README: Vortex Dynamics - Baxilisk Analysis Tools

Matilde Bureau, Gaston Ravanas

January 20, 2026

1 Overview

This folder contains two specialized Python scripts designed to automate the post-processing of Computational Fluid Dynamics (CFD) simulations generated using the **Basilisk** solver. The toolkit handles data organization, parametric comparison, and visual temporal analysis of vortex interactions. The attached scripts were developed to streamline the analysis of vortex interactions. By automating the extraction of circulation and trajectory data, the focus of the project remained on the underlying physics—specifically how the Reynolds number affects vortex merger or wall-interaction distances. The use of biased sampling in the video processing ensures that transient events, such as initial "leap-frogging" or dipole formation, are captured with high temporal resolution.

2 Script Descriptions

2.1 plot_basilisk.py

This script is the primary analytical engine. It parses numerical data files (`.out`) to evaluate the physical properties of the simulation.

- **Automatic Categorization**: Detects the simulation case (Corotating, Dipole, Single, or Image vortex) and organizes files into subdirectories automatically.
- **Trajectory Tracking**: Extracts coordinates (x_c, y_c) or vorticity extrema to track the spatial movement of vortex cores over time.
- **Physical Stability**: Plots circulation (Γ) to verify the conservation of vortex strength and the subsequent effects of viscosity.
- **Parametric Study**: Identifies simulations where one variable (e.g., Reynolds number Re) changes while others remain fixed, generating comparison plots to isolate physical influences.

2.2 mp4toframes.py

This script translates simulation videos into static visual reports for qualitative analysis.

- **Physics-Based Sampling**: Instead of simple uniform sampling, the script uses "biased sampling" for specific cases like "Image" or "Corotating" vortices to capture more frames during high-activity interaction phases.
- **Automated Labeling**: Extracts physical parameters from the filename and burns the simulation timestamp ($t = X.XXX$ s) directly onto the frames.
- **Grid Visualization**: Produces a high-resolution PDF grid for each video, allowing for side-by-side comparison of temporal evolution without manually scanning raw video files.

3 Workflow Summary

1. **Simulation**: Run Basilisk simulations, naming output files with parameters (e.g., `dipole_Re_1000_G2_1.0.out`).
2. **Data Organization**: Execute `plot_basilisk.py` to sort files and generate quantitative graphs and a `comparison_summary.csv`.
3. **Visual Analysis**: Execute `mp4toframes.py` to generate time-stamped image grids from the simulation videos.

4 Dependencies

The scripts require the following Python libraries:

- **Data Handling**: `pandas`, `numpy`
- **Visualization**: `matplotlib`
- **Video Processing**: `opencv-python (cv2)`
- **Utilities**: `os`, `re`, `glob`, `shutil`