FPCA applied to flight paths optimization

Lucas Ligny
Ecole Nationale de l'Aviation Civile (ENAC)
Toulouse, France
ligny.lucas9@gmail.com

***Abstract*—In this paper, we detail the steps that lead to optimized trajectories according to a selected criterion, in a low dimensional space. After presenting the main techniques for optimizing flight paths, as well as methods for reducing the size of the state space, we precise the modeling of our problem. We use the Karhunen-Loève transformation, or Functional Principal Components Analysis (FPCA), as our main tool to model the state space. We also select the constraints undergone by our airplane: here, we decide only to consider the impact of the wind. For its simplicity, the Simulated Annealing (SA) is chosen in order to find the optimized trajectory. Thus, once the modeling is finished, we launch our simulations and proceed to an analysis of our results.**

Keywords-FPCA; Simulated Annealing; Monte-Carlo; Flight path; Approach procedure

I. INTRODUCTION

Air Traffic Management (ATM) ensures the safety of flight by optimizing flows and maintaining separation between aircrafts. Optimizing flight paths is also a crucial issue for an airline, in order to save fuel and / or flight time. Most of the time, aircrafts positions are represented as radar plots so that many trajectory statistics conducted in ATM are spatial and do not consider the time dependence anymore. Moreover, the collection of radar plots describing the same trajectory can have a lot of redundant samples. From the trajectory design point of view, this redundancy is real handicap for the optimization process. Our idea is to find an alternative trajectory representation to eliminate this redundancy. Considering the work of Delahaye *et al.* [1], we choose the FPCA method to represent our state space. FPCA consists in decomposing the trajectory on a space where each dimension maximizes the variance, so that one can recreate a faithful approximation of the trajectory thanks to only a few components. This method is used in [2] to develop models for Monte Carlo simulations. While this paper also presents a Monte Carlo algorithm, it intends to go a little further by using SA, a powerful optimization tool. SA is one of the best known and simplest metaheuristics, which is widely used for real-life applications. Its interest regarding trajectory optimization is studied in [3]. Coupling FPCA with SA seems a really good idea, knowing that we must generate neighboring states at each iteration of the SA algorithm, process which would have been quite long if we had not diminished the size of our state space in the first place.

II. STATE OF THE ART

A. Optimization methods

1) Fast Marching

The Fast Marching method, introduced in [4], is a classic optimization method which suits particularly well to the study of flight paths. The principle consists in studying a wave front, monitoring its evolution and determining the minimum cost to reach any point in space. As a reminder, in our study, the cost corresponds to the time and / or fuel consumed.

This method is applicable if the speed of propagation of the front only depends on the position and remains of the same sign. Therefore, the case of fuel is more difficult to manage because the speed of propagation of the front would depend on the position and the direction of the wind. Algorithms, called Ordered Upwind, have been developed to deal with this situation, but their algorithmic complexity is higher.

In the case where the method is applicable, the calculation of the minimum cost C to reach any point in space from a starting point consists in solving the eikonal equation $|\nabla C(x)| = 1 / F(x)$ with $F(x) \geq 0$ and $C(x_{init}) = 0$, where $x \in \mathbb{R}^2$ represents the position in space, $C(x) \in \mathbb{R}$ the minimum cost to arrive in x from x_{init} and $F(x) \in \mathbb{R}$ the known propagation speed at any point x. This equation is a partial differential equation and can be easily solved with classical techniques.

Finally, the gradient descent method allows us to obtain the optimal trajectory between the arrival point and the departure point.

2) Dynamic programming

Dynamic programming was developed by Bellman, in [5], to solve problems of optimal paths. The principle of the method is to associate a notion of state for each problem. Each state is associated with an optimal value, and the dynamic programming equation links the value of a state at a given instant to those of the states that can be accessed at the next instant.

Bellman introduces the following principle of optimality which is at the basis of the method: An optimal solution for the problem contains the optimal solutions for all the sub-problems. For example, in the context of finding the shortest path, this principle is illustrated as follows: if (C) is an optimal path going from A to B and if C $\in$ (C) then the (C) sub-paths from A to C and from C to B are optimal. Thus, if a path is optimal, then it is formed of optimal sub-paths.

In the end, the problem is divided into sub-problems sequentially, and the resolution is performed recursively in order to produce the solution to the global problem. An example of an algorithm using the principle of dynamic programming is the Dijkstra algorithm.

3) Simulated annealing

The simulated annealing algorithm is a special algorithm. Indeed, it is an empirical algorithm, strongly inspired by the physical phenomenon of annealing [6].

To build this algorithm, we imagine that the function to be optimized is assimilated to the energy of a fictive physical system. At initialization, our physical system is in the state x_0 and has an energy E_{init} at temperature T_{init}. The energy of the system corresponds to the function for which the minimum is sought (cost function). The temperature is a parameter of the algorithm, which is initially set quite high.

Subsequently, we generate a neighboring state $x_1 = x_0 + \delta x$ of energy E_1, with δx a random variation. This state is accepted according to the Metropolis criterion. According to the laws of statistical physics, the probability of finding our system in the state x_1 is $\exp[-(E_1 - E_0) / k_B T]$ with k_B the Boltzmann constant. Therefore, we accept our new state if it decreases the energy of our system, but also if it increases it with a non-zero probability: we agree to degrade our solution, to avoid falling into a local minimum. We carry out this operation until the thermal equilibrium for the given temperature. Once equilibrium is reached, the temperature is lowered and the search for thermal equilibrium is restarted with the new temperature. The algorithm stops after a certain number of iterations, or when a minimum temperature T_{final} is reached. Thus, we reach the minimum energy state of the fictive physical system and find the minimum of our function.

This algorithm is both powerful and complex, because of the multitude of parameters to be taken into account. In particular, it is necessary to choose the initial temperature of the system, its law of decay, the final temperature or the number of iterations, but also the definition of a neighboring state and thermal equilibrium.

B. Dimension reduction

The principle of dimension reduction of our state space consists in expressing our trajectories on a basis of small dimension, of the form $\gamma(t) = \sum_{i=0}^{n} a_i h_i(t)$.

1) B-Splines approximation

Bézier curves are parametric polynomial curves described by Pierre Bézier in [7]. They require the introduction of n+1 points of the plane or space, called control points which we note $P_0, P_1, P_2, \ldots, P_n$.

We define the Bézier curve associated with these control points by $S(t) = \sum_{k=0}^{n} B^n_k(t) P_k$ with $0 \leq t \leq 1$. The $B^n_k(t) = C^n_k t^k (1-t)^{n-k}$ are the polynomials of the Bernstein basis. The Bézier curve has multiple properties, it is in particular of class C^∞ and is located in the convex envelope of the control points. In addition, we have $\mathbf{P_i P_{i+1}}$ as the tangent vector to the curve in P_i, $\forall i \in [[0, n]]$.

To generalize this concept, we introduce the notion of nodes, noted t_i, such as $0 \leq t_0 \leq \ldots \leq t_i \leq \ldots \leq t_m \leq 1$. We define the B-Spline curve of degree i (presented in [8]) associated with the control points $P_0, P_1, P_2, \ldots, P_n$, with $m \geq n + 1 + i$, by $S(t) = \sum_{k=0}^{n} B_{k,i}(t) P_k$ with $t_i \leq t < t_{n+1}$, composed of B-Spline functions of degree i defined by recurrence :

- for $0 \leq k \leq m - 1$, $B_{k,0}(t) = 1$ if $t \in [t_k , t_{k+1}[$, $B_{k,0}(t) = 0$ otherwise ;
- for $i \leq 1$ and $0 \leq k \leq m - i - 1$, $B_{k,i}(t) = [(t - t_k) / (t_{k+i} - t_k)] B_{k,i-1}(t) + [(t_{k+i+1} - t) / (t_{k+i+1} - t_{k+1})] B_{k+1,i-1}(t)$.

The Bézier curve associated with n+1 control points corresponds to the B-Spline curve of degree n having for nodes the points $t_0 = t_1 = \ldots = t_n = 0$ and $t_{n+1} = t_{n+2} = \ldots = t_{2n+1} = 1$.

To approximate a function f of class C^2 on [a, b] by a B-Spline curve, we can introduce the nodes t_k so that $t_0 = t_1 = \ldots = t_i = a < t_{i+1} \leq \ldots \leq t_n < b = t_{n+1} = \ldots = t_{n+1+i}$. We have $\gamma(t) = \sum_{k=0}^{n} f(t^*_k) B_{k,i}(t)$ with $t^*_k = (t_{k+1} + \ldots + t_{k+i}) / i$.

2) FPCA - Karhunen-Loève theorem

The Karhunen-Loève transformation is a method which consists in transforming correlated variables into new variables, our $h_i(t)$ called principal components, decorrelated from each other.

This transformation is mainly used to describe and visualize data. This transformation allows each component to maximize the available variance, which is extremely interesting for the study of flight paths because it allows us to have the most faithful approximation of the path by minimizing the number of coefficients to consider. The mathematical principle is quite simple: first, you must calculate the average of each dimension of the data set. It is also necessary to extract the covariance matrix, as well as its eigenvectors and its eigenvalues. Then we simply have to sort the eigenvectors in decreasing order of eigenvalues and to choose the n first eigenvectors, which constitute our matrix of passage towards the basis of Karhunen-Loève [9].

This analysis is suitable for small data tables, but do not by itself allow the generation of acceptable trajectories due to problems of data regularity. It is necessary to make this analysis functional, that is to say by no longer considering data tables, but functions: this analysis is called FPCA. In this continuous case, we have the expression of the following Karhunen-Loève theorem:

If the stochastic process X_t is centered and of continuous covariance K_X, then X_t admits a decomposition of the form $X_t = \Sigma^{\infty}_{k=0} Z_k e_k(t)$ with $e_k(t)$ the eigenfunctions of T_{Kx}, where T is the integral operator. We notice that by sorting the eigenvectors and keeping only the first ones until n, we have a decomposition of $\gamma(t)$.

III. MATHEMATICAL MODELING

A. Space State

We have at our disposal a fairly substantial database made of tens of thousands descent trajectories. Our goal is to manually generate similar trajectories: several problems then arise, in particular the choice of the approach procedure. Indeed, airport approach procedures allow airplanes to use one or more IAFs (Initial Approach Fix), and trajectories of our airplanes can be different depending on the approach considered. Therefore, a clustering phase is necessary: it makes discrimination of descent trajectories possible according to the chosen IAF. Thus, all of our trajectories behave more or less the same way and we can generate a state space that makes sense.

Another problem concerns the implementation of our principal component analysis. As explained above, it is mandatory to have our analysis functional. A prior conversion of our trajectory data into splines is a good idea to allow us to apply the Karhunen-Loève transformation. Therefore, we must fix the cumulative variance that we wish to obtain, that is to say the level of fidelity of curves generated compared to our data. With too low a fidelity, we create inflexible trajectories and the optimization is of little interest. With too much fidelity, not only do we dwell on details that slow down our algorithm, but we take into account unconventional trajectory behaviors, which we seek to avoid. 95% is a value which is acceptable *a priori*, which we keep for our simulations.

In Fig.1, we decompose our trajectories (nearly 20000 trajectories after clustering) on the Karhunen-Loève basis. Only 15 components are sufficient to describe 95% of the data. We show the probability densities linked to the first four coefficients, as well as 1000 random variates within the densities.

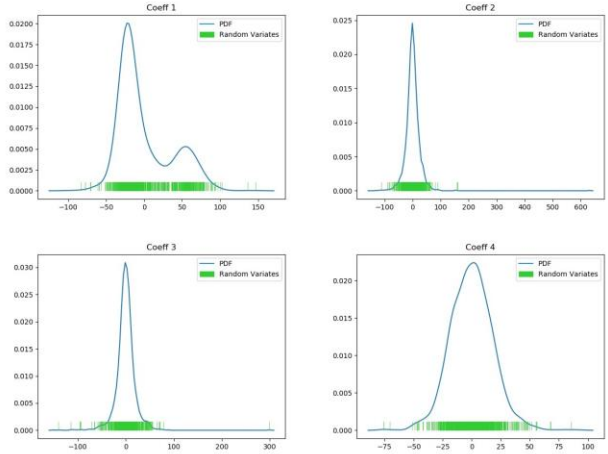

Figure 1. Probability Density Functions

We have seen that the Karhunen-Loève transformation makes it possible, from a set of data, to generate a space of small dimension. Indeed, by fixing the cumulative variance, we determine the number of components of our decomposition. Each component has its own probability density, and a trajectory is generated by a random realization of each coefficient. By randomly choosing a large number of coefficients in each probability density, we generate a multitude of random trajectories and we find the state space in which our functions live.

In Fig. 2, on the left, we have the database on which our study is based. On the right, we reconstruct 50000 trajectories on the Karhunen-Loève basis. We note that the state space is faithful and that we get rid of "unconventional" trajectories.

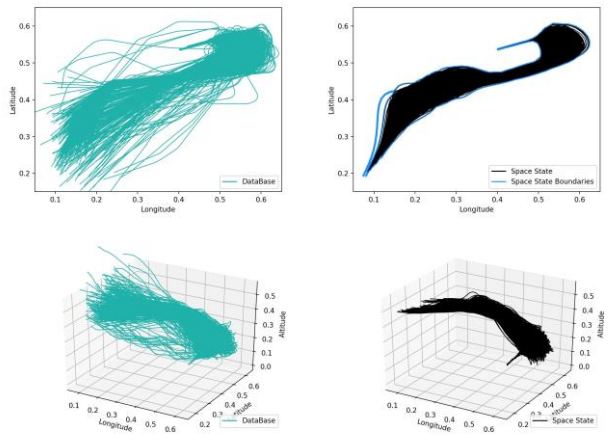

Figure 2. State Space

B. Constraints

After generating our "free" flight paths, we decide to constrain them. Optimization without constraints could be considered, but in reality our planes are subjected to several types of constraints of different natures. At first, we could force our trajectories to go through certain waypoints. In reality, an approach procedure is characterized by the IAF, but the aircraft must also pass through the IF and the FAF, intermediate and final approach marks. However, these requirements are not very restrictive because all of our generated trajectories have the same behavior as the initial trajectories, and consequently they naturally pass through the waypoints. Obstacle constraints can also be studied. A quick and effective solution to avoid obstacles is to adapt our state space, by removing the trajectories that pass through the obstacle. Finally, we decide not to study the constraints aiming to separate the aircrafts from each other for complexity reasons.

The constraint used in this paper is the wind field. Indeed, this constraint is an important factor in minimizing the fuel consumed, and significantly modify our approach.

C. Objective

The energy of our trajectory, or our objective function, is the criterion that we seek to optimize. In the case of descent trajectories, the criterion retained can be the shortness of the trajectory (case without constraints) or the fuel consumed (case with constraint of wind field). The fuel consumption can be easily linked with the wind by a proportionality relationship. In our simulations, the constraint retained is the opposite of the scalar product between the wind vector and the tangent vector to our trajectory at a spatial point considered.

When cruising, to optimize the fuel consumed, we put the airplane in effective rear wind (this allows us to reduce our flight speed while keeping the maxi range constant). Wind forecasts can be found in WINTEM maps. Trends are issued every 3 hours for France, and every 6 hours for the European domain. We imagine that our plane seeks to save fuel on approach, in the same way as it would on a cruise.

By using a Monte-Carlo type approach, that is to say by simulating random trajectories in the state space and by evaluating their energy at each iteration, we can easily determine the shapes of the trajectories that optimize our criterion.

In Fig. 3, we simulated 10000 flight paths in state space. Note that the purple curves (the most concise) are the ones that take the shortest turns on approach.

In Fig. 4, we simulated 10000 flight paths in the state space. On the left, we notice that the curves are not very different in energy: planes take the wind in the same way. On the right, gusts of wind on final approach encourage the pilot to take the shortest turn, to save fuel.

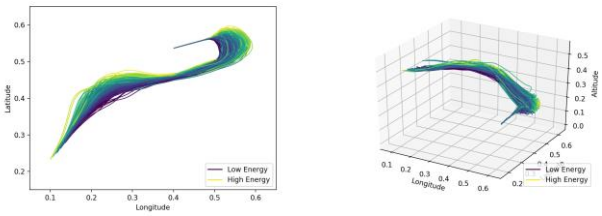

Figure 3. Flight paths ranked according to their shortness

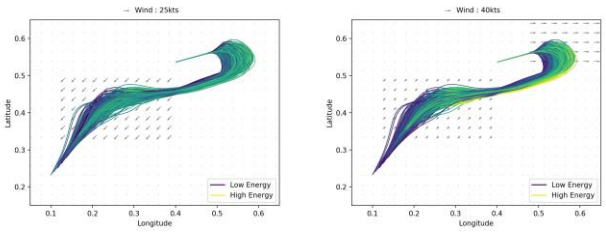

Figure 4. Flight paths ranked according to their fuel saving

However, this method is not perfect. It can be quite costly in time if we are looking for a precise solution, and only gives us an imprecise approximation of the optimal trajectory if we are looking for a quick solution. A well-known optimization technique will allow us to extract the optimal trajectory by combining these two criteria: speed and precision.

D. Simulation

Our simulation consists in generating a flight path in our state space. The coefficients of this trajectory constitute the input of our simulated annealing. The initial temperature is fixed in a way that the first neighboring state is accepted in 99% of the cases. We choose a geometric decrease of the temperature, parameter 0.99. In addition, a maximum number of iterations of 4000 has been chosen to ensure that the thermodynamic equilibrium is reached at each level. To generate the neighboring state, we choose to make a small Gaussian variation of each coefficient, taking care to remain in the state space.

IV. RESULTS

Results are shown in Fig. 5. We generated in about 2000 iterations of the simulated annealing algorithm the optimal trajectory for the favorable (left) and unfavorable (right) wind field.

We note that the simulated annealing method is faster, we only needed 2000 iterations of the algorithm to generate the optimal trajectory, against 10000 iterations of generation of random trajectories. An iteration of one or the other algorithm takes the same time to execute (around 15ms), which mainly consists in generating a new trajectory and evaluating its energy.

It is also more precise, since the trajectories generated are much lower in energy (123% better than the one given by the Monte-Carlo method on the left, and 452% better on the right). In addition, they have the expected behavior: the trajectory is perfectly collinear to the wind when it is favorable, and is almost orthogonal to it when it is unfavorable.

A criticism that we could make is that our trajectories leave our state space, or more precisely they are at the limit of our state space: the random draw in our probability densities is mainly marginal. If we want to force our optimized trajectories to remain in our "classic", that is to say non-marginal, state space, it is possible to constrain our probability densities by forcing our annealing to generate probable trajectories only. Another solution would be to directly alter our state space, by modifying the cumulative variance we wish to obtain. If we set the cumulative variance at 80% instead of 95%, we have fewer variable trajectories and therefore better inserted trajectories in our state space.

In Fig. 6, we generated in about 2000 iterations of the simulated annealing the optimal trajectory for the unfavorable wind field. On the left, we cut the lower and upper 10% of each probability density. On the right, we reduced the state space to 80% of the cumulative variance (only 6 components describe 80% of the data). Results are better: we managed to find more realistic trajectories, with a lower energy than those found with the Monte-Carlo method. Thus, we can validate the method using a revised state space to compute our trajectories.

What would happen if, instead of constraining our probability densities, we extended them?

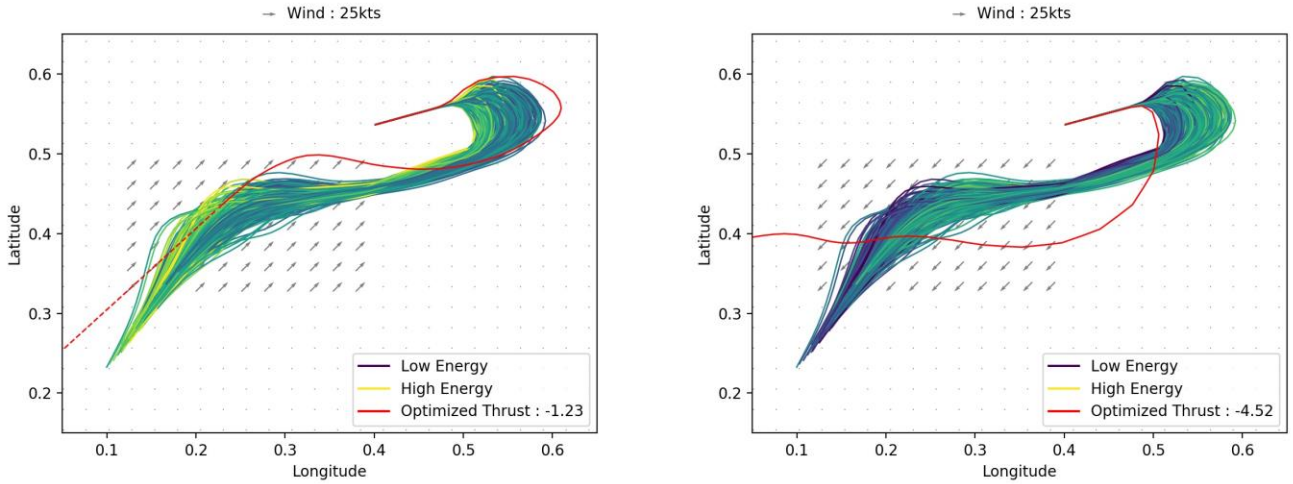

Figure 5. Optimal fuel flight paths

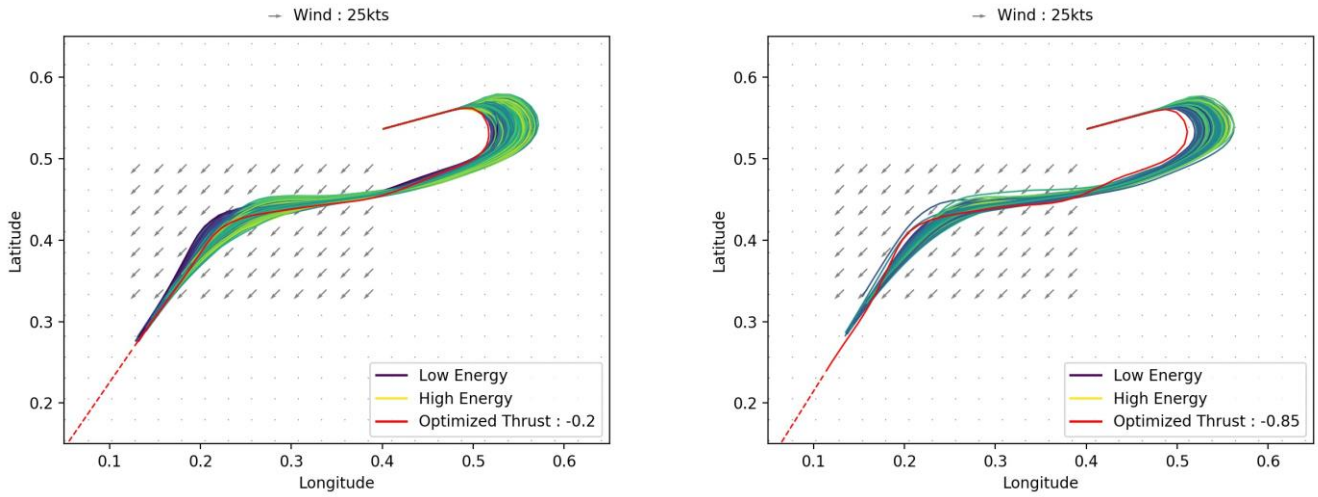

Figure 6. Optimal fuel flight paths (revised state spaces)

So far, we have searched for an optimal trajectory in our state space. Let's look at what happens when we are not seeking to optimize a trajectory, but optimize a state space. For this to work, we modify the average trajectory of our initial state space, which we control using a Monte-Carlo type method or a simulated annealing. This trajectory can move in an enlarged space, by a factor of 5 compared to our initial state space (that is to say that the limits of the probability densities have been extended by a factor of 5, around their median). Our optimization criterion is the direction of the entry point of our trajectories. For example, we no longer want our trajectories to come from the South West, but from the North West.

In Fig. 7, on the left, we have represented the new state space generated by moving the average trajectory using the Monte-Carlo method, and in the center thanks to the simulated annealing. On the right, we have generated 100 average trajectories within our enlarged state space.

We note that we manage to modify the entry of our flight paths at the cost of a U-turn to reach the runway, regardless of the algorithm chosen. In reality, the space of our average trajectories is quite limited, and that is why we cannot generate any trajectory. We note that no average trajectory comes from the North East, therefore it would have been impossible to generate a state space which comes from it. To be able to generate any approach procedure, it would be necessary to be able to modify the structure of our Karhunen-Loève basis.

Nevertheless, this study is quite interesting because it shows the robustness of our transformation: a state space naturally forms around any average trajectory.

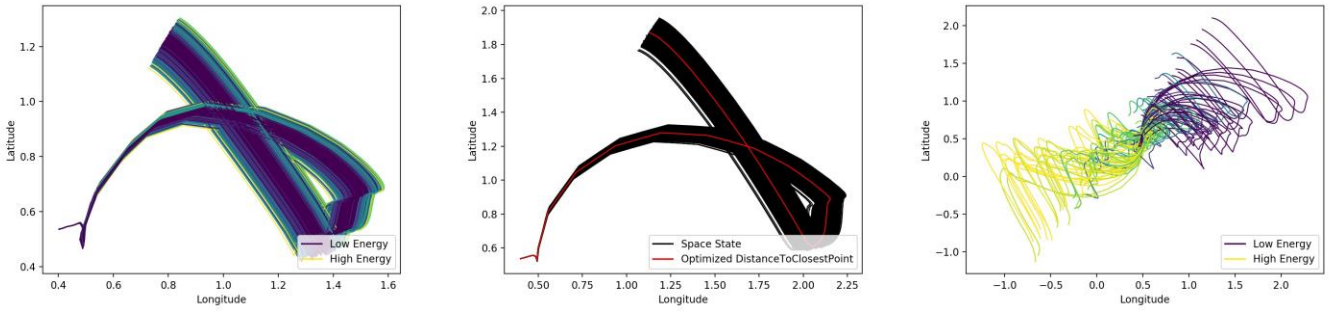

Figure 7. Flight paths from the North West

V. CONCLUSION

The interest of this study was to highlight the robustness of the FPCA in its application to flight paths. Only based on a sample of trajectories already flown, we were able to extract trajectories optimized according to the external wind constraints.

Multiple applications can now emerge from our study. We could imagine that the pilot downloads the wind maps at the end of the cruise, before the approach, and that the algorithm returns the optimal trajectory that should be followed to save fuel for his company. Our study can also very easily extend to climb or cruise trajectories.

Our algorithms can also be reused to solve other problems. Indeed, by taking up the idea discussed at the end of our paper, a new application could be the shifting of the approach procedure. By analyzing all the procedures generated, we can extract the procedure that minimizes noise for residents near the airport.

Therefore, our study consists in creation of more economical and more ecological flight paths, topics which are part of the major challenges of the twenty-first century.

ACKNOWLEDGMENT

I would like to express my very great appreciation to Pr. Daniel Delahaye for his support and his constructive ideas during the development of this research work. His interest in my work and his willingness to give his time have been very much appreciated.

I would also like to thank Gabriel Jarry and Alexis Bernard for their relevant advices and knowledge sharing.

REFERENCES

[1] Daniel Delahaye, Stéphane Puechmorel, Panagiotis Tsiotras, Eric Féron, “Mathematical Models for Aircraft Trajectory Design: A Survey”, EIWAC 2013, 3rd ENRI International Workshop on ATM/CNS, Tokyo, Japan. pp 205-247, Feb 2013.

[2] A. Eckstein, “Data driven modeling for the simulation of converging runway operations”, Proceedings of the 4th International Conference on Research in Air Transportation (ICRAT), 2010.

[3] Daniel Delahaye, Supatcha Chaimatanan, Marcel Mongeau. “Simulated annealing: From basics to applications”, Handbook of Metaheuristics, Springer, pages 1–35, 2019.

[4] J.A. Sethian, “Fast marching methods”. SIAM review, vol. 41, no. 2, pages 199–235, 1999.

[5] R. Bellman, “On the theory of dynamic programming”, Proceedings of the National Academy of Sciences of the United States of America, vol. 38, no. 8, page 716, 1952.

[6] S. Kirkpatrick, C. D. Gelatt & M. P. Vecchi, “Optimization by Simulated Annealing”, Science, vol. 220, no. 4598, pages 671–680, 1983.

[7] P. Bézier, “Courbes et Surfaces”, Hermès, 1986.

[8] C. de Boor, “A practical guide to Splines”, Applied Mathematical Sciences 27 - Springer-Verlag, 1978.

[9] R. A. Johnson and D. W. Wichern, “Applied Multivariate Statistical Analysis 5th Ed”, Upper Saddle River, NJ, Prentice Hall, 2002.