Numerical Programming Final Project (Task 2)
Transition to New Year Greeting

Student: **(Andria Gvaramia)** Course: Numerical Programming KIU

January 18, 2026

Task statement

Input: drone swarm at the handwritten name (Task 1 final formation) and the greeting text *"Happy New Year!"*.
Goal: move the swarm from the handwritten name positions to the holiday greeting positions.
Output: trajectory of each drone and visualization whose input is the trajectory.

1 Overview

Task 2 is implemented as a transition between two static point sets:

- **Start formation (Task 1)**: points in `task1/outputs/target_points.csv`.
- **Target formation (Task 2 greeting)**: points in `task2/outputs/target_points.csv`.

The greeting image is generated in `task2/inputs/greeting.png` and target points are extracted using the same tool as Task 1 (`task1/extract_target_points.py`). The transition trajectories are generated by `task2/transition.py` using a **BVP solved via shooting**.

2 Input data

2.1 Greeting image

The greeting image is produced programmatically (OpenCV text rendering) by:

```
python3 task2/generate_greeting_image.py --out task2/inputs/greeting.png
```

2.2 Target point extraction

We extract N target points from the greeting image (typically in skeleton / medial-axis mode):

```
python3 task1/extract_target_points.py \
  --image task2/inputs/greeting.png \
  --n 100 --mode skeleton --min-target-spacing 5 \
  --out-dir task2/outputs --debug-png
```

Important constraint: the number of drones must match in both tasks:

$$N_{\text{start}} = N_{\text{target}}.$$

In practice, the same `--n` is used in Task 1 and Task 2.

3 Mathematical model

We work in 2D pixel coordinates. For each drone i we use a second-order point-mass model:

$$x_i(t) \in \mathbb{R}^2, \qquad v_i(t) \in \mathbb{R}^2.$$

3.1 Dynamics

The per-drone ODE used in `task2/transition.py` is:

$$\dot{x}_i(t) = v_i(t), \tag{1}$$

$$\dot{v}_i(t) = \frac{1}{m}\left(k_p\,(T_i - x_i(t)) - k_d\,v_i(t)\right). \tag{2}$$

Here $T_i \in \mathbb{R}^2$ is the **fixed** greeting target assigned to drone i.

3.2 Boundary value problem (BVP)

Task 2 is formulated as a boundary value problem:

$$x_i(0) = x_{i,0} \quad \text{(from Task 1)}, \qquad x_i(T) = T_i \quad \text{(greeting targets)}.$$

4 Numerical method: shooting

We solve the BVP via **shooting**. For each drone, the unknown is the initial velocity $v_i(0)$.

- Given a guess $v_i(0)$, we integrate the ODE from $t = 0$ to $t = T$ using `solve_ivp` (RK45).
- We compute the terminal error $F(v_i(0)) = x_i(T; v_i(0)) - T_i$.
- We update $v_i(0)$ using a nonlinear solver.

4.1 Reducing end oscillation

To reduce overshoot near the end, we optionally solve a least-squares problem that also prefers $v_i(T) \approx 0$:

$$r(v_0) = \left[x(T; v_0) - T,\ w_v\,v(T; v_0)\right],$$

enabled via:

```
--bvp-match-final-velocity --bvp-final-velocity-weight 3.0
```

5 Collision discussion and verification

This Task 2 shooting implementation solves drones independently (no inter-drone repulsion term), therefore collision-free motion is **not guaranteed** by the model.

However, we can **verify** whether collisions occurred by measuring the minimum pairwise distance over time:

$$d_{\min} = \min_t \min_{i<j} \|x_i(t) - x_j(t)\|.$$

The script reports the closest approach using:

```
--collision-report --collision-threshold 12
```

6 Reproducibility (commands)

All commands are run from the project root.

6.1 Task 1 (ensure start formation with N drones)

```
python3 task1/extract_target_points.py --n 100 --mode skeleton --min-target-spacing 5
    --debug-png
```

6.2 Task 2 (transition)

```
python3 task2/transition.py \
  --start task1/outputs/target_points.csv \
  --targets task2/outputs/target_points.csv \
  --bg-target task2/inputs/greeting.png \
  --bvp-match-final-velocity --bvp-final-velocity-weight 3.0 \
  --k-p 2.0 --k-d 2.5 \
  --t-end 20 --steps 200 \
  --collision-report --collision-threshold 12 \
  --save-gif --save-traj-csv --save-traj-npy --save-traj-plot
```

7 Outputs

Generated outputs are written to `task2/outputs/`:

- `transition_motion.gif`
- `transition_trajectories.png`
- `transition_trajectories.csv` and `transition_trajectories.npy`
- `target_points.csv/.npy` and `debug_target_points.png` for the greeting

8 Limitations

- Shooting is per-drone and does not enforce collision avoidance.
- Large N increases runtime because shooting solves many small IVPs (one per drone, potentially multiple times due to root finding).
- The quality of the greeting formation depends on the target point extraction parameters (mode, spacing, and N).

9 AI usage disclosure

This project was developed with AI assistance (ChatGPT) used for:

- explaining BVP and shooting concepts,
- proposing model/parameter adjustments,
- helping implement and debug Python code, CLI arguments, and visualization.