constrained GAN architectures

An example of the flights generated by both ORCA GAN variants compared an original ORCA flight is shown in Figure 15. The resulting physics losses derived from the 6DOF-EOM applied to 10,000 generated Linear and ORCA flights are depicted in Figure 16. The raw loss metrics for the angular velocity & attitudes are used, and for translational velocity the min-max scaled values are shown. The error values for p, q, r are similar for both Feed-Forward and GRU configurations for linear and ORCA generated flights. The angular velocity errors for u, v, w are relatively large for u and w consistently across all constrained cases. The attitude errors for ϕ, θ, γ show a consistent error at the 1×10^{-2} order of magnitude for all flight cases. The error in γ is larger relative to α, β.

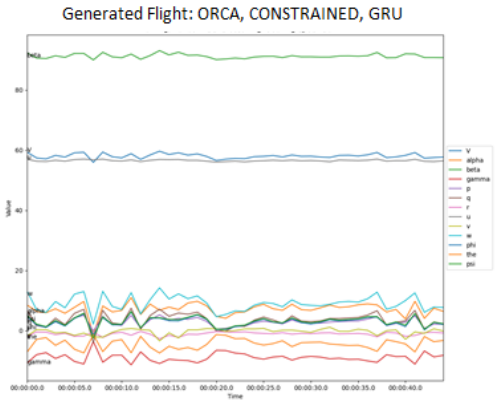
The total runtime required to generate 10,000 Linear and ORCA flights running 45 seconds and 100 seconds respectively is depicted in Figure 18. In comparison to the MATLAB Simulink model, the average runtime is ~ 50 seconds for a single linear flight lasting 15 seconds. The runtime analysis in 18 demonstrates a 1×10^2 magnitude increase in speed and volume of data generation for each respective flight condition.



(a) *Plot of X_{L+C} from original simulator*



(b) *Plot of X_{L+C} from trained Constrained Feed-Forward GAN*



(c) *Plot of X_{L+C} from trained Constrained GRU GAN*

Fig. 15 Original plot of L+C flight states and GAN-Generated variants

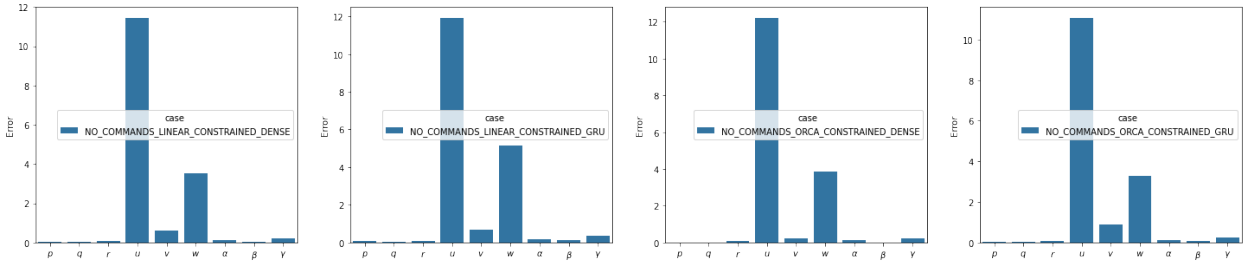
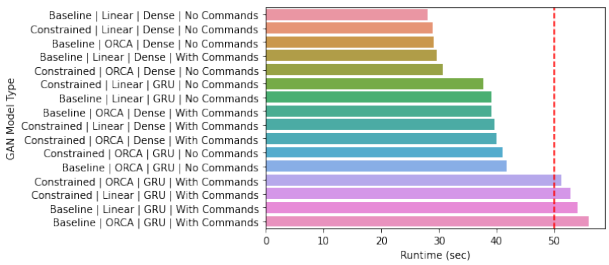
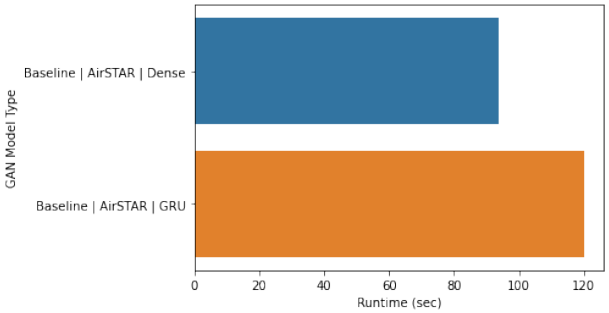


Fig. 16 L+C mean physics loss for 10,000 generated ORCA flights ($t_{ORCA} = 100$) and Linear flights ($t_{Linear} = 45$)

Angular velocity (p, q, r) loss units in $\frac{rad}{sec}$, translational velocity (u, v, w) loss is unitless due to min-max scaling, and attitude (ϕ, θ, γ) loss in deg



(a) *Total runtime for GAN generation of 10,000 experimental flights for Linear and ORCA flight cases*



(b) *Total runtime for GAN generation of 10,000 experimental flights for AirSTAR flight cases*

Fig. 18 Runtime analysis for the Lift-Plus-Cruise (Left) and AirSTAR (Right) T2 Flight Experiments

The runtime analysis for both are measured in seconds and account for the total time required to generate 10,000 flights for each respective condition. Linear flights were 45-seconds in length, ORCA flights were 100-seconds in length, and AirSTAR flights were 1000-seconds in length.

VI. Discussion

The following is a summary of the impact of the architectural choices on these experiments:

- Introducing a bias vector consisting of the means across each flight variable allowed the training process for each GAN to converge quickly, and resulted in values of the generated flight variables to be in realistic ranges.
- Including commands in the training dataset did not reliably improve the kinematic consistency across all flight variables. The commands will likely contribute to variational data augmentation when forces and moments are added to the state vector, and their corresponding kinematic equations are learned by the networks.
- Including the physical constraints in the GAN loss function requires significantly longer training times to evaluate the physics at each time step. In addition, the generator and discriminator approach Nash equilibrium time and time again before the loss functions spike because the physical constraints have yet to be minimized. Other approaches for incorporating physical constraints into the loss function should be integrated with the 6DOF-EOM that were specified here[61].

For AirSTAR commercial transport data generation using GANs, errors in generating kinematically consistent angles and rates can be reduced by introducing more layers to both the generator and discriminator. Nonetheless, depending on the use case, the results shown in this paper could sufficiently augment data size and richness for specific applications, such as designing networks that predict the onset of loss-of-control. In practical terms, imagine the Physics Evaluator from Figure 2 requires a threshold of $1\frac{\text{deg}}{s}$ or 1 deg on the rates and airflow angles, respectively, for the database to accept augmentations of the generic commercial aircraft T2 model. Recall that 1 deg $\approx$ 0.0174533rad. Based on errors described in Figure 9, most rejections of flights synthesized by the Feed-Forward GAN would be due KCC checks on r. The GRU-GAN would receive rejections based on KCC checks on both r and α. Even with these rejections, a dataset based on these flights could be sufficiently augmented, as the coefficient of dispersion for each of these variables would yield enough flights within the acceptable range.

The performance of generating L+C vehicle dynamics was less consistent than that of AirSTAR. The flights that were used for training data during L+C experiments fell into a much broader class of flight regimes than that of the training data from the AirSTAR maneuvers. The activity that occurs during a hover maneuver will likely require this flight regime to be separated, necessitating training separate GANs for each. For ease of learning and future experimentation, the database will need to properly classify these categories.

Overall, the results do show that GANs are a feasible reduced-order model for the flight dynamics variables considered in this study. And constrained networks with RNN layers are able to produce more reliable results over longer training times while simpler feed-forward networks provide reasonable results over short training times. However, further characterizing the error for these GANs will need to be considered in future studies. Describe the impact of the results. This methodology does not propose that ϵ from Equation 11 will be exactly zero. When fitted to this vector, the original datasets have error components that range from 10^{-35} to 10^{-5}. The physics evaluation step should therefore capture this error information from the original datasets to establish an acceptable threshold for inserting a generated flight into the database.

VII. Conclusion

This study used a traditional GAN model to interpret the capacity of the architecture to augment a flight dynamics dataset. The GAN model demonstrated the capacity to generate kinematically consistent flight dynamics data over multiple flight regimes. A traditional GAN, or even one constrained by physics equations, is unlikely to be the final architecture used to augment the ICM flight dataset. A variety of GANs exist for operations that can have other applications in the aerodynamics domain. BigGAN[44, 62] generators are the current state-of-the-art, taking into account extreme diversity in images and producing extremely high-resolution, high-quality images. Once trained, BigGANs are the likely choice for NASA ICM program synthesis of data to capture all necessary elements in our high-fidelity simulations. Speed experiments will need to occur in order to validate its utility in augmenting a dataset in-place of running the simulation itself. A deep convolutional GAN (DCGAN) is an extension of the GAN concept that uses convolutional and convolutional-transpose layers in the discriminator and generator, allowing translational invariance and feature extraction to become an inherent part of the architecture[42, 43]. This kind of architecture can be used to train on longer flights, learning entire maneuvers or flight regimes and placing them at various points in the synthesized flight. Much like previous studies demonstrating prediction of LOC using CVAE, applying conditions to synthetic data generation using conditional GANs[41, 63, 64] can become a trivial method of learning various known regimes of flight dynamics, such as LOC conditions, failure modes, or flight envelope violations using the same architectural model. In operation, this would allow a single model to synthesize data under all known flight regimes.

Such an outcome would be very attractive in creating a model that would enable efficient testing of corner cases, driven by flight dynamics, for evaluating autonomous vehicle flight and associated decision making efficacy.

Appendix

The data that need to be stored for the Intelligent Contingency Management (ICM) program are described in Tables 2 and 3.

Category	Description	# Variables
Flight Scenario	Directory for outputting binary outputs	1
Flight Scenario	Cruise, Maneuver, Hover, Variable, Full Flight	1
Flight Scenario	Simulation Units	44
Flight Scenario	Environment-Specific Parameters	38
Flight Scenario	Categories of Control, Vehicle Type, Atmospheric Modeling, Turbulence Modeling, Sensors, Propeller, Output Formats, and Optimization Functions	16
Flight Scenario	Switches for Binary Environmental Options	11
Flight Scenario	Vehicle Specifications (Model Type, Number of Engines, Number of Surfaces, etc.)	4
Vehicle Parameters	Initial Conditions for EOM	69
Vehicle Parameters	Trim Conditions	112
Vehicle Parameters	Control Parameters	34
Vehicle Parameters	Reference Input Specification and Parameters	38
	Total:	368

Table 2 Description of simulation input data that need to be stored for the Intelligent Contingency Management program

Category	Physical Category	Reference Frame	Physical Measurement	# Variables
Environment	Wind	NED	Translational Velocities and Accelerations	6
Environment	Turbulence	Body	Translational/Angular Velocities and Accelerations	6
Vehicle	Surface Actuator	Body	Position and Rates	10
Vehicle	Propeller Actuator	Body	Engine Speeds and Accelerations	18
Vehicle	Force-Moments	Body	Total Forces, Moments, Angular Momentums	12
Vehicle	Force-Moments	Body	Aerodynamic Forces and Moments	6
Vehicle	Force-Moments	Body	Vehicle Propulsion Forces, Moments, Angular Momentums	12
Vehicle	Force-Moments	Body	Propeller Forces, Moments, Total Angular Momentums	36
Vehicle	Gear		Gear	1
Vehicle	Equations of Motion	Inertial	Positions, Velocities, Accelerations, Gravitational Accelerations, Quaternions, Angular Velocities, Angular Rates	22
Vehicle	Equations of Motion	World-Relative	Positions, Velocities, Accelerations, Quaternions, Angular Velocities (Body and NED), Velocities	28
Vehicle	Equations of Motion	Air-Relative	Velocities, Accelerations, Angular Velocities (Body and NED), Quaternions	17
Vehicle	Equations of Motion	Sensor	Positions, Velocities, Accelerations, Quaternions, Angular Velocities, GPS (Lat-Lon-Alt), Gamma, Chi	22
Commands	Engine Controls		Engine Commands	9
Commands	Engine Controls		Engine Power	9
Commands	Surface Controls		Ailerons, Elevators, Rudder Commands	5
Commands	Surface Controls		Ailerons, Elevators, Rudder Power	5
Vehicle Failure	Surface Actuators		Initiate, Hold, Positions, Rates, Accelerations, Signal	95
Vehicle Failure	Engine Actuators		Initiate, Hold, Positions, Rates, Accelerations, Signal	159
Reference Inputs			Reference positions, velocities, chi, and chi-dot	8
			Total:	486

Table 3 Description of simulation output data that need to be stored for the Intelligent Contingency Management program

Acknowledgments

This research was supported by the NASA Aeronautics Research Mission Directorate (ARMD), Transformative Tools and Technologies (TTT) project, under the Autonomous Systems / Intelligent Contingency Management subproject. The T-2 flight data used was generated by the NASA Langley Research Center AirSTAR team under the NASA Aviation Safety Program, Vehicle Systems Safety Technologies (VSST) project. Additional thanks to the NASA Langley Research Center MidRange Computer User's Group for supporting experimental processing and analysis on the K-cluster.

References

[1] Neogi, N. A., and Holbrook, J., *Creating formal characterizations of routine contingency management in commercial aviation*, ???? https://doi.org/10.2514/6.2021-1116, URL https://arc.aiaa.org/doi/abs/10.2514/6.2021-1116.

[2] Gregory, I. M., Neogi, N. A., Grauer, J. A., Campbell, N. H., Holbrook, J., Bacon, B. J., Murphy, P. C., Moerder, D. D., Simmons, B. M., Acheson, M. J., Britton, T. C., and Cook, J., *Intelligent Contingency Management for Urban Air Mobility*, ???? https://doi.org/10.2514/6.2021-1000, URL https://arc.aiaa.org/doi/abs/10.2514/6.2021-1000.

[3] Haehnel, R. B., Wissink, A. M., George, G., Hardin, D., and Fegyveresi, J., *Automation Tools for Rotorcraft Analysis on High-Performance Computing Centers*, ???? https://doi.org/10.2514/6.2020-1756, URL https://arc.aiaa.org/doi/abs/10.2514/6.2020-1756.

[4] Haehnel, R. B., Christensen, S. D., Whitlow, J. L., Bauer, A. C., Meyer, A., Rangarajan, G., Wenren, Y., Hardin, D. L., Hoch, B., Clark, S., and Eiseman, A., *A Computational Prototyping Environment interface for DoD CREATE^{TM}-AV Helios Simulations*, ???? https://doi.org/10.2514/6.2021-0945, URL https://arc.aiaa.org/doi/abs/10.2514/6.2021-0945.

[5] Zhang, M., Melin, T., Gong, J., Barth, M., and Axner, L., "Mixed Fidelity Aerodynamics and Aero-Structural Optimization for Wings," *2018 International Conference on High Performance Computing Simulation (HPCS)*, 2018, pp. 476–483. https://doi.org/10.1109/HPCS.2018.00081.

[6] Paul, V. G., Periyasamy, S., and Nikam, K., "Unsteady aerodynamics analysis for small amplitude pitch oscillations of transonic cruiser aircraft," *2017 First International Conference on Recent Advances in Aerospace Engineering (ICRAAE)*, 2017, pp. 1–5. https://doi.org/10.1109/ICRAAE.2017.8297231.

[7] Zhang, M., Gong, J., Axner, L., and Barth, M., "Automation of High-Fidelity CFD Analysis for Aircraft Design and Optimization Aided by HPC," *2020 28th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, 2020, pp. 395–399. https://doi.org/10.1109/PDP50117.2020.00067.

[8] Shao, S., Wang, P., and Yan, R., "Generative adversarial networks for data augmentation in machine fault diagnosis," *Computers in Industry*, Vol. 106, 2019, pp. 85–93. https://doi.org/https://doi.org/10.1016/j.compind.2019.01.001, URL https://www.sciencedirect.com/science/article/pii/S0166361518305657.

[9] Hsu, W.-N., Zhang, Y., and Glass, J., "Unsupervised domain adaptation for robust speech recognition via variational autoencoder-based data augmentation," *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2017, pp. 16–23. https://doi.org/10.1109/ASRU.2017.8268911.

[10] Islam, Z., Abdel-Aty, M., Cai, Q., and Yuan, J., "Crash data augmentation using variational autoencoder," *Accident Analysis and Prevention*, Vol. 151, 2021, p. 105950. https://doi.org/https://doi.org/10.1016/j.aap.2020.105950, URL https://www.sciencedirect.com/science/article/pii/S000145752031770X.

[11] Calimeri, F., Marzullo, A., Stamile, C., and Terracina, G., "Biomedical Data Augmentation Using Generative Adversarial Neural Networks," *Artificial Neural Networks and Machine Learning – ICANN 2017*, edited by A. Lintas, S. Rovetta, P. F. Verschure, and A. E. Villa, Springer International Publishing, Cham, 2017, pp. 626–634.

[12] Chen, J., Lu, C., Chenli, B., Zhu, J., and Tian, T., "VFlow: More Expressive Generative Flows with Variational Data Augmentation," *Proceedings of the 37th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 119, edited by H. D. III and A. Singh, PMLR, 2020, pp. 1660–1669. URL https://proceedings.mlr.press/v119/chen20p.html.

[13] Navidan, H., Moshiri, P. F., Nabati, M., Shahbazian, R., Ghorashi, S. A., Shah-Mansouri, V., and Windridge, D., "Generative Adversarial Networks (GANs) in networking: A comprehensive survey and evaluation," *Computer Networks*, Vol. 194, 2021, p. 108149. https://doi.org/https://doi.org/10.1016/j.comnet.2021.108149, URL https://www.sciencedirect.com/science/article/pii/S1389128621002139.

[14] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., "Generative adversarial nets," *Advances in neural information processing systems*, Vol. 27, 2014.

[15] Jordan, T., Foster, J., Bailey, R., and Belcastro, C., *AirSTAR: A UAV Platform for Flight Dynamics and Control System Testing*, ???? https://doi.org/10.2514/6.2006-3307, URL https://arc.aiaa.org/doi/abs/10.2514/6.2006-3307.

[16] Silva, C., Johnson, W. R., Solis, E., Patterson, M. D., and Antcliff, K. R., *VTOL Urban Air Mobility Concept Vehicles for Technology Development*, ???? https://doi.org/10.2514/6.2018-3847, URL https://arc.aiaa.org/doi/abs/10.2514/6.2018-3847.

[17] Loiseau, J.-C., Noack, B. R., and Brunton, S. L., "Sparse reduced-order modelling: sensor-based dynamics to full-state estimation," *Journal of Fluid Mechanics*, Vol. 844, 2018, pp. 459–490.

[18] Loiseau, J.-C., and Brunton, S. L., "Constrained sparse Galerkin regression," *Journal of Fluid Mechanics*, Vol. 838, 2018, pp. 42–67.

[19] Brunton, S. L., Proctor, J. L., and Kutz, J. N., "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences*, Vol. 113, No. 15, 2016, pp. 3932–3937. https://doi.org/10.1073/pnas.1517384113, URL https://www.pnas.org/content/113/15/3932.

[20] Tri, N. C., Duong, H. N., Van Hoai, T., Van Hoa, T., Nguyen, V. H., Toan, N. T., and Snasel, V., "A novel approach based on deep learning techniques and UAVs to yield assessment of paddy fields," *2017 9th International Conference on Knowledge and Systems Engineering (KSE)*, 2017, pp. 257–262. https://doi.org/10.1109/KSE.2017.8119468.

[21] Basermann, A., "HPC Software Infrastructures at German Aerospace Center," *The International Conference for High Performance Computing, Networking, Storage, and Analysis 2018 (SC18)*, 2018. URL https://elib.dlr.de/125496/.

[22] Garcia, J. A., Melton, J., Schuh, M. J., James, K., Long, K., Vicroy, D. D., Deere, K. A., Luckring, J. M., Carter, M. B., Flamm, J. D., et al., "NASA ERA Integrated CFD for Wind Tunnel Testing of Hybrid Wing-Body Configuration," *54th AIAA Aerospace Sciences Meeting*, 2016, p. 0262.

[23] Colonius, T., *An overview of simulation, modeling, and active control of flow/acoustic resonance in open cavities*, ???? https://doi.org/10.2514/6.2001-76, URL https://arc.aiaa.org/doi/abs/10.2514/6.2001-76.

[24] Kerstens, W., Pfeiffer, J., Williams, D., King, R., and Colonius, T., "Closed-Loop Control of Lift for Longitudinal Gust Suppression at Low Reynolds Numbers," *AIAA Journal*, Vol. 49, No. 8, 2011, pp. 1721–1728. https://doi.org/10.2514/1.J050954, URL https://doi.org/10.2514/1.J050954.

[25] Shmilovich, A., Yadlin, Y., and Whalen, E. A., "Active Flow Control Computations: From a Single Actuator to a Complete Airplane," *AIAA Journal*, Vol. 56, No. 12, 2018, pp. 4730–4740. https://doi.org/10.2514/1.J056307, URL https://doi.org/10.2514/1.J056307.

[26] Jordan, P., and Colonius, T., "Wave Packets and Turbulent Jet Noise," *Annual Review of Fluid Mechanics*, Vol. 45, No. 1, 2013, pp. 173–195. https://doi.org/10.1146/annurev-fluid-011212-140756, URL https://doi.org/10.1146/annurev-fluid-011212-140756.

[27] Dupuis, R., Jouhaud, J.-C., and Sagaut, P., "Surrogate Modeling of Aerodynamic Simulations for Multiple Operating Conditions Using Machine Learning," *AIAA Journal*, Vol. 56, No. 9, 2018, pp. 3622–3635. https://doi.org/10.2514/1.J056405, URL https://doi.org/10.2514/1.J056405.

[28] Lui, H., and Wolf, W., *Convolutional Neural Networks for the Construction of Surrogate Models of Fluid Flows*, ???? https://doi.org/10.2514/6.2021-1675, URL https://arc.aiaa.org/doi/abs/10.2514/6.2021-1675.

[29] Campbell, N. H., Grauer, J. A., and Gregory, I. M., *Loss of Control Detection for Commercial Transport Aircraft Using Conditional Variational Autoencoders*, ???? https://doi.org/10.2514/6.2021-0778, URL https://arc.aiaa.org/doi/abs/10.2514/6.2021-0778.

[30] Wilborn, J., and Foster, J., *Defining Commercial Transport Loss-of-Control: A Quantitative Approach*, ???? https://doi.org/10.2514/6.2004-4811, URL https://arc.aiaa.org/doi/abs/10.2514/6.2004-4811.

[31] Pol, A. A., Berger, V., Germain, C., Cerminara, G., and Pierini, M., "Anomaly Detection with Conditional Variational Autoencoders," *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, 2019, pp. 1651–1657. https://doi.org/10.1109/ICMLA.2019.00270.

[32] Campbell, N. H., Grauer, J. A., and Gregory, I. M., "Use of Design of Experiments in Determining Neural Network Architectures for Loss of Control Detection," *2021 IEEE Aerospace Conference (50100)*, 2021, pp. 1–20. https://doi.org/10.1109/AERO50100.2021.9438231.

[33] Rothhaar, P. M., Murphy, P. C., Bacon, B. J., Gregory, I. M., Grauer, J. A., Busan, R. C., and Croom, M. A., "NASA Langley distributed propulsion VTOL tiltwing aircraft testing, modeling, simulation, control, and flight test development," *14th AIAA aviation technology, integration, and operations conference*, 2014, p. 2999.

[34] Clarke, M., Smart, J., Botero, E. M., Maier, W., and Alonso, J. J., *Strategies for Posing a Well-Defined Problem for Urban Air Mobility Vehicles*, ???? https://doi.org/10.2514/6.2019-0818, URL https://arc.aiaa.org/doi/abs/10.2514/6.2019-0818.

[35] Kim, K., Rahili, S., Shi, X., Chung, S.-J., and Gharib, M., *Controllability and Design of Unmanned Multirotor Aircraft Robust to Rotor Failure*, ???? https://doi.org/10.2514/6.2019-1787, URL https://arc.aiaa.org/doi/abs/10.2514/6.2019-1787.

[36] Gregory, I. M., Campbell, N. H., Neogi, N. A., Holbrook, J. B., Grauer, J. A., Bacon, B. J., Murphy, P. C., Moerder, D. D., Simmons, B. M., Acheson, M. J., et al., "Intelligent contingency management for urban air mobility," *International Conference on Dynamic Data Driven Application Systems*, Springer, 2020, pp. 22–26.

[37] Silva, C., Johnson, W., Solis, E., Patterson, M., and Antcliff, K., "VTOL Urban Air Mobility Concept Vehicles for Technology Development," 2018. https://doi.org/10.2514/6.2018-3847.

[38] editor (ed.), *A Strip Theory Approach to Dynamic Modeling of EVTOL Aircraft.*, ????

[39] Simmons, B. M., Buning, P. G., and Murphy, P. C., "Full-Envelope Aero-Propulsive Model Identification for Lift+Cruise Aircraft Using Computational Experiments," ????

[40] Acheson, M. J., Gregory, I. M., and Cook, J., *Examination of Unified Control Incorporating Generalized Control Allocation*, ???? https://doi.org/10.2514/6.2021-0999, URL https://arc.aiaa.org/doi/abs/10.2514/6.2021-0999.

[41] Mirza, M., and Osindero, S., "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[42] Radford, A., Metz, L., and Chintala, S., "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[43] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X., "Improved techniques for training gans," *Advances in neural information processing systems*, Vol. 29, 2016, pp. 2234–2242.

[44] Brock, A., Donahue, J., and Simonyan, K., "Large scale GAN training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018.

[45] Yang, L., Zhang, D., and Karniadakis, G. E., "Physics-informed generative adversarial networks for stochastic differential equations," *SIAM Journal on Scientific Computing*, Vol. 42, No. 1, 2020, pp. A292–A317.

[46] Xu, P., and Karamouzas, I., "A GAN-Like Approach for Physics-Based Imitation Learning and Interactive Control," *Proc. ACM Comput. Graph. Interact. Tech.*, Vol. 4, No. 3, 2021. https://doi.org/10.1145/3480148, URL https://doi.org/10.1145/3480148.

[47] Date, C. J., *Database design and relational theory: normal forms and all that jazz*, Apress, 2019.

[48] "MySQL 8.0 Reference Manual :: 15.6.2.2 the physical structure of an innodb index," , 2021. URL https://dev.mysql.com/doc/refman/8.0/en/innodb-physical-structure.html.

[49] Bayer, R., and McCreight, E., "ORGANIZATION AND MAINTENANCE OF LARGE ORDERED INDICES," Tech. rep., BOEING SCIENTIFIC RESEARCH LABS SEATTLE WA MATHEMATICAL AND INFORMATION . . . , 1970.

[50] Comer, D., "Ubiquitous B-Tree," *ACM Comput. Surv.*, Vol. 11, No. 2, 1979, p. 121–137. https://doi.org/10.1145/356770.356776, URL https://doi.org/10.1145/356770.356776.

[51] Krogh, J. W., Krogh, G., and Gennick, *MySQL Connector/Python Revealed*, Springer, 2018.

[52] Van Den Berg, J., Guy, S. J., Lin, M., and Manocha, D., "Reciprocal n-body collision avoidance," *Robotics research*, Springer, 2011, pp. 3–19.

[53] Nash, J. F., et al., "Equilibrium points in n-person games," *Proceedings of the national academy of sciences*, Vol. 36, No. 1, 1950, pp. 48–49.

[54] Kingma, D. P., and Ba, J., "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[55] Clevert, D.-A., Unterthiner, T., and Hochreiter, S., "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.

[56] "Scipy.spatial.transform.RotationSpline - Smooth Attitude Interpolation¶," , Mar 2019. URL https://github.com/scipy/scipy/files/2932755/attitude_interpolation.pdf.

[57] Shuster, M. D., et al., "A survey of attitude representations," *Navigation*, Vol. 8, No. 9, 1993, pp. 439–517.

[58] Jordan, T., and Bailey, R., "NASA Langley's AirSTAR Testbed: A Subscale Flight Test Capability for Flight Dynamics and Control System Experiments," 2008. https://doi.org/10.2514/6.2008-6660.

[59] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," , 2015. URL https://www.tensorflow.org/, software available from tensorflow.org.

[60] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[61] Raissi, M., Perdikaris, P., and Karniadakis, G., "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, Vol. 378, 2019, pp. 686–707. https://doi.org/https://doi.org/10.1016/j.jcp.2018.10.045, URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[62] Lučić, M., Tschannen, M., Ritter, M., Zhai, X., Bachem, O., and Gelly, S., "High-fidelity image generation with fewer labels," *International conference on machine learning*, PMLR, 2019, pp. 4183–4192.

[63] Antipov, G., Baccouche, M., and Dugelay, J.-L., "Face aging with conditional generative adversarial networks," *2017 IEEE international conference on image processing (ICIP)*, IEEE, 2017, pp. 2089–2093.

[64] Yu, C., and Zhang, Z., "Painting on placement: Forecasting routing congestion using conditional generative adversarial nets," *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.