# Autonomous Control of Air-Launched Fixed-Wing Drone Swarm

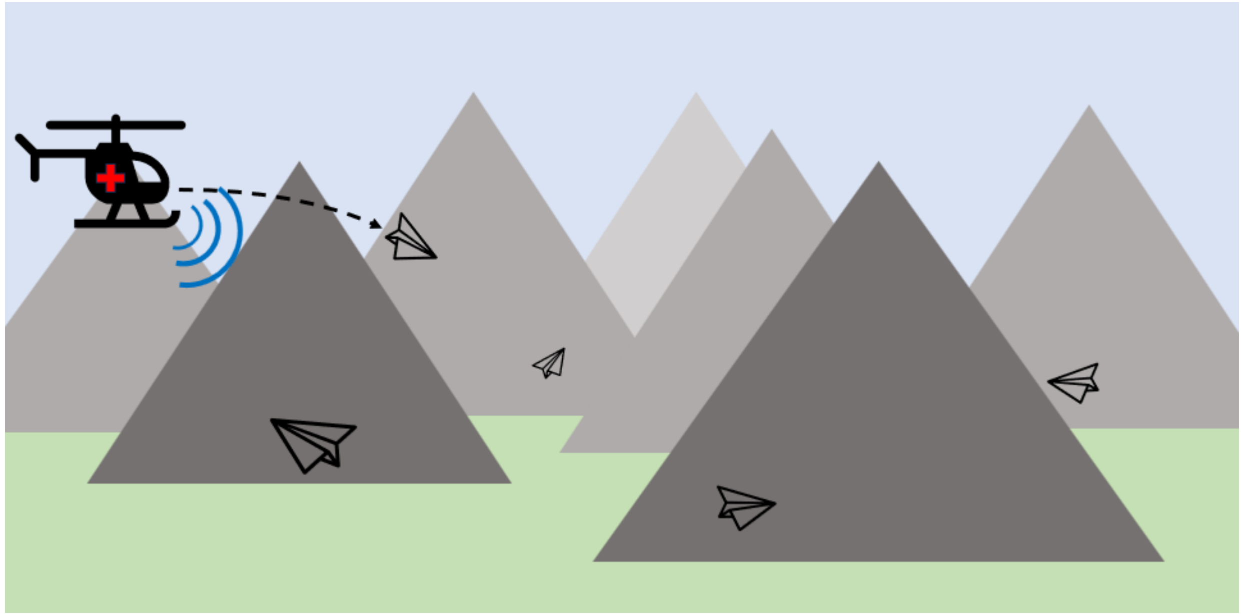
Culminating Experience Project in the Pursuit of Mechanical Engineering Master’s Degree

Alex Springer, June 2021 - May 2024

THE UNIVERSITY OF ALABAMA® | College of Engineering

# Project Definition and Overview

- Air-Launched Drone Swarms
- Independent Non-Linear Agent Control System
- User-Specified Commands:
    - Fly-Over Waypoints
    - Drone trajectories
- Agents Tracked by C2 Aircraft
- Agents Autonomously Assigned to User Commands



Use-Case: Search and Rescue in Mountainous Terrain

# Control System Hierarchy

### Agent States Continuously Estimated

- Each Agent Tracked by Command & Control Aircraft
- Agent Position, Velocity, and Acceleration Estimated by Random Finite-Set Based Filter

### User Input

- User Input of Fly-Over Waypoint (applications: search & rescue, surveillance, package drop)
- Latitude, Longitude, Altitude, Groundspeed

### Agent-Waypoint Assignment

- Cost Estimated for Each Agent: Distance, Altitude, Heading, and Groundspeed all considered
- Linear Hungarian Assignment Algorithm Used to Assign Agents and Minimize Overall Cost

### Individual Agent Controller

- Non-Linear Navigation Controller to Command Airspeed, Heading, and Rate of Climb
- Proportional Controller Converts Waypoint Assignments to Controller Inputs At Each Time Step

# Fixed-Wing Aircraft Non-Linear Guidance System (FANGS)

Individual Agent Flight Controller

# The Underlying Controller for Each UAS Agent

- Based on Nonlinear Aircraft-Performance Simulation by Dr. John Schierman in his Modern Flight Dynamics textbook
- Rigid fixed-wing aircraft operating in steady wind
- Uses a default state estimator with ideal equations of motion
  - User can provide a state solution at any time step; any unprovided state variables are estimated using the default ideal estimator
- Input: desired flight profile
  - Velocity, rate of climb, and heading

# Simulation Setup

- Drones modelled loosely from a C-130 fixed-wing aircraft
  - Long flight-time and/or range
  - Heavy payload (cameras, telemetry, first-aid package drop, etc.)
- Each drone launched from a C2 helicopter
- Flight controller in operation 150 feet after launch
  - Assume wings deploy during first 150 feet
  - After 150 feet, assumed 50 kts airspeed with no ascent or descent
- 8 drones launched ~5 seconds apart

# Ideal EOM: Governing Equations

Translational equations of motion for a vehicle with propulsive thrust aligned with the fuselage x-axis:

$$m\dot{V}_V = Tcos(\alpha)\cos(\beta) - D - mgsin(\gamma)$$

$$mV_V\big(\dot{\psi}_W\cos(\phi_W)\cos(\gamma) - \dot{\gamma}\sin(\phi_W)\big) = S + Tcos(\alpha)\sin(\beta) + mgsin(\phi_W)\cos(\gamma)$$

$$mV_V\big(\dot{\gamma}\cos(\phi_W) + \dot{\psi}_W\sin(\phi_W)\cos(\gamma)\big) = L + Tsin(\alpha) - mgcos(\phi_W)\cos(\gamma)$$

Reference: Modern Flight Dynamics by Dr. John Schierman

# Ideal EOM: Simplifying Assumptions

- Steady level flight or coordinated turns (sideslip $\beta$ and side force $S$ are zero)
- Assume sufficiently small angles of attack such that
  - $Tcos(\alpha) \approx T$
  - $Tsin(\alpha) \ll L$
- The governing equations become

$$\dot{V}_V = \frac{T - D}{m} - gsin(\gamma)$$

$$\dot{\gamma} = \frac{1}{mV_V}\left(Lcos(\phi_W) - mgcos(\gamma)\right)$$

$$\dot{\psi}_W = \frac{Lsin(\phi_W)}{mV_V \cos(\gamma)}$$

Reference: Modern Flight Dynamics by Dr. John Schierman

# Ideal EOM: Engine and Airframe Responses

- First-Order Differential Equations Used to Approximate the Engine and Airframe Responses

$$\dot{T} = -p_T T + p_T T_c$$
$$\dot{L} = -p_L L + p_L L_c$$
$$\dot{\phi}_W = -p_\phi \phi_W + P_\phi \phi_C$$

- Limits imposed on the responses

$$0 \leq T \leq T_{max}, L \leq K_{L_{max}}, |\phi_W| \leq \phi_{W_{max}}$$

- **Time constants $p_T, p_L, p_\phi$ set to 2.0, 0.5, and 1.0, respectively, in simulation**

Reference: Modern Flight Dynamics by Dr. John Schierman

# Ideal EOM: Aerodynamic Responses

- The model used in the ideal equations of motion for aerodynamic lift and drag

$$C_L = C_{L_\alpha}(\alpha - \alpha_0)$$

$$C_D = C_{D_0} + \frac{C_L^2}{K_D}$$

where $L = C_L q_\infty S_w$, $D = C_D q_\infty S_w$, $q_\infty = \frac{1}{2}\rho_\infty V_\infty^2$, and $K_D = \pi A e_{eff}$

Note: FANGS is designed to operate within the presence of wind, where $V_\infty \neq V_V$

Reference: Modern Flight Dynamics by Dr. John Schierman

# Ideal EOM: Angle of Attack and Drag

- Invert the prior equations to find the inferred Angle of Attack and Drag given a Lift command

$$D = K_{D_0} V_\infty^2 + K_{D_I}\left(\frac{L^2}{V_\infty^2}\right)$$

$$\alpha = K_L\left(\frac{L}{V_\infty^2}\right) + \alpha_0$$

where $K_{D_0} = \frac{1}{2}\rho_\infty S_w C_{D_0}$, $K_{D_I} = \frac{2}{\rho_\infty S_w K_D}$, and $K_L = \frac{2}{\rho_\infty S_w C_{L_\infty}}$

Reference: Modern Flight Dynamics by Dr. John Schierman

# Guidance Laws – Transfer Functions

- Inputs: Commanded **Velocity** $V_c$, **Rate of Climb** $\dot{h}_c$, and **Heading** $\psi_c$
- Outputs: Commanded **Thrust** $T_c$, **Lift** $L_c$, and **Bank Angle** $\phi_{W_c}$

$$\frac{T_c(s)}{V_E(s)} = \frac{mK_{T_P}\left(s + K_{T_I}/K_{T_P}\right)}{s}$$

$$\frac{L_c(s)}{\dot{h}_E(s)} = \frac{mK_{L_P}\left(s + K_{L_I}/K_{L_P}\right)}{s}$$

$$\frac{\phi_{W_c}}{\psi_E} = K_{\phi_P}\left(\frac{V_c}{g}\right)$$

Reference: Modern Flight Dynamics by Dr. John Schierman

# Guidance Laws - ODEs

- Convert the transfer functions to ODEs to be programmed in Python 3 and solved using scipy.integrate.solve_ivp with the RK45 method:

$$\dot{x}_T = mV_E$$
$$T_c = K_{T_I} x_T + K_{T_P} m V_E$$
$$\dot{x}_L = m\dot{h}_E$$
$$L_c = K_{L_I} x_L + K_{L_P} m \dot{h}_E$$
$$\phi_{W_c} = K_{\phi_P} (V_C/g) \psi_E$$

where $V_E \triangleq V_c - V_V$, $\dot{h}_E \triangleq V_c(\sin\gamma_c - \sin\gamma)$, and $\psi_E \triangleq \psi_c - \psi_w$

Reference: Modern Flight Dynamics by Dr. John Schierman

# Implementation (Python 3)

- FANGS.py
  - Create an object of class GuidanceSystem
    - Initialization inputs:
      - vehicle: Object of type FixedWingVehicle
      - TF_constants: Dictionary of PI Controller transfer function coefficients $(K_{T_P}, K_{T_I}, K_{L_P}, K_{L_I}, K_{\phi_P})$
      - InitialConditions: Dictionary of initial conditions
    - User-Accessible Functions:
      - setCommandTrajectory()
      - setFlyoverCommand()
      - getGuidanceCommands()
      - updateSystemState()

# Implementation (Python 3), continued

- After initialization, the drone is commanded to fly at initial conditions.
- The user may give any drone a command of either:
  - Trajectory
    - The user will supply a desired airspeed, heading, and rate of climb.
  - Flyover
    - The user will supply a desired waypoint, altitude, and groundspeed.
- Alternatively, the user may give the swarm a list of Flyover commands
  - See ATAMS section for more.

# User Commands: Trajectory

- User may request a trajectory command at any time:
  - **Velocity** $V_c$, **Rate of Climb** $\dot{h}_c$, and **Heading** $\psi_c$
  - Use setCommandTrajectory()
- Guidance Laws calculated when getGuidanceCommands() is called
  - Run this each time step

- Use-Case: Loiter, Send-To-Home

# User Commands: Flyover

- The user may request a flyover command at any time:
  - **Groundspeed** (feet/sec), **Altitude** (ft, MSL), **Waypoint** (latitude, longitude)
  - Use setCommandFlyover()
- An internal flag will note that a flyover has been requested
- At each timestep, when getGuidanceCommands() is run, FANGS will internally convert flyover commands to trajectory commands for the aircraft guidance logic
  - Proportional Controller: see next slide

# User Commands: Flyover, continued

1. Correct aircraft altitude before adjusting velocity

   Calculate required glideslope $\dot{h}_c$ to reach command altitude

$$
\begin{aligned}
&\text{if abs}\left(\dot{h}_c\right) > 15°: \\
&\qquad \dot{h}_c = sign\left(\dot{h}_c\right) * 15° \\
&\text{elseif } abs\left(\dot{h}_c\right) < 3°: \\
&\qquad \dot{h}_c = 0 \\
&\text{else:} \\
&\qquad \dot{h}_c = \dot{h}_c + sign\left(\dot{h}_c\right) * K_\alpha * \dot{h}_c
\end{aligned}
$$

# User Commands: Flyover, continued

2. Once altitude is stabilized, fly as fast as possible until within 2 miles of the flyover point

if $\dot{h}_c = 0$:

    if distance from target $>$ 2 miles:

        velocity $=$ maximum allowable velocity - 15

    else:

        velocity $= K_v * (input\ velocity - current\ airspeed) + current\ airspeed$

# User Commands: Flyover, continued

3. While changing velocity and altitude, continually adjust heading to intercept flyover point

$$\text{heading} = \text{bearing between}(\text{current latitude/longitude}, \text{target latitude/longitude})$$

4. If the aircraft is within 300 feet of the target, “let go” of the flyover waypoint and switch to a trajectory controller holding the current flight path for intercept.
   - Aircraft will maintain this flight path until the user specifies a new trajectory or flyover command

# Example Simulation – Single Agent

- Initial Conditions
  - Airspeed = 50 knots
  - Altitude = 7000 feet above mean sea level
  - Ascent = 0 degrees
  - Heading = 0 degrees (North)
  - (Latitude, Longitude) = (36.2434, -112.2822) degrees
  - Weight = 80 pounds
  - Wind = 15 knots east

# Example Simulation – Single Agent

- Drone Parameters
  - Max speed = 75 knots
  - Min speed = 25 knots
  - $p_T, p_L, p_\phi$ = (2.0, 0.5, 1.0)
  - Max thrust = 45 pounds
  - Max $K_L$ = 0.26
  - Max bank angle = 45 degrees
  - $C_{D_0}$ = 0.05
  - $C_{L_\alpha}$ = 0.5 radians
  - $\alpha_0$ = -0.1 degrees
  - Wing area = 8 square feet
  - Aspect ratio = 12
  - Wing efficiency factor = 0.8

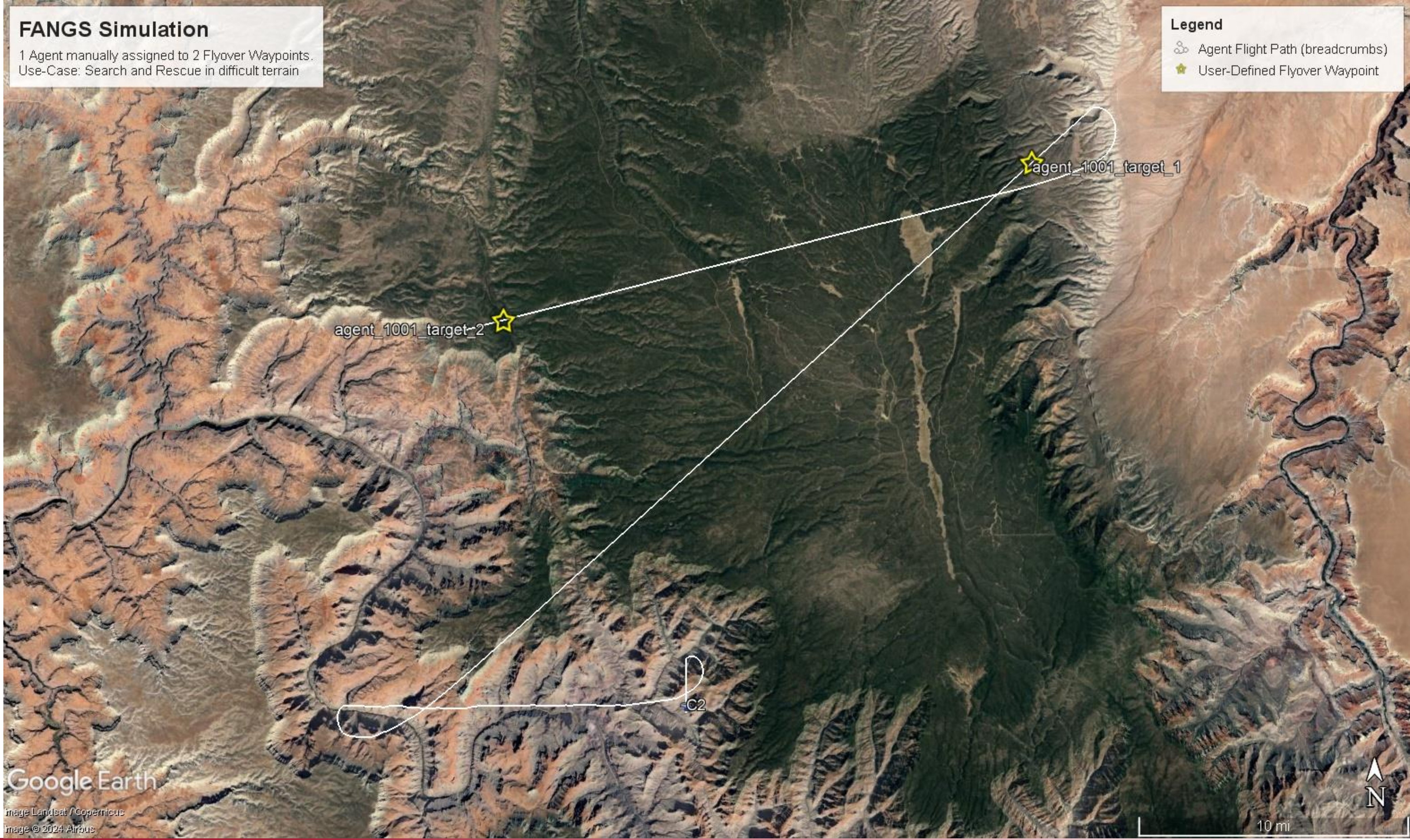
# Example Simulation – Single Agent

- Controller Transfer Function Gains
  - $K_{T_P} = 0.15$
  - $K_{T_i} = 0.05$
  - $K_{L_P} = 0.3$
  - $K_{L_i} = 0.03$
  - $K_{\phi_P} = 0.03$
  - $K_{\alpha} = 0.05$
  - $K_{v} = 0.05$

# Example Simulation – Inputs

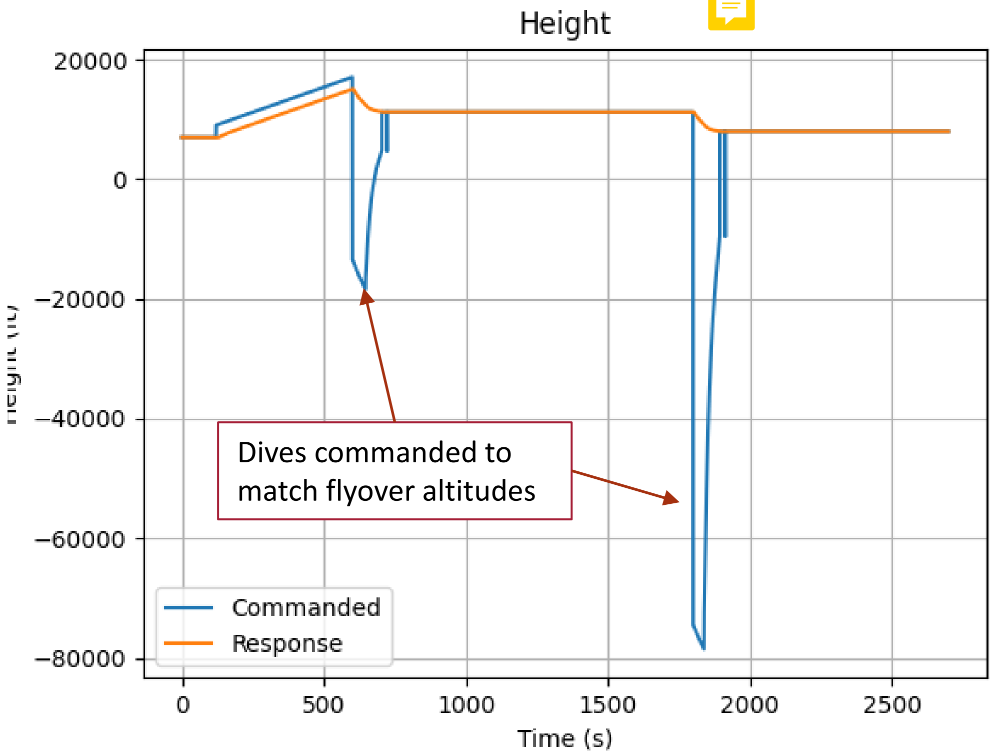
- Initial conditions: 50 knots, level flight, North heading ($0^o$)
    - 15 knot wind out of the west
- After 120 seconds, command trajectory: 95 knots, 6-degree ascent, West heading ($270^o$)
- After 600 seconds, command flyover: 50 knots, 11000 feet MSL, (36.530367, -112.057600)
- After 1800 seconds, command flyover: 50 knots, 7700 feet MSL, (36.449291, -112.399009)
- Simulate 45 total minutes of flight time

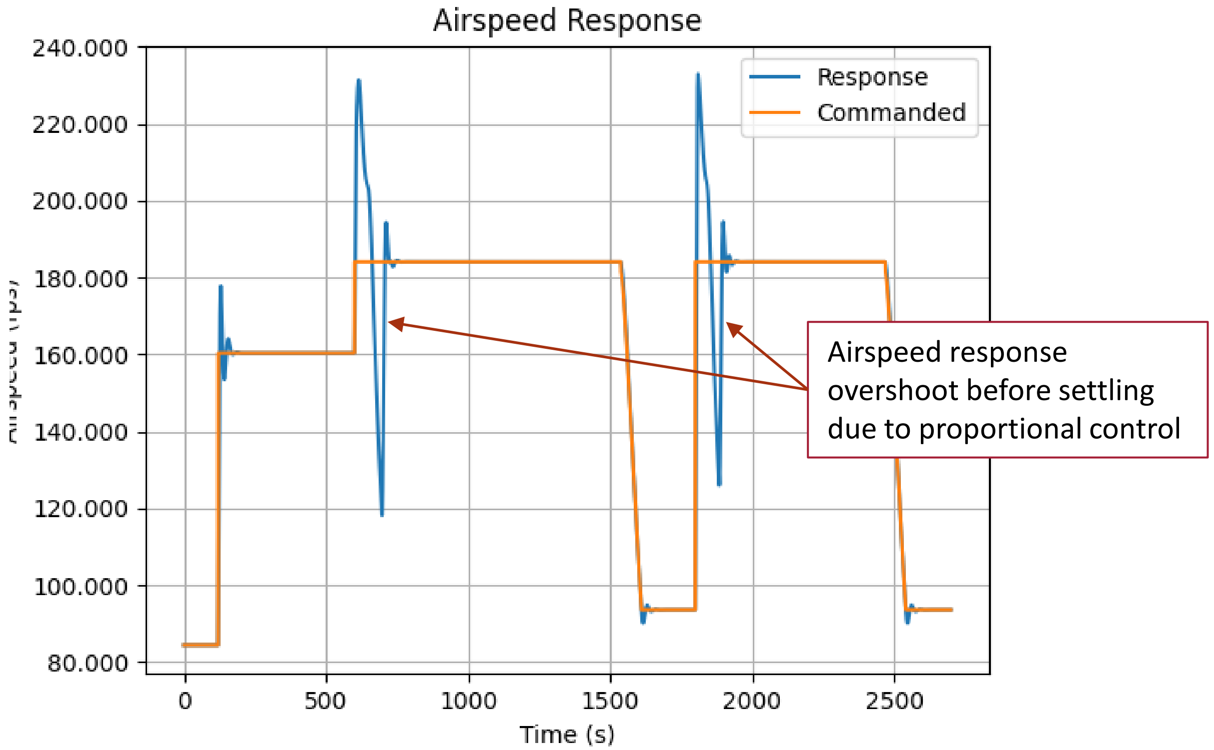
# Results



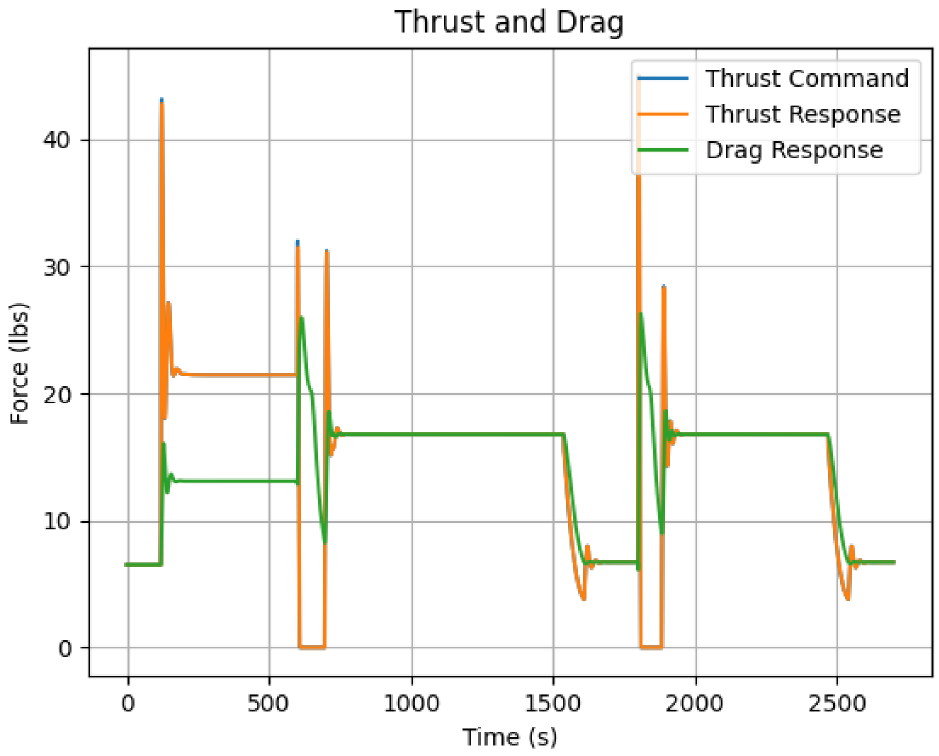
FANGS Simulation
1 Agent manually assigned to 2 Flyover Waypoints.
Use-Case: Search and Rescue in difficult terrain
Legend
Agent Flight Path (breadcrumbs)
User-Defined Flyover Waypoint
agent_1001_target_1
agent_1001_target_2
C2
Google Earth
10 mi
N

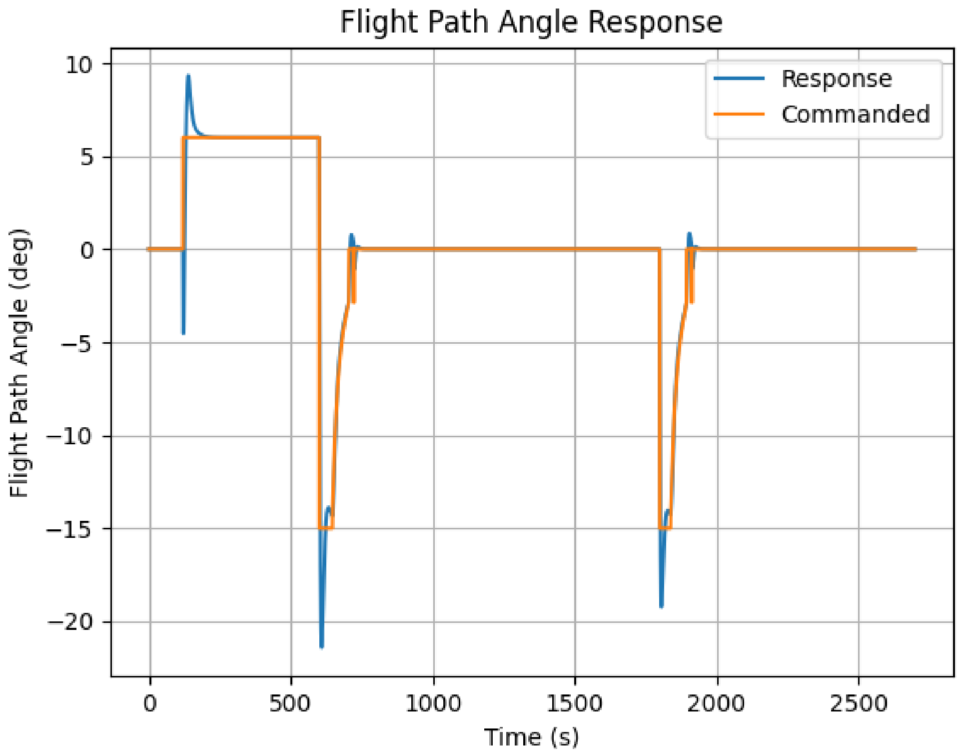
# Example Simulation Results, Altitude and Airspeed



# Example Simulation Results, Thrust, Drag, and Flight Path Angle



# Agent Tracking and Assignment Management System (ATAMS)

Agent-Waypoint Assignment

# Target Assignments – Cost

- Cost value assigned to each target-agent assignment
  - Each calculation uses a weighting that can be user-defined
    - $W_d$ : Weighting of Distance Cost
    - $W_a$ : Weighting of Altitude Cost
    - $W_v$ : Weighting of Velocity Cost
    - $W_h$ : Weighting of Heading Cost

- Calculate the cost of each possible agent-target assignment
  - Ex: If 8 active agents and 4 assignments, 32 total cost calculations

# Target Assignments – Cost, Continued

- Distance Cost:
  - Simply multiply the distance between the agent and the target by $W_d$
- Heading Cost:
  - Simply multiply the heading change required to intercept the target by $W_h$
- Altitude Cost:
  - Multiply the required change in altitude by $W_a$
  - If the required altitude is lower than the current agent altitude, multiply the cost by 0.75
    - It is cheaper to lower an agent's altitude than raise it
- Velocity Cost:
  - Simply multiply the velocity change required by $W_v$
- **Total Assignment Cost: Sum the Distance, Heading, Altitude, and Velocity cost**

# Target Assignments – Cost Matrix

- Cost Matrix calculated using Hungarian Algorithm
  - Use Python 3 scipy package scipy.optimize.linear_sum_assignment
  - Returns cost matrix of n targets to m agents
- Targets assigned by Cost Matrix

# Target Assignments - Example

- Eight active airborne agents
  - Each defined using parameters in Single Agent example (slides 21-26)
  - Launched from C2 helicopter ~5 seconds apart
  - Each given unique random trajectory 30 seconds after initialization
    - All agents given a 10 degree rate of climb
- Define 8 target flyovers
  - Command flyovers to swarm 120 seconds after $1^{st}$ drone initialization
  - Target flyovers defined on next slide

# Target Assignments – Example, Continued

- Weights used in assignment:
  - $W_d$ : 10
  - $W_a$ : 1
  - $W_v$ : 0.1
  - $W_h$ : 100
  - Assumptions:
    - Distance from target is more important than altitude change required
    - Velocity change required is not very important because FANGS will speed each aircraft up before intercept
    - Heading change is very important because turning induces instability and costs a lot of flight time

# Target Assignments – Example, Continued

**8 Flyover Assignments**

| Target | Latitude (Deg, North) | Longitude (Deg, West) | Altitude (MSL) | Groundspeed (knots) |
|---|---|---|---|---|
| 1 | 36.530367 | 112.057600 | 11000 | 50 |
| 2 | 36.179491 | 111.951595 | 12000 | 50 |
| 3 | 36.276756 | 112.687766 | 9600 | 50 |
| 4 | 36.542782 | 112.124202 | 12000 | 50 |
| 5 | 36.089355 | 112.409598 | 10000 | 50 |
| 6 | 36.218540 | 111.969167 | 12600 | 50 |
| 7 | 36.449291 | 112.399009 | 7700 | 50 |
| 8 | 36.580698 | 111.866192 | 9000 | 50 |

# Target Assignments – Example, Continued

## Agent States at 120 seconds (time of cost estimate)

| Agent | Latitude (Deg, North) | Longitude (Deg, West) | Altitude (MSL) | Groundspeed (knots) | Heading (Deg) | Flight Path Angle (Deg) |
|---|---|---|---|---|---|---|
| 1 | 36.23111 | -112.27017 | 8525.751 | 80.058 | 182.888 | 10.00212 |
| 2 | 36.25135 | -112.24750 | 9046.887 | 106.007 | 99.566 | 10.00093 |
| 3 | 36.23797 | -112.29225 | 8535.040 | 96.050 | 268.649 | 10.01889 |
| 4 | 36.26619 | -112.28151 | 8425.689 | 80.023 | 1.920 | 9.999995 |
| 5 | 36.23514 | -112.27546 | 8424.269 | 102.037 | 208.198 | 10.02664 |
| 6 | 36.25800 | -112.26417 | 8195.515 | 94.866 | 82.316 | 10.00612 |
| 7 | 36.23767 | -112.28039 | 8077.037 | 84.689 | 225.048 | 10.03656 |
| 8 | 36.23940 | -112.28349 | 8047.320 | 90.942 | 245.432 | 10.07828 |

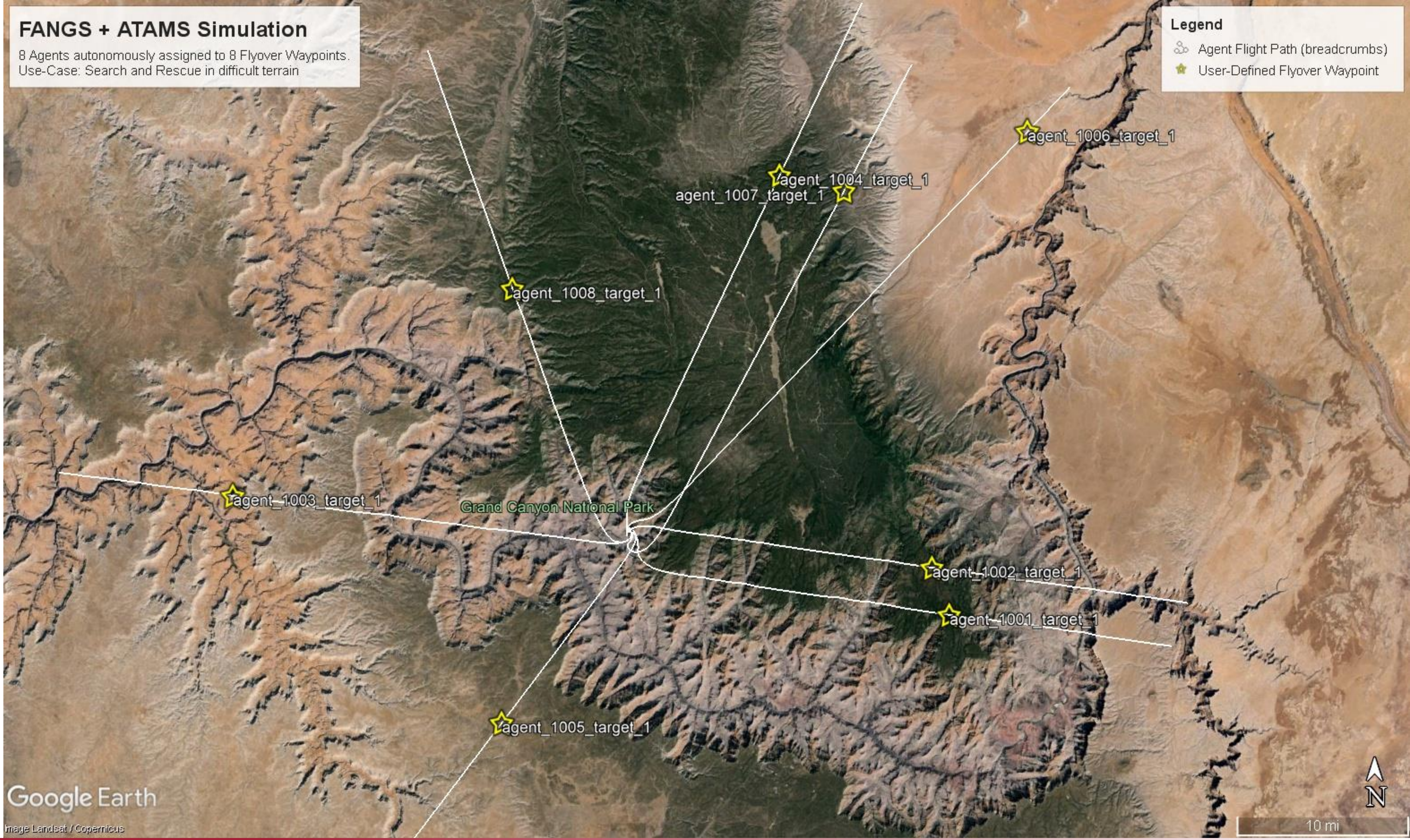
# Target Assignments – Example, Continued

## Agent-Target Cost Matrix with Optimal Assignments

| | AGENT 1 | AGENT 2 | AGENT 3 | AGENT 4 | AGENT 5 | AGENT 6 | AGENT 7 | AGENT 8 |
|---|---|---|---|---|---|---|---|---|
| TARGET 1 | 2980.23 | 3798.31 | 1475.21 | 3988.04 | 1661.90 | 4399.78 | **1051.47** | 1049.03 |
| TARGET 2 | **2298.85** | 3139.92 | 1106.01 | 3309.01 | 1307.57 | 3714.36 | 1401.93 | 447.31 |
| TARGET 3 | 2923.33 | 3951.30 | **1302.78** | 3897.66 | 1685.80 | 4551.22 | 905.93 | 1038.87 |
| TARGET 4 | 2852.11 | 3953.20 | 1558.17 | **3824.42** | 1980.81 | 4524.15 | 738.10 | 969.83 |
| TARGET 5 | 3124.64 | 3947.58 | 1529.54 | 4108.66 | **1717.72** | 4548.34 | 929.64 | 1194.79 |
| TARGET 6 | 3114.98 | **4031.45** | 1936.90 | 4123.21 | 2178.88 | 4601.87 | 719.90 | 1185.37 |
| TARGET 7 | 3450.58 | 4324.94 | 1842.07 | 4425.55 | 2065.54 | 4925.22 | 636.43 | **1565.91** |
| TARGET 8 | 3446.11 | 4392.38 | 1834.55 | 4420.77 | 2133.14 | **4992.35** | 578.07 | 1561.64 |

**Total assignment cost = 20784.93**

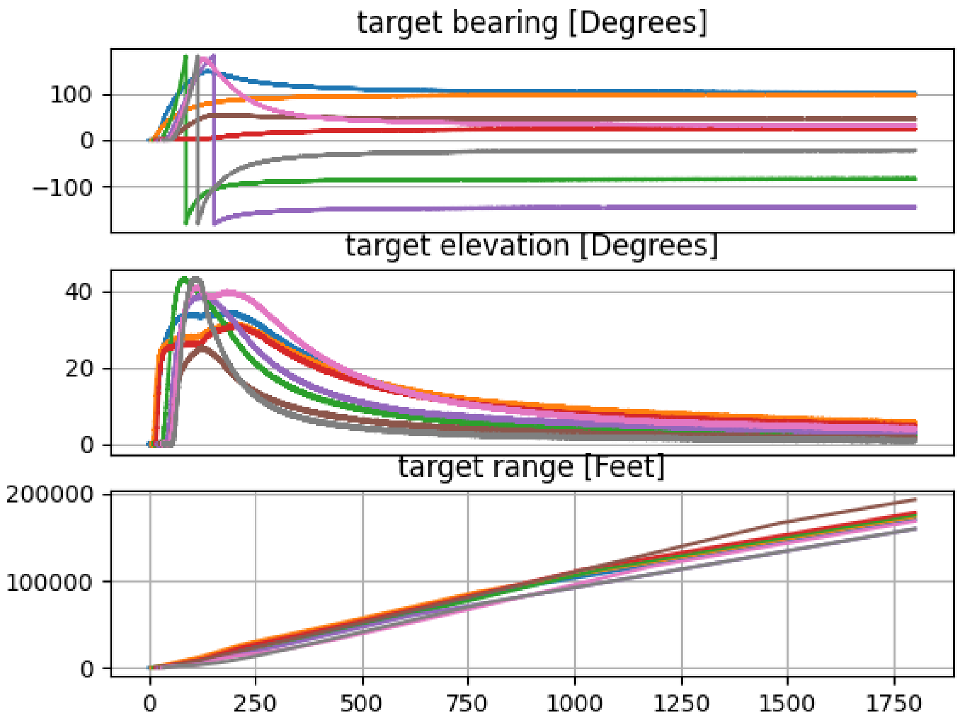
# Results



FANGS + ATAMS Simulation
8 Agents autonomously assigned to 8 Flyover Waypoints.
Use-Case: Search and Rescue in difficult terrain
Legend
Agent Flight Path (breadcrumbs)
User-Defined Flyover Waypoint
agent_1006_target_1
agent_1004_target_1
agent_1007_target_1
agent_1008_target_1
agent_1003_target_1
Grand Canyon National Park
agent_1002_target_1
agent_1001_target_1
agent_1005_target_1
Google Earth
Image Landsat / Copernicus
N
10 mi

# Future Development - Agent Tracking

- Each agent tracked by the C2
  - Assumption: onboard radar, infrared, or other sensor system
- Raw track data output by FANGS
  - Target bearing, elevation, and range relative to C2
  - Simulated sensor noise
    - Gaussian noise
    - Future iterations may simulate false tracks, track drop-outs, non-Gaussian noise, and range-dependent noise



# Future Development - Agent Tracking, Continued

- Future Development: Implement random finite-set based tracking algorithms from University of Alabama LAGER Python package CARBS.
- Add air-to-air targets and track these targets
  - Intercept air-to-air targets with drones commanded by FANGS