Simple Pattern Traffic Generation for Automated Flight Research in Non-Towered Traffic Patterns

Daniel R. Hill* and Andrew P. Patterson† and Irene M. Gregory‡
NASA Langley Research Center, Hampton, VA, 23681, USA

Research efforts into autonomous air traffic will necessitate tools for testing algorithm capabilities. Testing will require flexible methods for creating large quantities of artificial data to verify the safety of automated systems. The traffic generation method in this work was developed to test traffic prediction and replanning algorithms for an autonomous vehicle attempting to land at a non-towered airport. The traffic generation method produces airport approach trajectories supporting a wide range of pattern entry types and typical pattern modification maneuvers for multiple aircraft types with varying performance capabilities. For each aircraft there are options for choosing the approach type, modifying how the approach is flown, and imposing scenario-driven temporal constraints, such as spacing between pairs of aircraft. The tool uses simplified aircraft dynamics to produce position and velocity profiles for traffic vehicles. Additionally, the tool supports standalone simulation tests or batch/bulk testing efforts, multiple output data options, and facilitates post-processing analysis.

Nomenclature

c	=	geometric chord along an arc
D	=	distances in feet
dt	=	time step
f, g	=	generic functions
$\overrightarrow{\mathrm{v}}$	=	vector notation
$\overrightarrow{\mathrm{v}}.x$	=	vector component notation $(x\|y\|z)$
$\hat{\mathrm{u}}$	=	unit vector notation
h	=	height / altitude in feet
h_{ref}	=	altitude above the local tangent plane reference point in feet
P	=	current position of the aircraft along the horizontal plane in feet
ROT	=	rate of turn in degrees per second
α_{T}	=	angle of horizontal path turn in degrees
ϕ	=	bank angle in degrees
GS	=	glideslope in degrees
R	=	turn radius in feet
V	=	airspeed in knots

I. Introduction

Unmanned aviation is rapidly approaching merger with the manned aerospace world. To facilitate that future, research into how unmanned vehicles will integrate within these existing human-centric systems is ongoing. One goal is to develop algorithms that will enable unmanned aerial vehicles (UAVs) to functionally participate in terminal area patterns at non-towered airfields. Without the structure that a managed airspace provides, there is an increased risk of miscommunication, misunderstandings, and unsafe landing conditions. A testing tool that efficiently simulates large number of aircraft in relevant scenarios for this environment will be critical for developing everything from operational requirements to UAV Traffic Pattern Integration (TPI) algorithms.

*Lead Software Developer, Analytical Mechanics Associates, Inc., daniel.r.hill@nasa.gov
†Researcher for Navigation Guidance & Control Systems, andrew.patterson@nasa.gov, AIAA Member
‡Senior Technologist for Advanced Control Theory and Applications, irene.m.gregory@nasa.gov, AIAA Fellow

To develop the algorithms and tools, researchers need access to a large set of test aircraft flight path histories. It is also crucial to have the flexibility to adjust various parameters to handle any number of testing and validation scenarios. The Simple Pattern Traffic (SPT) tool, presented in this work, is a collection of algorithms that allow the generation of such scenarios.

The primary concept behind this tool is to mimic the behavior of a fixed-wing aircraft entering non-towered airspace and landing at a runway. This behavior is designed to appear as a reasonable proxy without providing high-fidelity flight dynamics. The tool is built to operate under the auspices of the Federal Aviation Administration's (FAA) guidelines on general traffic patterns. It provides access to a variety of generic approach types based on the legs of a traffic pattern and specific researcher requests[1]. The aircraft following these patterns, along with its performance capabilities, can be selected to explore a range of behaviors. This paper covers the SPT tool and its capability set.

The SPT algorithm has been developed to test and benchmark autonomy algorithms in traffic perception, intent prediction, and trajectory replanning. A description of these problems can be found in Refs. [1, 2]. A prediction method is studied using the SPT algorithm for traffic in Ref. [3]. Several different collision avoidance and optimal replanning methods are being tested on scenarios where traffic is generated using the SPT algorithm. These replanning methods can be found in Refs. [4–7]. Other approaches to the prediction and replanning problem can be found in Refs. [8, 9], which focus on machine learning methods.

II. Background

A. Traffic Patterns

In non-towered airspace, there is an established pattern (Ref. [10]) for traffic making approach and landing on the runway (Fig. 1). The pattern altitude, speed, and width are roughly set by aircraft type and therefore performance. Upon entering the pattern airspace each aircraft is expected to fly at an approximate traffic altitude for the pattern and to fly within an expected pattern airspeed range until beginning final descent. The aircraft will typically begin its final descent when it is abeam the runway threshold, and conduct a turn onto the base leg when approximately 45° abeam from the runway threshold.

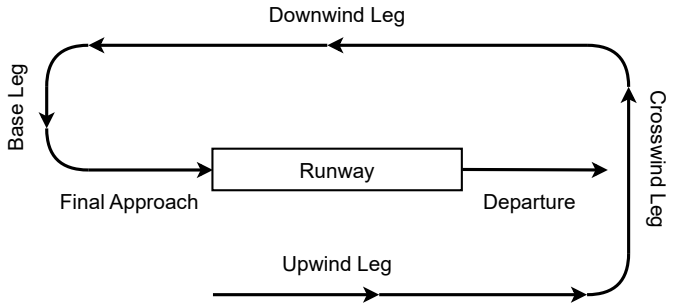


Fig. 1 Generic non-towered airport traffic pattern diagram.

The pattern flow is a rectangular motion around the target runway[2]. There are five main legs of the pattern: final, base, downwind, crosswind, and upwind. Traffic entering pattern airspace can enter the pattern on any of these legs. Therefore, SPT provides options to support entry on any leg. See Appendix A for examples of each pattern entry type. Furthermore, there are some common changes to a pattern that may occur on approach, e.g., extending the downwind leg, turning to base leg early, turning to base leg late, and aborting landing into a go-around.

SPT models the environment using East-North-Up (ENU) coordinate representation, where the local tangent plane is established with a reference point that, in this work, is placed at the runway threshold. With this, all SPT aircraft will converge on the reference point to land and subsequently terminate operations[3]. This also makes constructing pattern geometries more intuitive, because FAA materials discuss patterns in terms of distances relative to the runway and

[1] E.g., entering a pattern directly on a downwind leg vs. a standard 45° entry.

[2] SPT supports both left-handed and right-handed pattern directions.

[3] The reference point is often put slightly above the physical runway.

approach end point. Similarly, with a local tangent plane and a reference point, commonly used methods for converting to geodetic coordinates can be applied for integration with visualizations or other testing environments.

B. Geometric / Kinematic Equations

The following section covers the key background mathematical preliminaries that drive much of SPT's dynamics. In order to simplify turn dynamics, it is assumed that there will be no airspeed changes during turn phases[4].

1. Turn Angle

The value of upcoming turns along the horizontal path plane are determined using the scalar product of state vectors. It is assumed the turn angles never exceed ±180°. The vectors are directions or flight path vectors between the way-points involved (Fig. 2). Using this information, we can calculate our turn angle along the lateral path (α_{T}) as follows:

$$\alpha_{\mathrm{T}} = \operatorname{acos} \frac{\overrightarrow{AB} \cdot \overrightarrow{BC}}{\|\overrightarrow{AB}\|\|\overrightarrow{BC}\|} \tag{1}$$

where $\overrightarrow{\mathrm{AB}}$ is a vector from current aircraft horizontal position to the target waypoint or position, and $\overrightarrow{\mathrm{BC}}$ is a vector from the upcoming waypoint (or position) to the subsequent waypoint (or position).

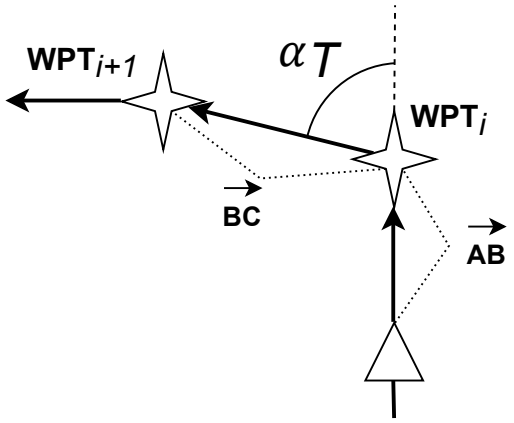


Fig. 2 Overview of how turn angle α_{T} is calculated.

2. Turn Factor

The turn factor metric is a value that signals if an upcoming turn angle represents a left turn, right turn, or no turn at all. This value is important for making minor axis / value adjustments being used in the underlying functions. Given that the relevant turn state vectors ($\overrightarrow{AB}$ and $\overrightarrow{BC}$) and α_{T}, which was calculated in Eq. (1), the turn factor itself can be calculated by:

$$\overrightarrow{F_T} = \overrightarrow{AB} \times \overrightarrow{BC} \tag{2}$$

where both vectors exist in the horizontal plane with zero in the vertical component. The resultant vector F_T will have only a value in the third component. This component, labeled $\overrightarrow{F_T}.z$, can be evaluated to determine the turn direction thusly:

$$\text{Turn Direction } (z) = \begin{Bmatrix} \text{To the right,} & \text{if } z < 0 \\ \text{Parallel; No turn,} & \text{if } z = 0 \\ \text{To the left,} & \text{if } z > 0 \end{Bmatrix} \tag{3}$$

[4]This is assumed to simplify the dynamics rooted in the turn equations used from the FAA Pilot's Handbook, but trades off some realism on speed profiles.

3. Rate of Turn

The expected rate of a turn for an aircraft given a bank angle and an airspeed is determined by a formula provided by the FAA (Ref. [11]). This value can be calculated with:

$$\text{ROT} = \frac{1091 \tan(\phi)}{V}, \tag{4}$$

where the constant 1091 is the FAA provided conversion factor for a rotational rate in degrees per second, given angles (ϕ) in degrees and speeds (V) in knots.

4. Bank Angle

An aircraft's bank angle can be determined from airspeed and rate of turn by rearranging the formula used in Eq. 4 to:

$$\phi = \text{atan2}(V * \text{ROT}, 1091) \tag{5}$$

5. Turn Radius

Using the calculated bank angle and known airspeed of the aircraft, the turn radius for an upcoming turn, using an FAA Pilot's Handbook formula (Ref. [11]), is calculated as follows:

$$R = \frac{V^2}{11.26 \tan(\phi)}, \tag{6}$$

where the constant 11.26 is the FAA provided conversion factor for a turn radius in feet, given angles in degrees and speeds in knots.

6. Along Path Distance In and Out of Waypoints

When handling turns, SPT makes calculations to determine the ideal point along the path to start and ideally end an upcoming turn. To determine this, the previously calculated turn information creates a circular arc (Fig. 3). Using that arc, a chord c from the start of turn to the end of turn can be used in combination with the law of cosines to acquire the along path distance that represents the minimum distance needed to start and complete the turn. We can use the law of cosines this way by assuming that the along path distance in is the same as the along path distance out. The chord length

$$c = \sqrt{2R^2 * (1 - \cos(\alpha_\text{T}))} \tag{7}$$

can in turn be used to find the along path distances of the circular turn arc

$$D_\text{AP} = \sqrt{\frac{c^2}{2(1 - \cos(180^\circ - \alpha_\text{T}))}} \tag{8}$$

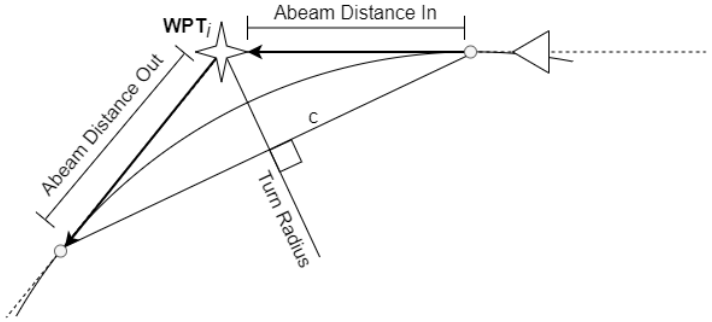


Fig. 3 Overview of a simple turn along a circular arc.

7. Flight Path Angle Approach Distance Projection

As SPT creates flight plans, it has to build outward from the runway when determining where to place the initial position of the aircraft. To accomplish this, an approximation is made utilizing a negative 3° flight path angle[5] out to the airspace operating limit, which is 10 Nm for SPT[6]. This metric will give roughly a 3:1 ratio for altitude per nautical mile change[7] (Ref. [12]). This gives us the following:

$$D_{\text{fromFPA}} = \frac{\Delta h_{\text{ref}}}{C_{\text{gs2ft}} * \text{GS}} * C_{\text{Nm2ft}} \tag{9}$$

where C_{Nm2ft} is a conversion constant equaling 6076.12 ft per nautical mile and C_{gs2ft} is a conversion constant for the number of vertical feet change per nautical mile per degree of flight path angle, which is set to 100.

8. Bounding Box Intersection

During flight data generation, SPT runs checks in certain cases to make sure the aircraft is not slipping too far off course along the path. In order to determine if the aircraft is within an ideal spatial horizontal region, the SPT runs a bounding box test using a rectangle formed between four spatial points with points (A, B, C, D). The region is drawn either clockwise or counterclockwise. This assures that the lengths between A $\rightarrow$ B and A $\rightarrow$ D always represents the length and width edges of the rectangle, respectively. Given the nature of this tool, the rectangular region is configured with two horizontal path waypoints and applied lateral offsets orthogonal to the flight path (e.g., a waypoint ±1000 ft cross path). The boolean test is conducted by evaluating:

$$\text{Intersects}^{8} = \begin{array}{l}((\overrightarrow{AP} \cdot \overrightarrow{AB}) \geq 0) \wedge ((\overrightarrow{AP} \cdot \overrightarrow{AB}) \leq (\overrightarrow{AB} \cdot \overrightarrow{AB})) \wedge \\ ((\overrightarrow{AP} \cdot \overrightarrow{AD}) \geq 0) \wedge ((\overrightarrow{AP} \cdot \overrightarrow{AD}) \leq (\overrightarrow{AD} \cdot \overrightarrow{AD}))\end{array} \tag{10}$$

where $\overrightarrow{AP}$, $\overrightarrow{AB}$, $\overrightarrow{AD}$ are vectors from bounding box point A to the current aircraft position P and bounding box points B and D, respectively.

III. Implementation

SPT execution consists of two main processes: (1) the creation of a configuration and (2) running the configuration through the flight dynamics. While in practice these steps are often done in tandem, they are kept separate to allow flexibility. Early in development, it proved useful during testing to run a configuration once while also retaining the ability to re-run the flight dynamics during debugging sessions or creating different output products.

A. Configuration

The first step to generating flight data with SPT involves obtaining appropriate configuration data items. This data is best acquired by running an accompanying tool named `configSPTScenario`, but it is possible for seasoned users to replicate what `configSPTScenario` does and construct the necessary items themselves. `configSPTScenario` itself leverages a number of parameters for tailoring the scenario configuration[9], such as number of aircraft, aircraft types, runway specification, and pattern entry methods. The aircraft type, in particular, will directly affect how the underlying trajectory generation routines resize the traffic pattern[10] and set speed targets. The resultant scenario configuration contains a series of structures, where each structure is composed of an aircraft identifier, type classification, start time, and a flight plan. The flight plan itself is a structure composed of a series Cartesian positions and speed targets. Each aircraft requested will have one of these configuration structures.

[5]This value was an arbitrary selection based on general aviation practices when a fixed-wing aircraft is on approach.

[6]This value was arbitrarily chosen by project researchers, but can be adapted to other distances.

[7]Follows common piloting rule of thumb named "Rule of Three".

[8]In Eq. 10, $\wedge$ represents the logical AND operation; not the bitwise XOR operation.

[9]SPT's User Manual contains a robust list of the current command line options for scenario configuration; available with source release.

[10]Previously discussed in the Background section II.

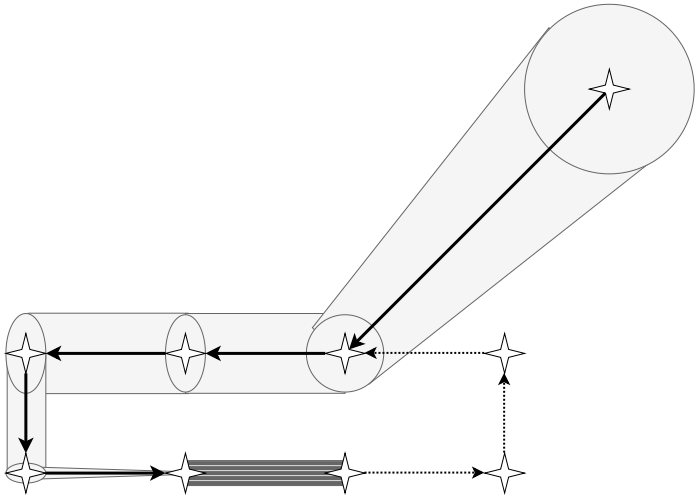


Fig. 4 Horizontal variance bounds in a standard 45° pattern entry.

The underlying baseline paths of the flight plans generated for pattern entry can be nominal based on aircraft performance. To create more varied output, a feature to apply variance was implemented. The variance model operates on the three separate aspects of the flight profile: horizontal, vertical, and speed. Horizontal and vertical variance place waypoints within certain radii (sampled from a random normal distribution) offset from the nominal waypoint location. For instance, in the case of most starting points, the (x, y) or (E, N) coordinate is given an offset for (E, N) that is ±150% for the current traffic pattern width[11]. Speed variance applies a speed target modifier for the target speed of a waypoint (sampled from a random normal distribution). For example, in most cases when traveling along any pattern legs at traffic pattern speed, the model applies a sample value that is within ±15% of the current traffic pattern speed. Any combination of the three can be enabled. The range of the sampling for variance constricts as the plan converges upon the runway threshold; i.e., ±150% at start, ±15% to ±10% on most pattern legs, and the 5% on final. Figure 4 shows how the variance model is applied to the horizontal path geometries during a standard 45° pattern entry. The offset variance from the nominal flight path is applied to each waypoint in a given leg, which effectively creates a tube for each major leg operation. Vertical variance behaves similarly, but prohibits ascending above the most recent waypoint altitude (ensuring that the approach is a smooth, consistent descent). Speed variance behaves like the vertical variance model, limiting the variance to the most recent speed to create smooth deceleration curves.

B. Flight Generation

After the configuration routine `configSPTScenario` has been run or the appropriate data has been generated directly, the user proceeds to execute the tool `RunSimSPT`, which leverages SPT's core dynamics engine to mimic flight. The process iterates over each aircraft, generating a full mimicked flight data set at once, and then conducts any requested post-processing, e.g., temporal spatial assurance and analysis plots. Each aircraft flight history generated by SPT acts as a standalone flight and does not factor possible interactions with other traffic within a larger scenario. In such cases, SPT will ignore any possible conflicts as though they do not exist. It is expected that users manipulate the resultant data directly or leverage several builtin post-processing features to construct more advanced situations.

[11] Currently, the variance range values are not easily customized by the user. They would have to directly modify the code.

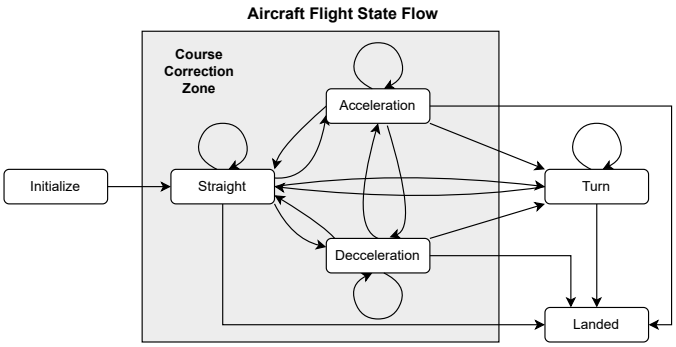


Fig. 5 State diagram of an aircraft in SPT's dynamics.

While processing a flight, the dynamics follow a state machine style of operation (Fig. 5). Each aircraft begins at an initial state and then proceeds into straight and level flight. From here, SPT processes the flight along both horizontal and vertical profiles during each state. While conditions met can affect state transitions in both horizontal and vertical modes of flight, they still maintain their own physical state and logical processing. During each phase iteration, any logical determinations based on the previous physical state are conducted first, such as detecting a forthcoming turn or detecting a need for course correction. Once these are determined, SPT processes the physical changes for this step along the horizontal path and then the vertical path. After physical state changes are made along the horizontal path, there is another logical check for state changes that now exist given the physical state changes. The dynamics then iterate to the next time step, and continues until the aircraft has landed or maximum allowed processing time has been reached[12].

In the cases of turns, once a turn is about to be initiated, SPT determines the necessary flight performance data that is allowed: bank angle, turn rate, turn degrees, turn radius, and turn center. This data is then used in the turn to slowly change the aircraft position along the flyable circular arc per time step. A check is also made on the aircraft's position change against a logical waypoint gate (based on the turn center and turn radius) to see if the next planned waypoint has been sequenced. Upon exiting a turn, there is a check to see if a course correction must be made to account for any overshoots on the turn. Future development aims to include guidance and safety checks against creating infeasible and "unflyable" situations.

Upon completion of an SPT aircraft flight, there are several optional post-processing features that can be used. First, there is active spacing assurance. SPT can take a given spatial time value specified during configuration and apply it to the time data for each aircraft's flight history to assure a minimal temporal separation assurance at landing time. Aircraft are sorted in order of arrival, and then a gap is applied by shifting the start times to all aircraft where their landing time violates the spacing restriction. The time shift is cumulative, which means the final aircraft in a string of landings could have its start time shifted significantly.

Second, specific landing times or landing intervals can be applied to aircraft. Much like spacing assurance, these values are specified during configuration and are intended as a device to force specific scenario situations, e.g., pattern spacing violations. Both of these options, like before, adjust the start time for the flight history to accommodate the final landing times. One option sets a specific landing time (e.g. 31 s), the other allows for a randomly sampled landing time chosen from desired range of times (i.e., 345 s is chosen from a range of 330 s to 390 s).

Flight data histories are by default returned from `RunSimSPT`; being deposited in the base workspace. However, there is also an option that extracts data from the flight histories and puts them into a series of structures containing time, position, and velocity arrays sampled at 1 Hz (one structure per aircraft flight). This option is appealing for users that do not require the larger set of data items contained in a flight history, e.g., turn information, state tracking, waypoint tracking, or course correction tracking.

Several graphics can also be generated as end products of the flight generation phase. There are summary charts of the horizontal and vertical flight profiles, a chart of the speed profile, debug charts that show operation critical metrics like current aircraft states and turn data, and a landing timeline. These charts are all optional and can also be directly accessed via the SPTPlot library class. Said class contains several useful rendering tools such plotting overlapping flight profiles for horizontal, vertical, and speed paths, the generated flight plans from the `configSPTScenario` tool, and individual data items from the structure making up the larger SPT data history.

[12]The maximum processing time defaults to 1800 seconds, but its a configurable parameter.

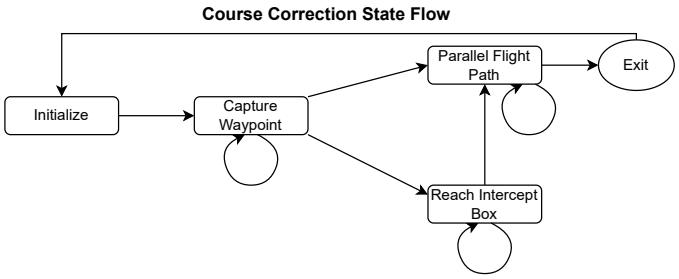


Fig. 6 State flow of course correction algorithm

1. Course Correction

Course correction is a necessary measure implemented to account for regular variations that can occur in the dynamics due to time step, vehicle performance, and off-nominal trajectory perturbations added by variance models during configuration. It acts as its own state machine underneath the main state machine; see Fig. 6 for a breakdown of the states and Alg. 1 for pseudocode on the process. When the vehicle enters a non-turn flight segment, there is a check to see if the next flight waypoint is reachable given the current track. This is important since the primary way to track changes is through turn segments. If an inability to capture is detected, the course correction algorithm engages and runs in parallel with the normal flight algorithm (Fig. 7).

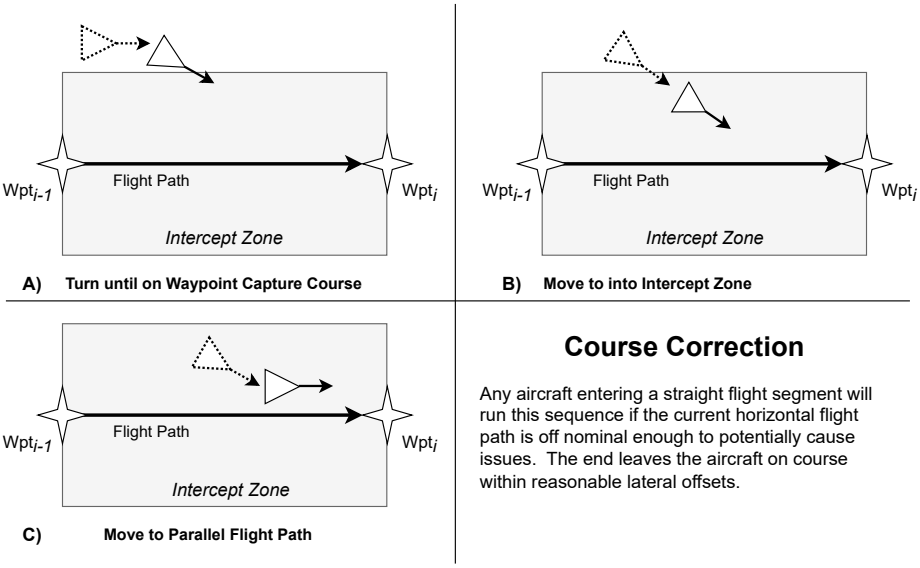


Fig. 7 Visual layout of course correction state sequence.

The aircraft begins making a small turn (1° per second)[13] in order to set a track that allows a capture of the waypoint. Once the turn has achieved capture, the algorithm checks to see if the aircraft has already entered the capture bounding box or "Intercept Zone" (as in part B of Fig. 7) that is constructed by the upcoming waypoint, previous way-point, and a reasonable horizontal offset abeam each waypoint. If the aircraft has yet to enter this zone (determined using the bounding box test (Eq. (10))), the aircraft continues along the current track until this is accomplished. Once the aircraft is within the interception zone, the aircraft conducts another 1° turn to the opposite direction of the initial correction

[13]In cases of high track offset, this turn speed is increased; e.g., 2°/s for >5° and 3°/s for >10° offsets.

turn[14]. This turn continues until the aircraft's velocity vector $\overrightarrow{v}$ is parallel with the flight path vector $\overrightarrow{FP}$.

Algorithm 1: General course correction algorithm

Input : current velocity v, flight path vector $\overrightarrow{FP}$, Upcoming waypoint wpt, previous CourseCorrectState (Default is Initialize)

Output : updated current velocity v, updated CourseCorrectState)

```
function CourseCorrection;
Determine unit vectors: v̂, F̂P, ŵpt;
if Current course (v̂) does not capture next wpt then
    /* Aircraft requires slight turn adjustment to assure the next waypoint is
       captured.                                                              */
    if CourseCorrectState == Initialize then
        /* Entering Course Correction for the first iteration                 */
        Set CourseCorrectState to CaptureWaypoint;
    end
    switch CourseCorrectState do
        case CaptureWaypoint do
            Progress Course Correct Turn;
            Run a CrossingTest using v̂, v̂new, F̂P;
            if v̂new hasCrossed or is parallel to ŵpt then
                if P intersects Capture Box then
                    Set CourseCorrectState to ParallelFlightPath;
                else
                    Set CourseCorrectState to ReachInterceptZone;
                end
            end
        case ReachInterceptZone do
            if P intersects Capture Box then
                Set CourseCorrectState to ParallelFlightPath;
            else
                Continue Straight Flight;
            end
        case ParallelFlightPath do
            Progress Course Correct Turn;
            Run a CrossingTest using v̂, v̂new, F̂P;
            if v̂new == F̂P then
                End Course Correction; Desired Path Attained;
            else if v̂new hasCrossed F̂P then
                End Course Correction; Snap to F̂P;
            end
    end
    Update v with any changes using v̂new;
end
function CrossingTest(v1, v2, v_target);
return ((v1 × v_target) * (v2 × v_target) < 0);
```

[14]Degree rates of the course correction turns are not user configurable.

2. Waypoint Sequencing

As the aircraft progresses in its flight along the given flight plan, SPT tests to detect a sequencing or capture of the current waypoint target. The initial version of the logic utilized a simplistic distance check using an arbitrary radius from the waypoint as an origin. This logic alone proved insufficient, as aircraft will fail to make waypoints and break their approach along otherwise manageable flight plans.

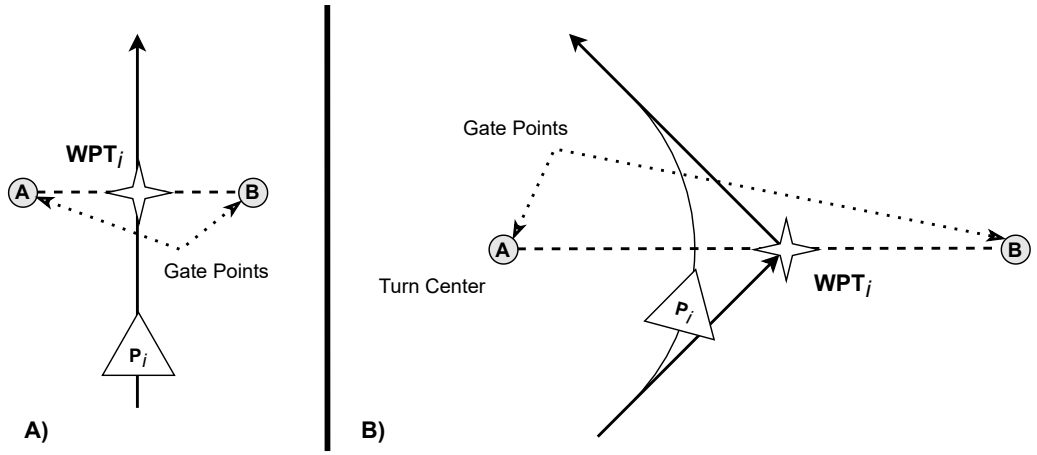


Fig. 8 A) How a logical gate is setup around a waypoint during a straight flight segment B) How a logical gate is setup up during a turn segment.

To address this issue, vector mathematics was employed to create a complex logical gate between two points of arbitrary distance (Fig. 8A), where A and B represent the logical gate points. In practice, the gate point would be placed along the expected flight path with enough distance to allow horizontal deviations without causing failures. However, in the case of turns (Fig. 8B), the gate points are placed at the center of turn and two times the distance from the waypoint to center of the turn.[15]

[15]The distances for a turn are set to well beyond the waypoint to accommodate very shallow turns (mostly), where it was possible to be mid turn and on the opposite side of the waypoint from the turn center point. Most cases see the ownship between the turn center point and waypoint.

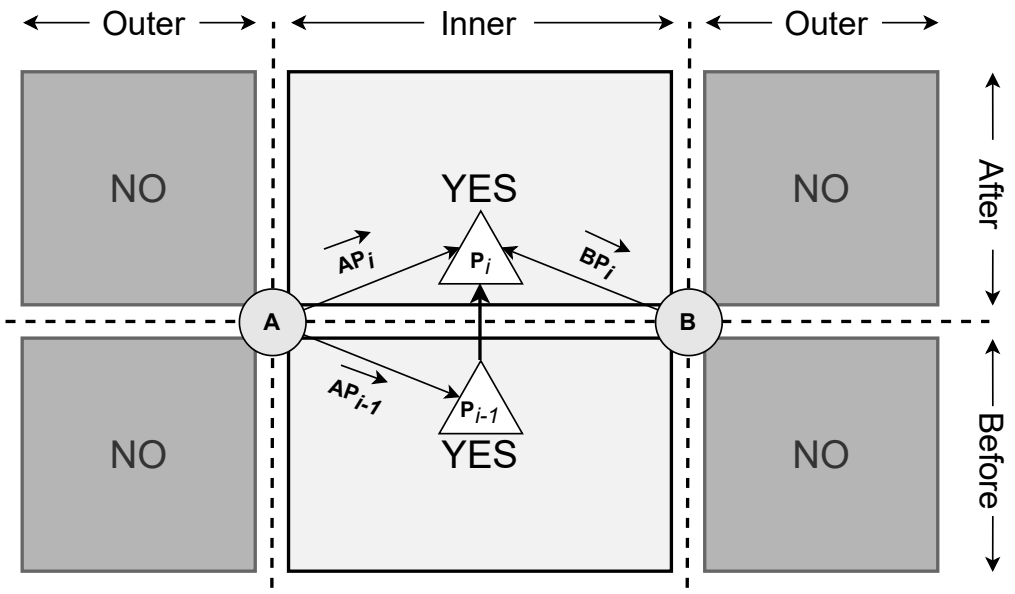


Fig. 9 Layout of testing sectors and the key vectors / positions for logical gate crossing.

Fig. 9 shows how the testing is done to accomplish the waypoint sequencing, which is further broken down in Alg. 2. The method tests the aircraft's before position and after position to determine in which of the six sectors each position resides. If any of the four outer sectors are detected, the sequencing fails because the aircraft cannot cross outside of the gate points (A and B). A gate crossing is detected if the position change has gone from a "before" inner sector to an "after" inner sector. Furthermore, it is possible to have positions end up parallel with the gate. In that case, it was decided to detect a crossing only if the "old" position was found to be in parallel with the gate. If the new position is in parallel, then it really has not completed a crossing. Other implementations could choose to do this differently, but this approach was the most logical given SPT's needs.

Algorithm 2: Way-point sequencing along the horizontal path

Input : Current Position P_i, Last Position P_{i-1}, nominal flight path vector $\overrightarrow{FP}$, Upcoming waypoint WP
Output : Flag if the Aircraft crossed the gate since last position change $(true, false)$

function WaypointGateCheck;
Use WP to get 2 gate points (A and B) orthogonal to nominal flight path.;
Calculate Gate crossing vectors $\overrightarrow{AB}$ and $\overrightarrow{BA}$. ;
Calculate vectors to positions $\overrightarrow{AP_i}$, $\overrightarrow{AP_{i-1}}$, and $\overrightarrow{BP_i}$. ;
Test scalar products to new position P_i to see if aircraft has moved outside gate bounds.;
if $((\overrightarrow{AB} \cdot \overrightarrow{AP_i}) < 0)$ ***or*** $((\overrightarrow{BA} \cdot \overrightarrow{BP_i}) < 0)$ **then**
    return false;
end
/* Get the cross products so we can use the 3rd component of each vector to test direction. */
Calculate cross product $\overrightarrow{CP_{new}}$ between the Gate ($\overrightarrow{AB}$) and the new Position P_i ($\overrightarrow{AP_i}$).;
Calculate cross product $\overrightarrow{CP_{old}}$ between the Gate ($\overrightarrow{AB}$) and the old Position P_{i-1} ($\overrightarrow{AP_{i-1}}$).;
Test R where R is $\overrightarrow{CP_{new}}.z * \overrightarrow{CP_{old}}.z$.;
/* Note the explicit use of the z or third component of the cross product vector results. */
if $R < 0$ **then**
    return true;
else if $R > 0$ **then**
    return false;
else
    /* Test for a position vector of length 0. */
    if $\|\overrightarrow{AP_{i-1}}\|$ *or* $\|\overrightarrow{AP_i}\|$ **then**
        return false;
    end
    /* One of the positions is parallel with the Gate $\overrightarrow{AB}$ */
    if $\overrightarrow{AP_{i-1}}$ *is parallel* **then**
        return true;
    else if $\overrightarrow{AP_i}$ *is parallel* **then**
        return false;
    end
end

IV. Performance

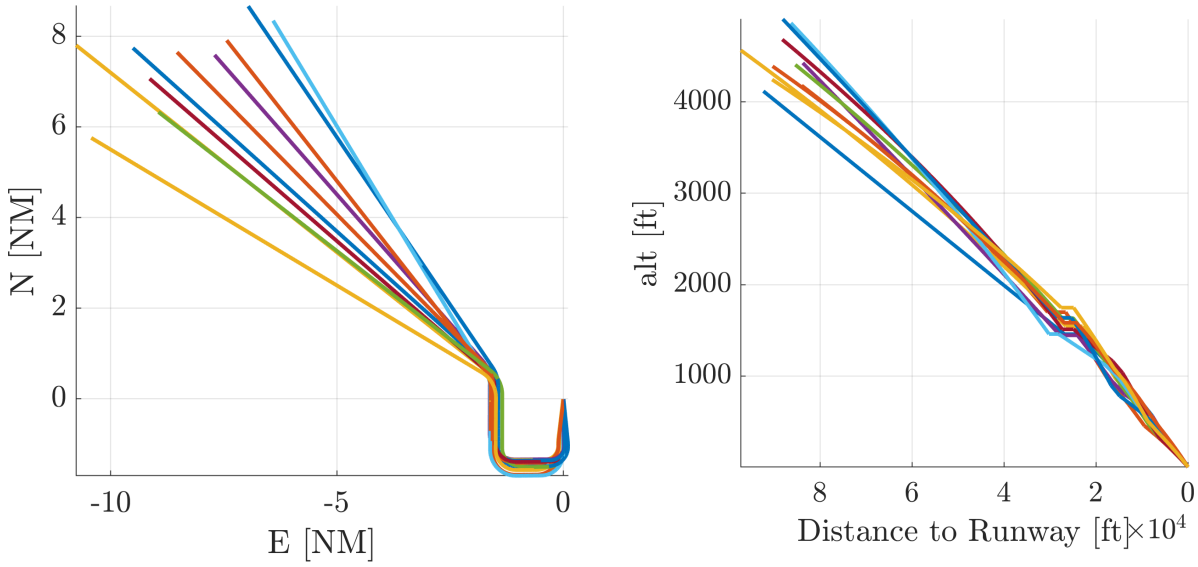
SPT provides desired output quickly, with each run of an aircraft completed with 1-2 seconds. Even relatively long flights, such as a full teardrop approach with an abort-go-around, returns results on a moderate laptop within 3-4 seconds. However, process time is not a significant factor for what SPT provides because it is a preprocessing step.

In terms of data fidelity as a proxy for real flight data, SPT's profiles lean to the more nominal flight shapes. The variance algorithm creates profiles that are not duplicates of the baseline flight path but do not deviate significantly behavior-wise from what is expected from observation. The reason for this is two-fold: (1) to give researchers profiles that are unambiguous as they vet and refine their own algorithms and (2) to reduce the likelihood of SPT entering a flight state of undefined behavior[16].

Figures 10 and 11 are examples of a batch run of a single aircraft with horizontal and vertical variance enabled. Figure 10 depicts an aircraft approaching the notional runway using the standard 45° pattern entry. As can be seen in

[16]SPT does not check for flight path feasibility. It gives as much power and risk to the researcher. User selected criteria could lead to unexpected behavior due to infeasible flight paths, e.g., landing a 737 class at a 2300 ft landing strip

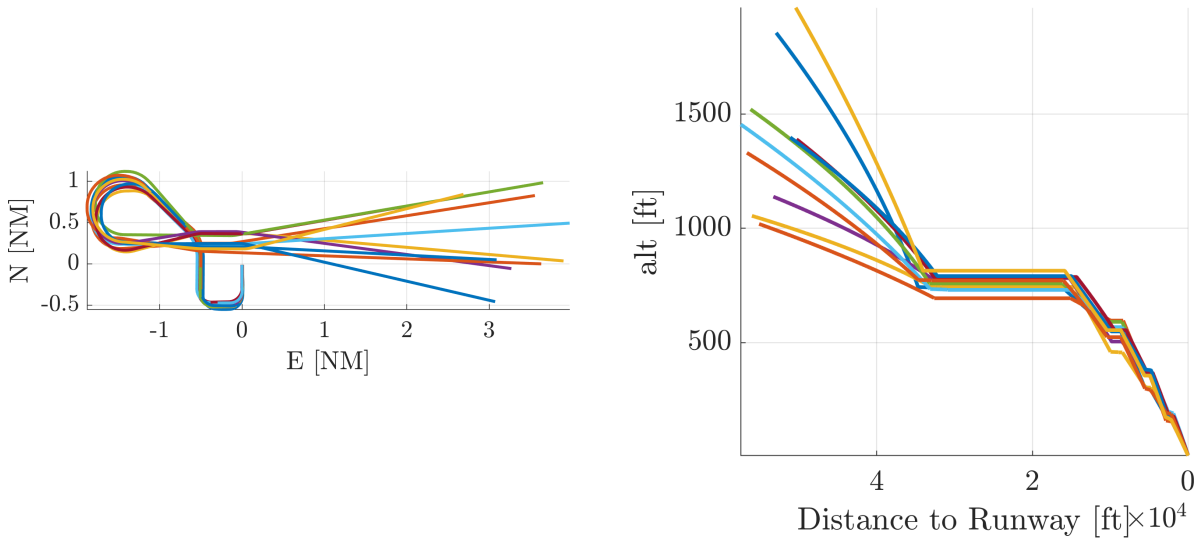
both parts of Fig. 10 and Fig. 11, the initial points along both the lateral and vertical profiles apply the most variance to the expected flight approach pattern. Once the aircraft joins the pattern, on the downwind leg in the case of the standard 45° approach as seen in Fig. 10, the variance is still present but converges. This is in line with much of the expectations of real world aircraft as they get closer to their final descent and landing.



Lateral Profile **Vertical Profile**

Fig. 10 Ten aircraft flights using a standard 45 pattern entry approach.

The second example below shows a midfield cross approach(Fig. 11). In this case, the aircraft is moving across the pattern, roughly crossing above the middle of the destination runway at a pattern crossing altitude (slightly above normal pattern altitude), clears the pattern laterally, and conducts a rough procedure turn, which then allows the aircraft to enter the downwind leg at a rough 45° angle. Normal pattern operations continue on from there. The horizontal profile in Fig. 11 is a good visual example of how the minor variances applied to the legs can still yield some meaningful alterations from the nominal flight pattern.



Lateral Profile **Vertical Profile**

Fig. 11 Ten aircraft flights using a teardrop entry approach.

SPT is currently being used to provide pattern traffic at a non-towered airport in a validation scenario testing suite for algorithms exploring TPI solutions, being able to fulfill a range of performance needs, e.g. multiple aircraft sizes, pattern entry types, and potential aircraft conflict situations. In addition, a separate study was done to explore range of flight data variety that SPT provides a researcher looking at a single scenario. Figure 12 shows a heatmap of all available pattern entry approaches run by a series of aircraft. The volume of approach options shown highlights the key feature of SPT, expansive and flexible options.

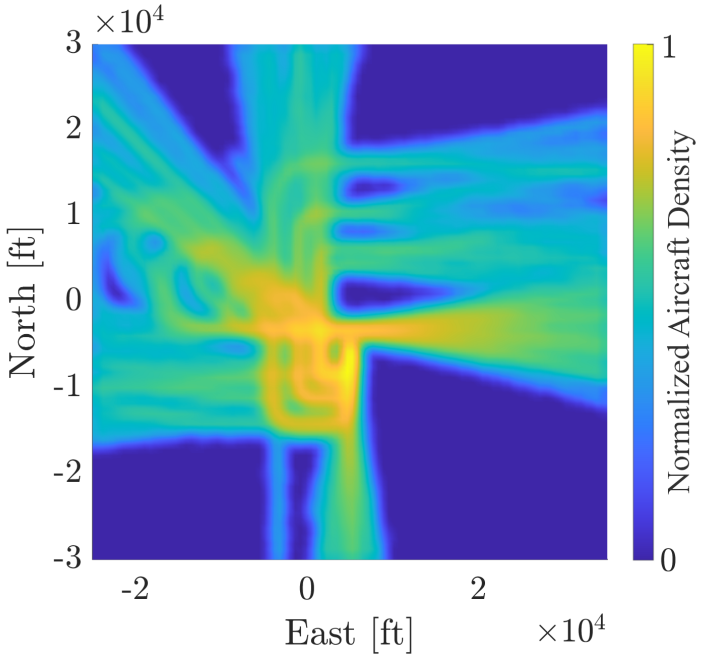


Fig. 12 Heatmap showing the density of approaches for SPT Pattern generation when variance options are enabled.
(560 total flight plans)

V. Summary

The Simple Pattern Traffic toolset has enabled NASA Langley researchers to generate usable proxies of aircraft approach profiles. These profiles enable the ongoing research into novel unmanned aircraft terminal area pattern integration technologies. The software provides flexibility to create a range of custom performance aircraft flying into real or experimental airspace, and engage with these airspace categories with multiple approach patterns and modifications. The software operates in modular components for individual configuration and execution of the mimicked flight dynamics, which allows researchers and developers flexibility as they work. Researchers can even leverage the components to conduct bulk flights. Future work will allow more scenarios, performance types, temporal constraints, and robust performance.

VI. Appendix

Example Flight Trajectories

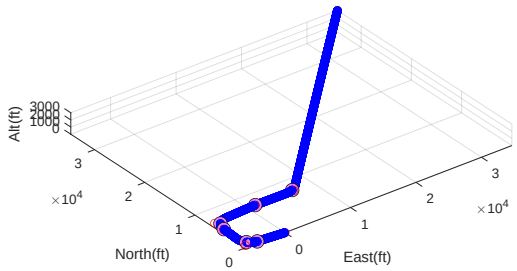


Fig. 13 Flight on standard 45 entry.

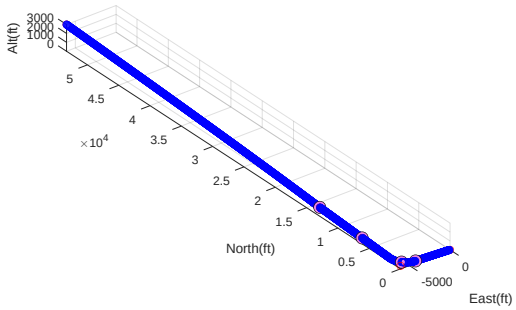


Fig. 14 Flight on base entry.

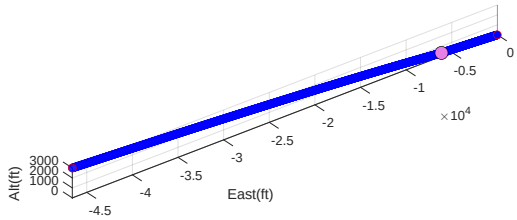


Fig. 15 Flight on final entry.

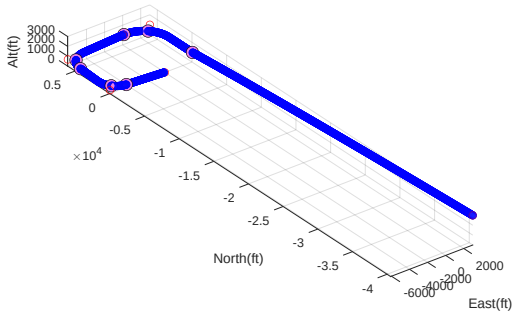


Fig. 16 Flight on midfield cross entry.

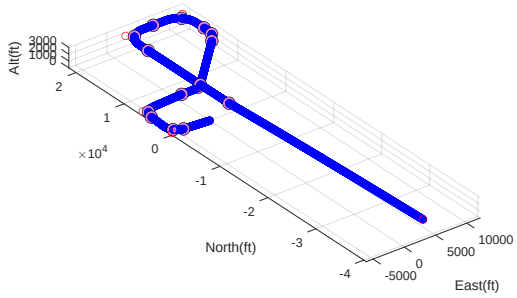


Fig. 17 Flight on a standard teardrop entry.

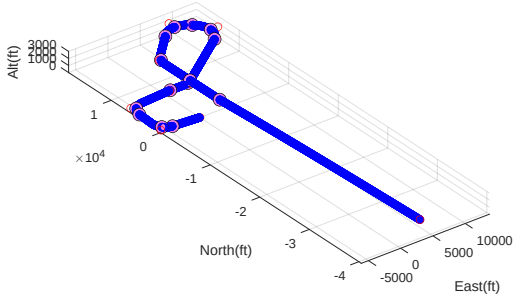


Fig. 18 Flight on a tight teardrop entry.

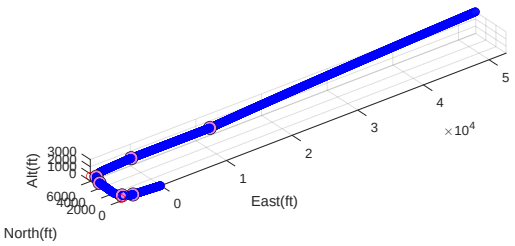


Fig. 19 Flight on downwind entry.

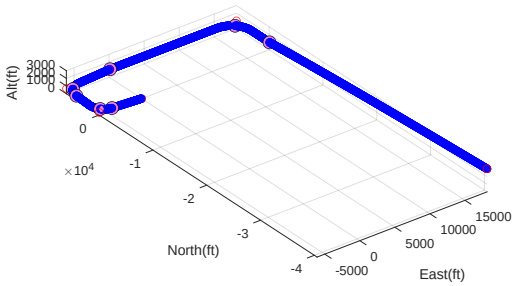


Fig. 20 Flight on crosswind entry.

Example Pattern Flights With Plan Modification

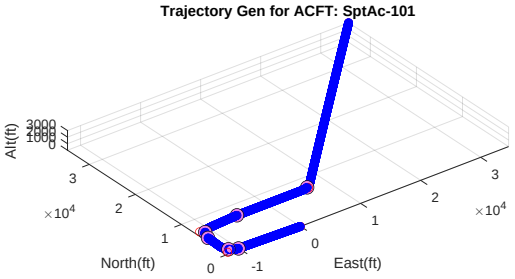


Fig. 21 Flight with an extended downwind leg.

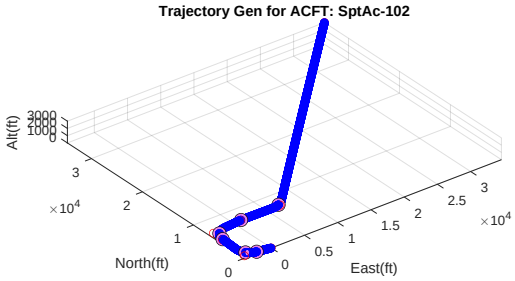


Fig. 22 Flight with an early base leg turn.

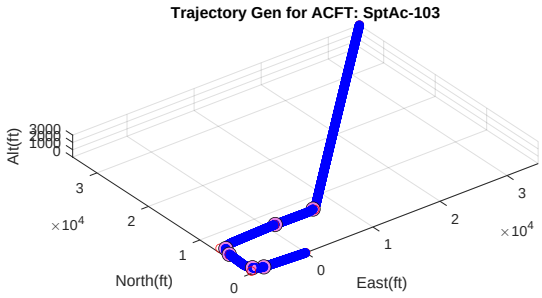


Fig. 23 Flight with a late base leg turn.

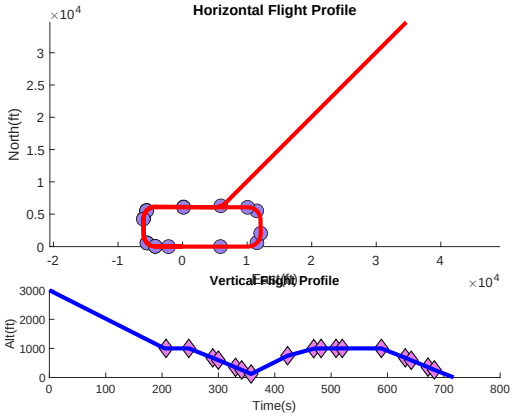


Fig. 24 Flight with an abort-go-around over the runway.

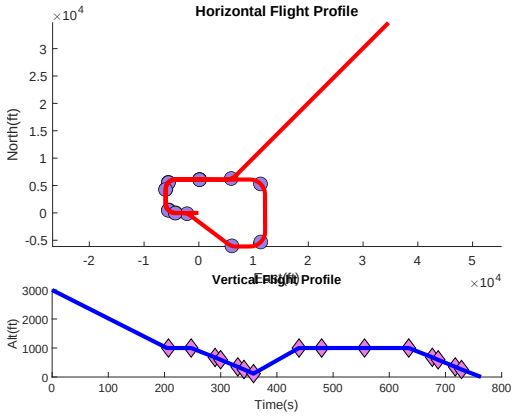


Fig. 25 Flight with an abort-go-around using upwind leg.

VII. Acknowledgments

This work was supported by the Transformation Tools and Technologies (TTT) and the Air Traffic Management–eXploration (ATM-X) projects. John Dana McMinn and Michael J. Acheson of NASA Langley Research Center, and Thomas C. Britton of Science and Technology Corporation contributed insights, development, testing, and validation throughout the life cycle of this software.

References

[1] Gregory, I. M., Acheson, M. J., Patterson, A. P., Houghton, M. D., Oshin, A., Ackerman, K. A., and Cook, J., "Candidate Performance Metrics for Generalized Control for Autonomous Flight," *AIAA SciTech*, National Harbor, MD, USA, 2023, pp. AIAA 2023–2650.

[2] Patrikar, J., Dantas, J. P. A., Ghosh, S., Kapoor, P., Higgins, I., Aloor, J. J., Navarro, I., Sun, J., Stoler, B., Hamidi, M., Baijal, R., Moon, B., Oh, J., and Scherer, S., "Challenges in Close-Proximity Safe and Seamless Operation of Manned and Unmanned Aircraft in Shared Airspace," *arXiv preprint arXiv:2211.06932*, 2022.

[3] McMinn, J. D., Patterson, A. P., and Gregory, I. M., "Traffic Prediction for Uncommunicative Aircraft in Terminal Area: Development Framework and Performance Evaluations," *AIAA SciTech*, Orlando, FL, USA, 2025. Accepted for publication.

[4] Houghton, M. D., Acheson, M. J., Patterson, A. P., Oshin, A., and Gregory, I. M., "COBRA-DDP: Trajectory Generation and Collision Avoidance Augmentations for eVTOL Vehicles," *AIAA SciTech*, Orlando, FL, USA, 2024, pp. AIAA 2024–0719.

[5] Houghton, M. D., Acheson, M. J., Patterson, A. P., Oshin, A., and Gregory, I. M., "Combined Bernstein Polynomial Optimal Reciprocal Collision Avoidance Differential Dynamic Programming for Trajectory Replanning and Collision Avoidance for UAM Vehicles," *AIAA SciTech*, National Harbor, MD, USA, 2023, pp. AIAA 2023–2544.

[6] Patterson, A., MacLin, G. L., Acheson, M. J., Tabasso, C., Cichella, V., and Gregory, I. M., "On Hermite Interpolation using Bernstein Polynomials for Trajectory Generation," Tech. Rep. TM-2023-0013467, National Aeronautics and Space Administration, Hampton, VA, USA, 2023.

[7] MacLin, G., Cichella, V., Patterson, A., Acheson, M., and Gregory, I., "Optimal Control using Composite Bernstein Approximants," *IEEE CDC*, accepted, 2024.

[8] Patrikar, J., Moon, B., Oh, J., and Scherer, S., "Predicting Like A Pilot: Dataset and Method to Predict Socially-Aware Aircraft Trajectories in Non-Towered Terminal Airspace," *arXiv preprint arXiv:2109.15158*, 2022.

[9] Navarro, I., Patrikar, J., Dantas, J. P. A., Baijal, R., Higgins, I., Scherer, S., and Oh, J., "Learned Tree Search for Long-Horizon Social Robot Navigation in Shared Airspace," *arXiv preprint arXiv:2304.01428*, 2023.

[10] *Airplane Flying Handbook*, Federal Aviation Administration, faa-h-8083-3c ed., 2022.

[11] *Pilot's Handbook of Aeronautical Knowledge*, Federal Aviation Administration, faa-h-8083-25c ed., 2023.

[12] Staff, A. S., "Glideslope Gouges," *Aviation Safety Magazine*, 2016.