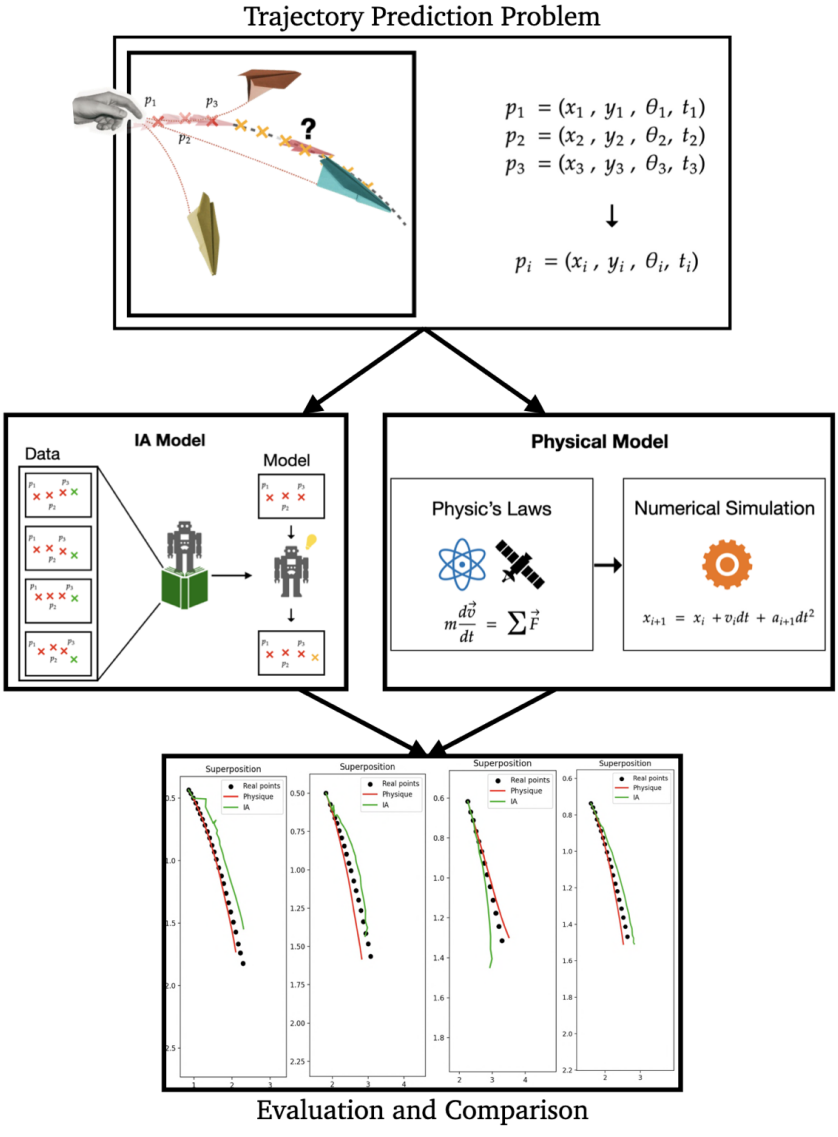
TIPE Project Report: Comparing Physical and AI Models for Predicting Paper Airplane Trajectories

Arthur Oudeyer

September 22, 2025

Abstract

This report presents a one-year research project in the context of CPGE TIPE between 2023 and 2024, comparing traditional physics-based models and modern AI techniques for predicting the trajectory of paper airplanes. The project involves experimental data collection, analysis using computer vision tools, and the comparison of the two approaches in terms of accuracy, generalization, and reliability. The goal is to determine when AI models might be viable or even more effective than classical physical models. The study concludes that AI models are viable for specific cases where it's easy to collect lot of data while the physical model demands a deeper theoretical understanding but is more generalizable.



Contents

1 Introduction

1.1 Context

A paper airplane is an origami constructed in such a way that it can glide when thrown. Its trajectory depends on its profile and how it is launched.

This can be modeled using aerodynamic laws [1]. These models have existed for a long time: as early as the 19th century, the actions of weight, lift, drag, and thrust had been theorized [2]. In the case of paper airplanes, there is no question of thrust; only the initial impulse is taken into account. The drag and lift forces are modeled using drag coefficients called Cx and Cz, which are characteristic of the system. However, there is no general explicit analytical method for determining these coefficients. They are generally determined experimentally [3].

To do this, it is possible to observe the behavior of the system to determine them. By recording videos of paper airplane trajectories, it is possible, using computer vision techniques available in libraries such as OpenCV [4], and particularly object detection based on pre-trained artificial neural networks like YOLOv8 [5, 6], to first recover the trajectory, then the speed and acceleration. From this data, it is possible to estimate the applied forces to deduce the drag coefficients. Once determined, it is possible, using the fundamental principle of dynamics, to determine the trajectory of the airplane from initial conditions.

To predict the trajectory, another approach can be considered, based on the use of tools recently developed in the field of artificial intelligence, such as neural networks. This type of model was first theorized in 1958 with the invention of the perceptron [7]. More recently, multi-layer perceptrons, which are non-linear regression models, randomly initialized, and whose parameters are estimated iteratively by the gradient descent optimization technique and from a database, have shown great capabilities in many application domains [8]. This type of model is now relatively simple to use, particularly thanks to the Scikit-Learn library in Python [9].

However, a data collection campaign is necessary: in this case, it involves collecting a set of varied trajectories of paper airplane(s). With these, it is possible to train a neural network to predict their trajectories. This procedure is very different from the previous method. It does not start a priori from the laws governing the trajectory but seeks regularities in the data, without any particular physical meaning, allowing for precise prediction of the result. In some domains, many problems are very complex to model: this is why this approach offers certain advantages, notably the lack of need for initial physical modeling [10].

1.1.1 Bibliographic References

1 SERVICE D'EXPLOITATION DE LA FORMATION AÉRONAUTIQUE (SEFA): Précis d'aérodynamique et de Mécanique de vol: https://www.dassaultvoltige.fr/wp-content/uploads/2018/03/precis-aerodynamiqe-V7.pdf (2007)

2 ANDERSON, J. D.: A History of Aerodynamics and its Impact on Flying Machines: Cambridge University Press (1997)

3 FALKOVICH, G.: Fluid Mechanics (A Short Course for Physicists): Cambridge University Press (2011)

4 JIA, Q., PENNISI, A., & RAMACHANDRA, V.: OpenCV, Open Source Computer Vision Library: https://opencv.org

5 REDMON, J., DIVVALA, S., GIRSHICK, R., & FARHADI, A.: You Only Look Once: Unified, Real-Time Object Detection: In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 779-788) (2016)

6 JOCHER, G., CHAURASIA, A.: Ultralytics YOLOv8, a Real-Time Object Detection and Image Segmentation Model: https://docs.ultralytics.com

7 ROSENBLATT, F.: The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain: Psychological Review 65 (6): 386–408 (1958)

8 LECUN, Y., BENGIO, Y., & HINTON, G.: Deep Learning: Nature 521(7553), 436-444 (2015)

9 PEDREGOSA ET AL.: Scikit-Learn: Machine Learning in Python: JMLR 12, pp. 2825-2830 (2011)

10 WILLARD, J., JIA, X., XU, S., STEINBACH, M., & KUMAR, V.: Integrating Scientific Knowledge with Machine Learning for Engineering and Scientific Applications: (Year not provided)

1.2 Problematic

The prediction of paper airplane trajectories is a complex problem that can be approached using both traditional physics-based models and modern AI techniques [3][4]. This project aims to explore the viability and limitations of these methods.

The specific question addressed in this project is:

In what cases and to what extent can a model developed by AI be more relevant than a physical model in predicting the trajectories of paper airplanes?

1.3 Objectives

The main objectives of this project are:

- To collect experimental data on paper airplane trajectories.
- To develop and compare physical and AI models for trajectory prediction.
- To evaluate the accuracy, generalization, and reliability of each approach.
- To study the advantages and disadvantages of two methods (using the laws of physics or using AI) aiming to effectively predict the trajectory of a paper airplane.

1.4 Concepts and Tools

1 OpenCV: An open-source computer vision and machine learning software library. It provides a common infrastructure for computer vision applications and accelerates the use of machine perception in commercial products.

2 YOLOv8: A state-of-the-art, real-time object detection model developed by Ultralytics. It is used for detecting objects in images and videos.

3 Scikit-learn: A machine learning library for the Python programming language. It features various classification, regression, and clustering algorithms.

4 MLPRegressor: A multi-layer perceptron regressor in Scikit-learn that optimizes the squared-loss using LBFGS or stochastic gradient descent.

5 Euler Method: A numerical procedure for solving ordinary differential equations (ODEs) with a given initial value.

6 Gradient Descent: An optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient.

7 Normalization: A process often applied as part of data preparation for machine learning. The goal is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values.

8 Auto-regression: A time series model that uses observations from previous time steps as input to a regression equation to predict the value at the next time step.

2 Technical Challenges

2.1 Models requirements

The problem is to predict the trajectory of a paper airplane from three initial points, where each point consists of coordinates, an angle, and a time.

To solve this, the physical model is based on the laws of mechanics and uses numerical simulation tools. In contrast, the AI model has no physical understanding; it relies on observed regularities to predict the trajectory. For this, a model is trained on the following task: predict the fourth point in a series of three. From this, to predict the complete trajectory, auto-regression [8] is used.

For implementing these models, several things are needed:

- The drag and lift coefficients of the airplane for the physical model.
- A large amount of training data for the AI.

In both cases, collecting a large number of trajectories is necessary for their implementation.

2.2 Data Collection

The data collection process involved:

- Recording videos of paper airplane flights at 60 frames per second.
- Using YOLOv8 for object detection to extract trajectory data.
- Converting pixel coordinates to real-world units.

Figure 1: *The video acquisition setup, with scale object at the bottom left. The green box is the detection of the paper airplane by the YoloV8 model. The bottom right corner of the green box is taken a the followed point.*

2.3 Data Processing

The data processing steps included:

- Encoding and decoding trajectory data.
- Normalizing [9] and filtering data points.
- Calculating speed and acceleration using Euler's method [5].

Once the trajectories were collected, the coordinate data as a function of time, the angle of attack was also needed. For this, it was approximated that the angle of attack is the same as that made by the vector passing through the center of the frame and the bottom right corner of the detection box.

Using the scale, pixel coordinates were converted to usable units and all trajectories were encoded in text files. In the end, a total of 90 clean recorded trajectories were obtained, including 77 'classic' ones and 13 more original ones.

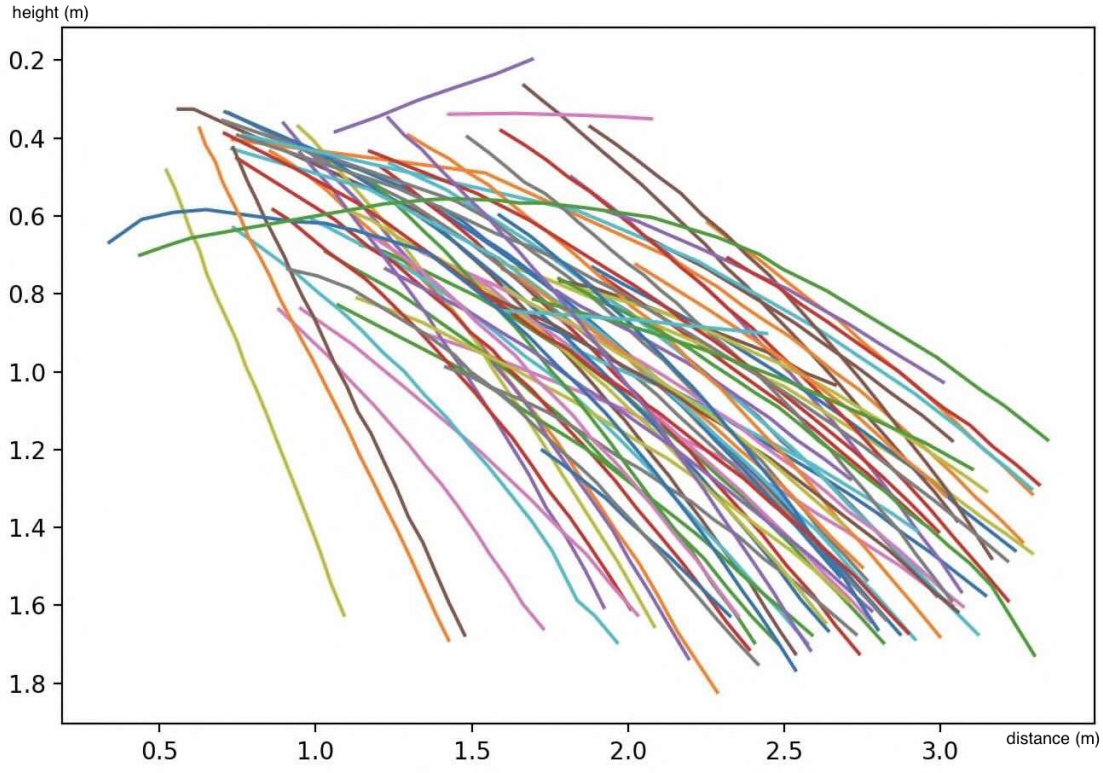


Figure 2: *Trajectories Collected : Here are plotted all of the trajectories extracted from the video taken during the experiment. There is a total of 77 'classical' trajectories and 13 more 'original' ones.*

3 Modeling

3.1 Physical Model

The physical model aims to predict the trajectory of paper airplanes using fundamental principles of mechanics and aerodynamics. The model relies on the following parameters and assumptions:

- Mass of the airplane: $m = 10$ grams
- Surface area: $S = 0.03$ m^2
- Reynolds number: $\text{Re} = \frac{\rho V L}{\eta} \approx 10^5$

3.1.1 Hypotheses

- Only weight and aerodynamic forces are considered.
- Drag coefficient (C_x) and lift coefficient (C_z) are assumed to be constant.

3.1.2 Aerodynamic Forces and Equations

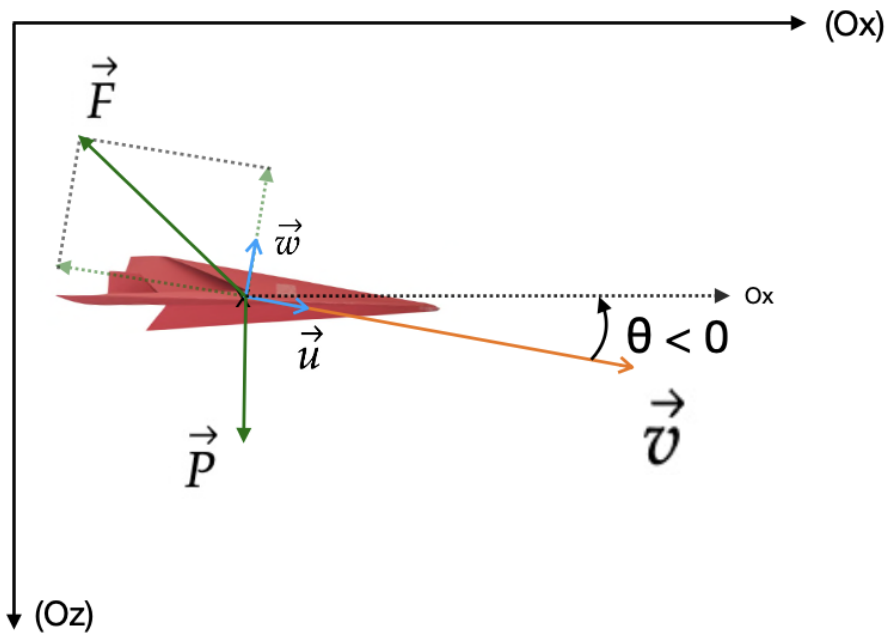


Figure 3: *Scheme of Physics Model Formalization*

The aerodynamic forces acting on the airplane are given by:

$$F_u = -\frac{1}{2}\rho_{\text{air}} S V^2 C_x$$

$$F_w = \frac{1}{2}\rho_{\text{air}} S V^2 C_z$$

The fundamental equations of motion are derived from Newton's second law:

$$m\dot{v} = -\frac{1}{2}\rho S V^2 C_x - mg\sin(\theta)$$

$$-mv\dot{\theta} = \frac{1}{2}\rho S V^2 C_z - mg\cos(\theta)$$

From these equations, we can derive expressions for the drag and lift coefficients:

$$C_x = \frac{-2m(\dot{v} + g\sin(\theta))}{\rho S V^2}$$

$$C_z = \frac{-2m(v\dot{\theta} - g\cos(\theta))}{\rho S V^2}$$

The physical model is based on aerodynamic principles and involves defining constants such as mass, surface area, drag coefficients (C_x, C_z), and initial conditions. The model calculates aerodynamic forces and acceleration, and simulates the trajectory over time using numerical integration.

3.1.3 Drag and Lift Coefficients

The drag and lift coefficients were determined experimentally from the trajectories. The average values obtained are:

$$C_x = 0.77 \pm 0.08$$

$$C_z = 3.01 \pm 0.06$$

These values were used in the physical model to simulate the trajectories.

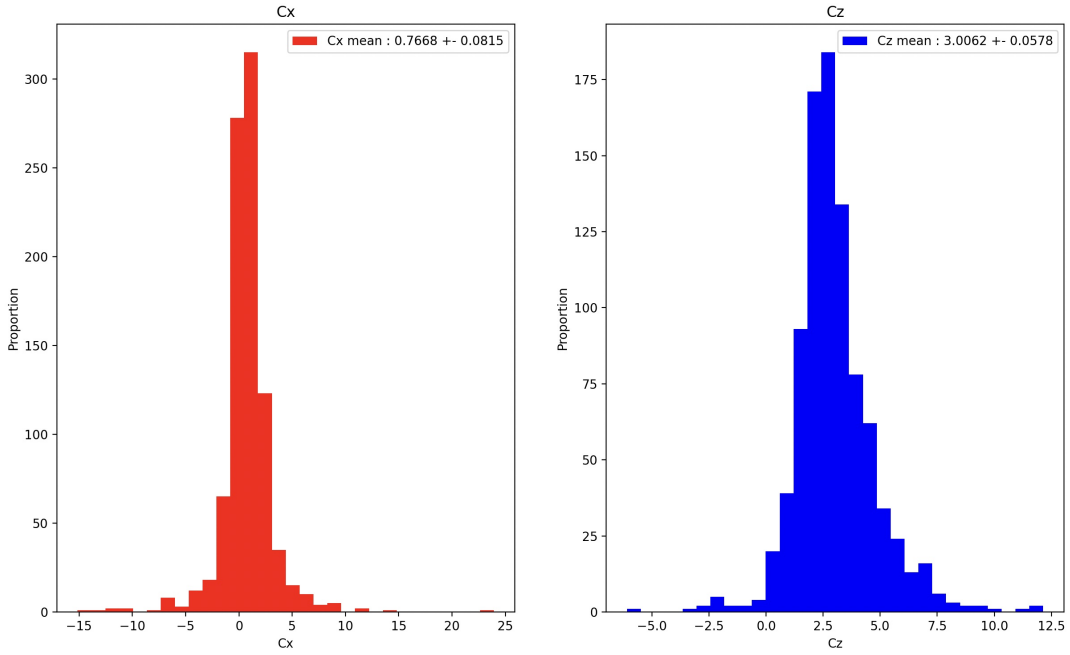


Figure 4: *Cx & Cz are calculated using all the trajectories available and the Euler's method to determine speed and acceleration along the trajectories.*

3.1.4 Force Balance and Integration

The forces are decomposed into horizontal (e_x) and vertical (e_z) components:

$$m\vec{v}_x = F_{\text{drag}} \cdot \cos(\theta) + F_{\text{lift}} \cdot \sin(\theta)$$

$$m\vec{v}_z = -F_{\text{drag}} \cdot \sin(\theta) + F_{\text{lift}} \cdot \cos(\theta) - mg$$

The trajectory is integrated over time using numerical methods. The integration process involves updating the position and velocity at each time step:

$$\vec{a}(t + \Delta t) \approx \frac{\vec{v}(t + \Delta t) - \vec{v}(t)}{\Delta t}$$

$$\vec{v}(t + \Delta t) \approx \frac{\vec{x}(t + \Delta t) - \vec{x}(t)}{\Delta t}$$

The discrete time integration is performed as follows:

$$x_{i+1} = x_i + v_i \Delta t + \frac{1}{2} a_i (\Delta t)^2$$

$$v_{i+1} = v_i + a_i \Delta t$$

3.1.5 Python Implementation

The following Python functions are used to calculate the speed, acceleration, and integrate the trajectory:

```python
def getSpeed(traj, i):
    dt = traj['t'][i] - traj['t'][i-1]
    vx = (traj['coordx'][i] - traj['coordx'][i-1]) / dt
    vy = (traj['coordy'][i] - traj['coordy'][i-1]) / dt
    return (vx, vy)

def getAcceleration(traj, i):
    dt = traj['t'][i] - traj['t'][i-1]
    v2 = getSpeed(traj, i)
    v1 = getSpeed(traj, i-1)
    ax = (v2[0] - v1[0]) / dt
    ay = (v2[1] - v1[1]) / dt
    return (ax, ay)

def integrate(p, v, angle, dt, duration):
    t = 0
    trajectory = {'position': [p], 'angle': [angle], 't': [t]}
    while t < duration:
        a = force(v) / mass
        t = t + dt
        v = v + a * dt
        p = p + v * dt
        angle = angle + angular_v * dt
        trajectory['position'].append(p)
        trajectory['angle'].append(angle)
        trajectory['t'].append(t)
    return trajectory
```

Listing 1: Speed and Acceleration Calculation

3.2 AI Model

The AI model uses a Multi-Layer Perceptron (MLP) Regressor from the Scikit-learn [3] library to predict the trajectory of paper airplanes.

3.2.1 Data Preparation

The input data consists of sequences of points, where each point is defined by its coordinates (x, y), angle (theta), and time (t):

$$p_1 = (x_1, y_1, \theta_1, t_1)$$
$$p_2 = (x_2, y_2, \theta_2, t_2)$$
$$p_3 = (x_3, y_3, \theta_3, t_3)$$

The goal is to predict the next point $p_i = (x_i, y_i, \theta_i, t_i)$ in the sequence.

3.2.2 Data Normalization

Data normalization [7] is crucial for training neural networks. The min-max normalization technique is used to scale the data to a fixed range, typically [0, 1]:

$$x_{\text{normalized}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

Here is the Python function for normalization [7]:

```
def normalisation(trajs):
    x_max, x_min = max(trajs['x']), min(trajs['x'])
    y_max, y_min = max(trajs['y']), min(trajs['y'])
    for traj in trajs:
        traj['x'] = (traj['x'] - x_min) / (x_max - x_min)
        traj['y'] = (traj['y'] - y_min) / (y_max - y_min)
    return trajs
```

Listing 2: Data Normalization

3.2.3 Model Training

The MLPRegressor is trained using the following parameters:

```
regn = MLPRegressor(
    hidden_layer_sizes=(50, 50),
    random_state=3,
    max_iter=8000,
    tol=1e-12,
    activation="logistic",
    solver='adam',
    learning_rate='adaptive',
    shuffle=False,
    epsilon=1e-8
).fit(X_train, y_train)
```

Listing 3: MLPRegressor Training

The model uses gradient descent [6] to minimize the loss function, which is the sum of the squared differences between the predicted and actual values:

$$\text{Loss} = \sum_i (y_i - y_{i,\text{predict}})^2$$

The AI model, specifically a neural network, requires a large amount of training data. The model is trained to predict the fourth point in a sequence of three points, and uses auto-regression [8] to predict the complete trajectory.

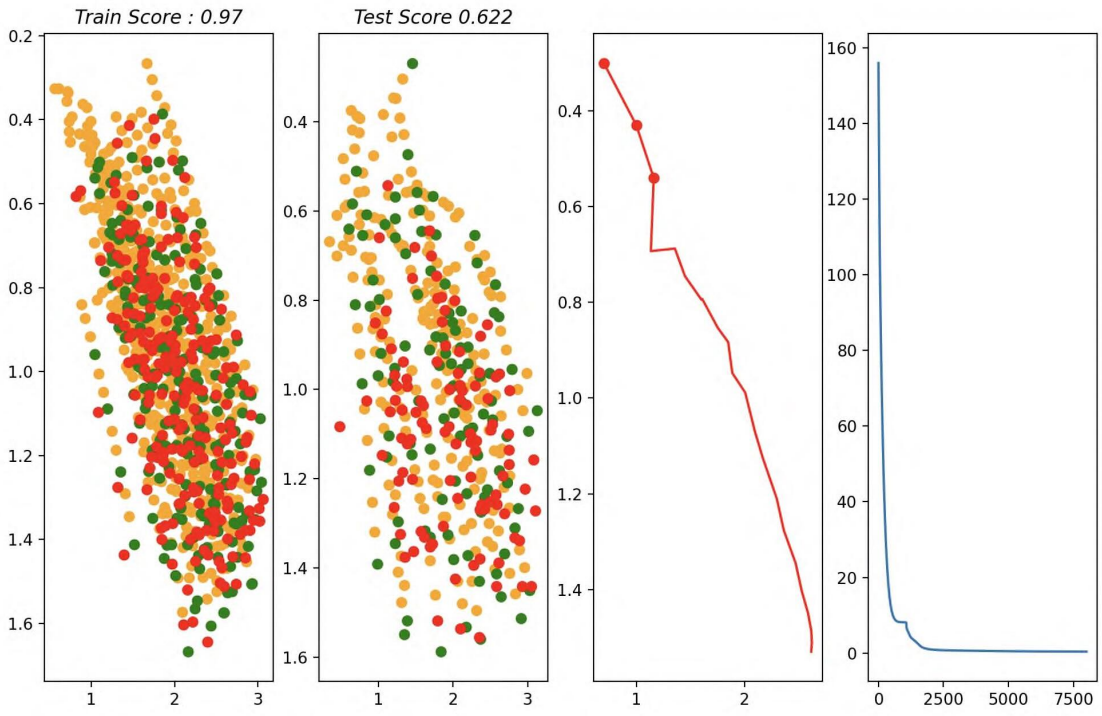


Figure 5: *Training of MLPRegressor : The AI model use the MLPRegressor of the python library SKlearn. Here are represented from left to right the train result on the train dataset, the test result of the model on the test dataset, a prediction by auto-regression of a trajectory to verify the physical coherence of the model, and the evolution of the loss function during the training. For the train and test plot, orange points are the 3 initials points, the green are the real 4th point and the red are the prediction of the model for the 4th point.*

3.2.4 Auto-Regression

Auto-regression [8] is used to predict the complete trajectory. Given an initial set of points, the model predicts the next point, which is then added to the set of points to predict the subsequent point, and so on.

Here is the Python function for auto-regression:

```
def auto_regression(initial_points, dt, duration):
    trajectory = [initial_points]
    t = 0
    while t < duration:
        next_point = predict_IA(initial_points, t)
        trajectory.append(next_point)
        initial_points = initial_points[1:] + [next_point]
        t = t + dt
    return trajectory
```

Listing 4: Auto-Regression Function

3.2.5 Model Evaluation

The score function measures the performance of the model by comparing the predicted values to the actual values. The score is defined as:

$$\text{score} = \left(1 - \frac{u}{v}\right)$$

Where:

$$u = \sum_i (y_i - y_{i,\text{predict}})^2$$

$$v = \sum_i (y_i - y_{\text{mean}})^2$$

4 Comparison and Discussion

4.1 Verification of Model Consistency

The models were verified by comparing their predictions with real trajectories.

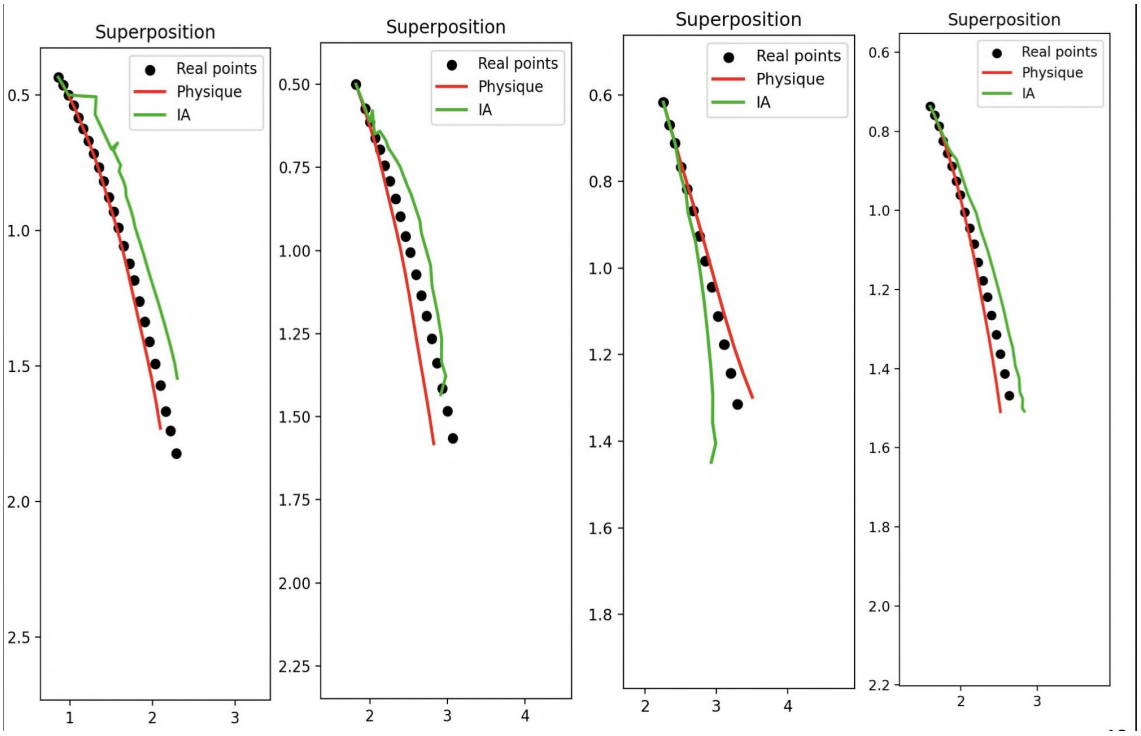


Figure 6: *Models Coherence Verification : To ensure the validity of the models, here are plotted the comparison of the prediction of the AI model (green), the physical model (red) and the real trajectory measured (black points).*

4.2 Model Comparison

The comparison involved:

- Measuring the distance between real and predicted trajectories.
- Evaluating the error of the models to study their accuracy and the robustness.
- Evaluating the influence of the amount of training data on model accuracy.

4.2.1 Error Quantification

It is important to quantify the error of the models. For this, a distance or error function is introduced, which is the average distance of the predicted points to the real ones:

$$d_i = \sqrt{(x_{\text{predict},i} - x_i)^2 + (y_{\text{predict},i} - y_i)^2}$$

$$\text{Error} = \frac{1}{n} \sum_{i}^{n} d_i$$

4.3 Accuracy

With the drag coefficients determined earlier and an AI trained with the 77 'classic' trajectories, both models were asked to predict the complete trajectory that would follow and measure their error. The distribution of errors was observed, and it was noted that both remain competitive in the total amount of error. Nevertheless, the distribution is not the same : AI Model makes less large errors but more medium ones while the Physic Model make more little error but also some larger errors.

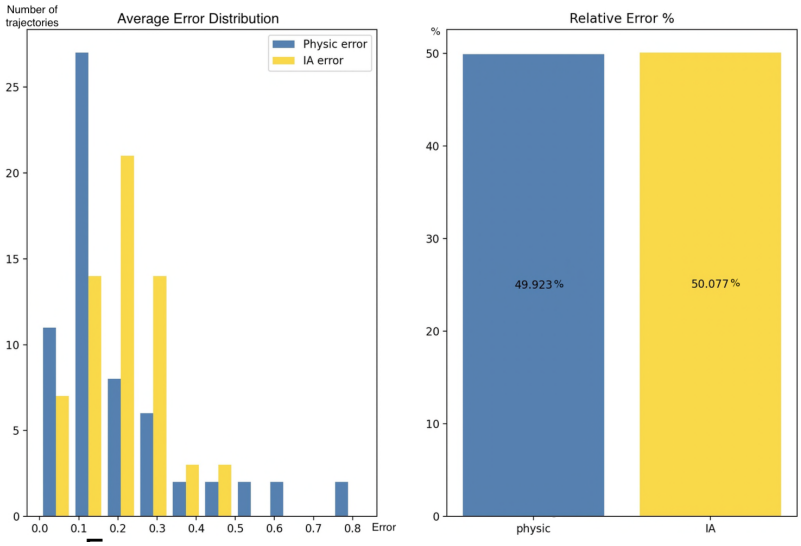


Figure 7: *Average error distribution of the models prediction on 'classic' trajectories (left). Relative total error between the two models in % (right).*

4.4 Robustness

Keeping the same C_x and C_z, as well as the same training for the AI, both models were asked to predict the continuation of the 13 more original measured trajectories, such as one with an ascending start. It was immediately seen that the AI predictions were way off while the physics remained consistent. It was measured that in this case, the AI makes twice as much error as the physics on average, which is significant.

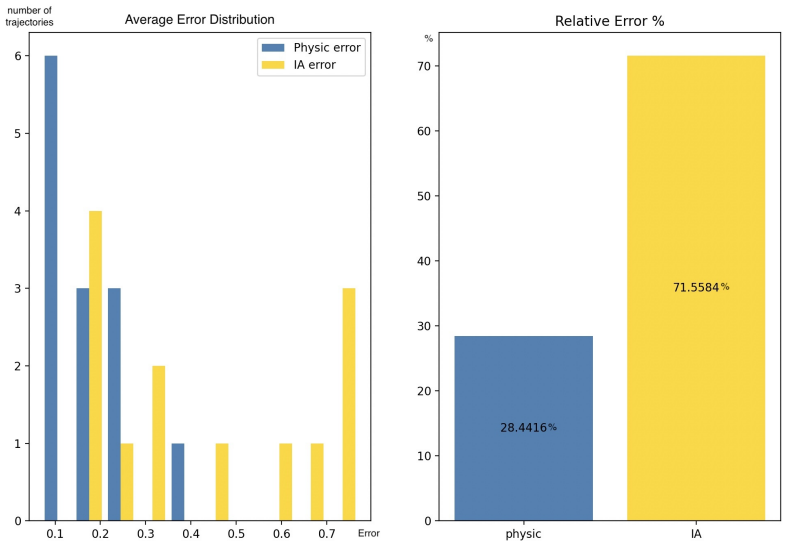


Figure 8: *Average error distribution of the models prediction on 'original' trajectories (left). Relative total error between the two models in % on their prediction (right).*

4.5 Influence of the Amount of Data

By varying the AI training, its performance changes a lot. A key factor is the amount of data used to train the model. To observe this, the relative error between the two models on the classic trajectories was plotted as a function of the proportion of data used to train the AI. Here, the physical model was unchanged; the C_x and C_z determined with all available trajectories were kept.

When the two curves meet at 50%, it means that both models make on average the same amount of errors. An important point is that the influence of the amount of data, not the data itself, is considered. Therefore, to see this well, the experiment was carried out several times, reshuffling the data each time. A decreasing trend in the relative error of the AI compared to physics was observed. With this trend, it was estimated that at least 120% of the available amount of data, which corresponds to 92 trajectories, would be needed for the AI to equal the physical model. However, this must be qualified, as with more trajectories, the precision on C_x and C_z might be improved.

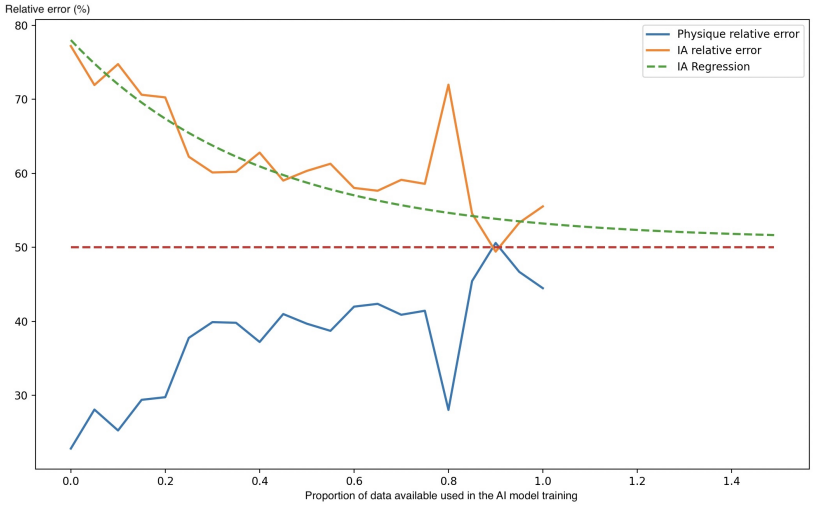


Figure 9: *Influence of the Amount of data (% of available) on the relative error between the two models (Cx/Cz fixed). The more data are used to train the model, the less error it makes. With all the data available, the two model seems equivalent on average. There is not enough data to see the curves crossing or not. Further investigation must be perform to see the real trend.*

On the other side, the amount of data also change the precision of the Cx and Cz determined. To reach the asymptote of the Cx and Cz, a minimum of 50% of the data available is needed, which corresponds to 38 trajectories.

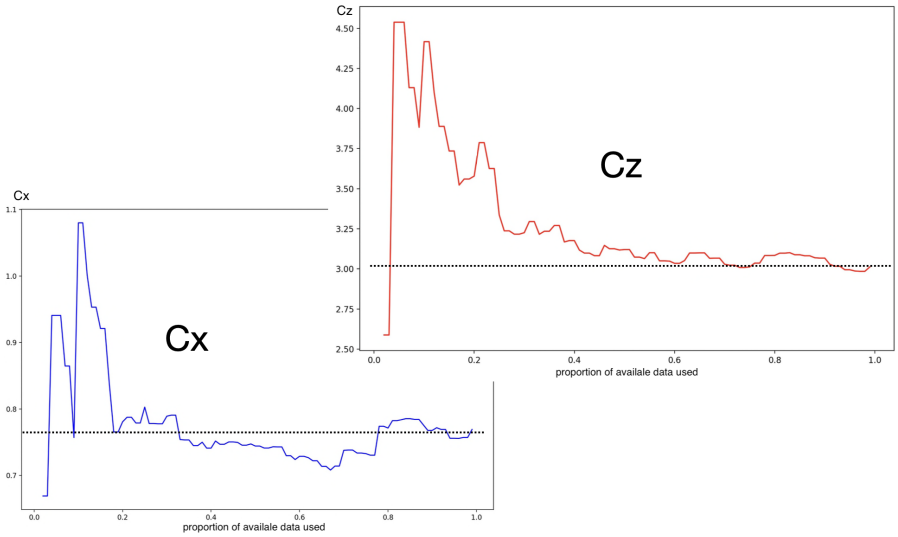


Figure 10: *Influence of the Amount of data to determine Cx & Cz : the more data are available, the more accurate are the Cx and Cz. It seems that the asymptote is reached with approximately 50-75% of the data available.*

5 Results

5.1 Physical Model

The physical model, based on aerodynamic principles, is more generalizable but requires precise theoretical understanding. It relies on theoretical knowledge and coefficients like drag and lift and involves defining constants such as mass, surface area, and initial conditions. The model calculates aerodynamic forces and acceleration, and simulates the trajectory over time using numerical integration.

5.2 AI Model

The AI model, specifically a neural network, excels within its training domain but requires extensive data. The model's accuracy is highly dependent on the amount and quality of training data.

The model is trained to predict the fourth point in a sequence of three points, and uses auto-regression [8] to predict the complete trajectory. The AI model is a viable alternative but is limited by the available data, whereas the physical model needs precise theoretical understanding.

6 Conclusion

6.1 General Conclusion

The project concludes that AI models are viable for specific cases where it's easy to collect lot of data. The physical model, on the other hand, demands a deeper theoretical understanding but is more generalizable.

The number of data is very important for both models: for the AI training and the determination of the physical characteristic such as C_x and C_z.

The model developed by AI remains viable in a restricted framework for predicting the trajectories of paper airplanes. Its reliability and ability to generalize are limited by the available data. The physical model requires a real understanding of the paper airplane flight physics. In some cases, situations are too complex to model or poorly understood, and an AI model is a viable alternative.

6.2 Openning : Hybrid Models

A hybrid approach where AI works with the physical model might be interesting. Using an AI model to correct a physical prediction for exemple.

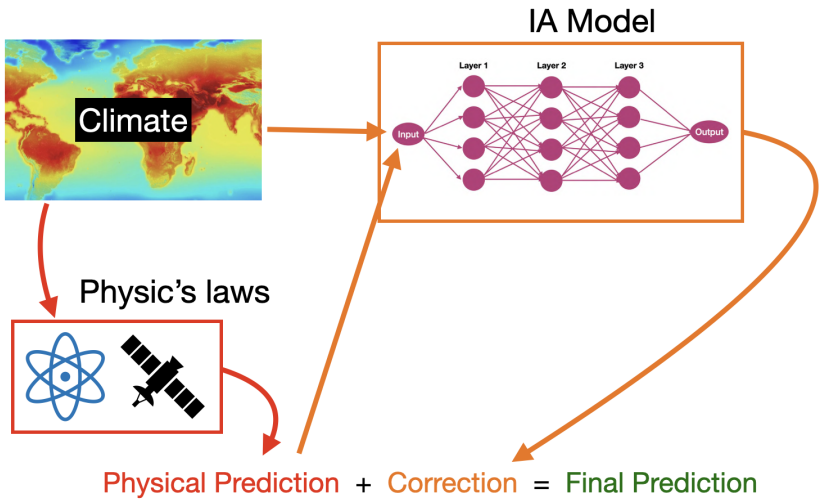


Figure 11: *Idea : Hybrid Model for weather prediction with an AI trained model on many data which could provide small correction to a physical model.*

7 Python Codes

7.1 ModelAI.py

This script trains a neural network to predict trajectories using a Multi-Layer Perceptron (MLP) Regressor.

7.2 aero.py

This script simulates the trajectory using physics-based modeling.

7.3 Data_Analyser.py

This script processes video footage to extract trajectory data.

7.4 Analyser.py

This script compares the performance of the physics and AI models.

8 Project Timeline

The project timeline includes several key milestones:

- July 2023: Meeting with Christophe Airiau (Researcher in Fluid Mechanics) to define scientific questions and establish an experimental protocol.
- July 2023: First experiment, recording videos of ball trajectories. Initial approach to trajectory extraction using Python and OpenCV. Results were inconclusive due to noise and insufficient frames per second.
- October 2023: Second attempt at recording video trajectories at 60fps with better lighting. Discussions with Grgur Kovac (AI Researcher at Inria) on Python image analysis tools, leading to the use of YOLOv8. Successful extraction of paper airplane trajectories.
- October - December 2023: Second and final recording of paper airplane trajectories. Development of trajectory analysis program, encoding and decoding, normalization [7], filtering of points, Euler [5] method, and database construction.
- December 2023 - February 2024: Familiarization with the Python Sklearn library and MLPRegressor (Multi-Layer Perceptron Regressor) for implementing a neural network. Training the AI and developing the auto-regression [8] function.
- February 2024: Evaluation of drag and lift coefficients for the physical model using the recorded trajectories. Development of numerical simulation based on physical laws.
- March - April 2024: Verification of model consistency by comparing them to reality. Evaluation of model performance, study of their precision and robustness.

9 acknowledgements

- Christophe Airiau for his help in this project. He helped me to established the direction of the project and many precious methodological advices.
- Olivier Gally, my physics teacher for his advices and his corrections.
- Emmanuelle Schmidt, my engineering science professor for her advices and support.
- Pierre-yves Oudeyer, for his support, his methodological advices and his review of the report.

Note: This document is a synthesis and translation of many documents initially in french. Translation made with the assistance of AI tools.