Extending Flat Motion Planning to Non-flat Systems. Experiments on Aircraft Models Using Maple

François Ollivier

Extending Flat Motion Planning to Non-flat Systems. Experiments on Aircraft Models Using Maple

François Ollivier

LIX, UMR CNRS 7161
École polytechnique
91128 Palaiseau cedex
France

francois.ollivier@lix.polytechnique.fr

May, 12th 2022

Abstract. Aircraft models may be considered as flat if one neglects some terms associated to aerodynamics. Computational experiments in Maple show that in some cases a suitably designed feed-back allows to follow such trajectories, when applied to the non-flat model. However some maneuvers may be hard or even impossible to achieve with this flat approximation. In this paper, we propose an iterated process to compute a more achievable trajectory, starting from the flat reference trajectory. More precisely, the unknown neglected terms in the flat model are iteratively re-evaluated using the values obtained at the previous step. This process may be interpreted as a new trajectory parametrization, using an infinite number of derivatives, a property that may be called *generalized flatness*. We illustrate the pertinence of this approach in flight conditions of increasing difficulties, from single engine flight, to aileron roll.

Keywords : flat systems, motion planning, aircraft control, Newton operator, symbolic-numeric computation, generalized flatness.

Résumé. Des modèles d'avions peuvent être considérés comme plats si on néglige certains termes associés à l'aérodynamique. Des expériences de calcul en Maple montrent que dans certain cas, un bouclage convenable permet de suivre de telles trajectoires, en utilisant le modèle non plat. Certaines manœuvres peuvent néanmoins être difficiles, voire impossible à réaliser avec cette approximation plate. Dans cet article, nous proposons un processus itératif pour calculer une trajectoire plus aisée à suivre, en commençant par l'approximation plate de référence. Plus précisément, les termes inconnus négligés dans le modèle plat, sont itérativement réévalués, en utilisant les valeurs obtenues à l'étape précédente. Ce processus peut être interprété comme un nouveau paramétrage utilisant une infinité de dérivées, une propriété qui peut être appelée *platitude généralisée*. Nous illustrons la pertinence de cette approche dans des conditions de vol de difficulté croissante, incluant un vol avec un seul moteur, une descente en vol plané avec glissade et une manœuvre de voltige.

Mots-clés : systèmes plats, planification de trajectoire, contrôle de vol, opérateur de Newton, calcul symbolique-numérique, platitude généralisée.

Introduction

We illustrate the use of computer algebra for experimental investigations relying on numerical simulations in the field of automatic control. We consider here the notion of flat systems and some possible generalizations in order to solve motion planning problems for aircrafts models.

The solutions of flat systems [2, 3, 10] can be parametrized by a set of functions, called *flat outputs*, and a finite number of their derivatives. This property is particularly useful for motion planning of non-linear systems, *i.e.* the design of a control law able to generate a trajectory joining a given starting point to a given end point. Though flatness is not a generic property, flat systems are ubiquitous in practice. There is no known complete algorithm to decide flatness (see *e.g.* Lévine [11] for necessary and sufficient conditions), but the flat outputs have often simple expressions that may be guessed by physical considerations.

This work takes place in a systematic study of *apparent singularities* of flat systems, *i.e.* points where the parametrization provided by given flat outputs ceases to be defined [6, 7]. In practice, such situations are more likely to appear when a failure modifies the symmetries of the system or involves the loss of some controls, thus requiring an alternative flat output.

Among the classical examples of flat systems are cars, trucks with trailers, cranes, aircrafts, etc. Note that aircraft models have been studied since long in [14, 15]. Although aerodynamics models are complex and may involve many parameters, they turn out to be flat if one neglects the thrust created by control surfaces (rudder, elevator and ailerons) or associated to angular speeds, a legitimate approximation in many cases.

In practice, we aim at designing a suitable feed-back able to compensate both perturbations and modelling errors. In order to investigate its robustness in the context of maneuvers and failures of increasing difficulties, we have designed a package in Maple. Its implementation is presented and we illustrate its use by a few numerical simulations of trajectory tracking. More details will be given in a forthcoming papers with Y.J. Kaminski.

We focus here on a notion of *generalized flatness*, suggested by computational experiments, trying to improve trajectory tracking when the design of a suitable feed-back becomes hard. We first noticed that, considering trajectories with constant controls and attitude angles, these con-

trols and angles may be computed by solving an algebraic system, *i.e.* a non-differential one. The real model is in this case more complicated, but of the same nature as the simplified one. We sometimes needed to use an alternative simplified model, where control values are not set to 0 but to constant values provided by *ad hoc* calibration functions.

We tried then to go further and to improve the parametrization provided by the simplified model. We have needed to neglect some terms, depending on the controls U. As the flat parametrization provides a first evaluation $U^{[0]}$ for the controls, we can use this value in the perturbation terms of the full model, instead of setting them to 0. We get so a second evaluation $U^{[1]}$ for the controls that may be used to improve the evaluation of the perturbation terms, providing a third evaluation $U^{[2]}\ldots$ This process can be iterated *ad libitum*. In our experiments, this simple change provides, using only 4 iterations, a precise motion planning for the full aerodynamic model, which suggests the introduction of a notion of *generalized flatness* for such systems. "Precise" means here that the trajectories remain close to the values of the flat outputs, without using any feed-back. See simulations in sec. 6. As each iteration implies more derivatives of the flat outputs, such a generalized flat parametrization potentially involves an *infinite* number of derivatives of the flat outputs of the unperturbed flat system.

Flat systems and their singularities are introduced in sec. 1. Detailed aircraft models, for which this motion planning algorithm has been taylored, are presented in sec. 2 and their approximate flatness and singularities are studied in sec. 3. Then, their motion planning, tracking feed-back and the associated Maple package are presented in sec. 4, the implementation of generalized flatness in section 5, followed by examples of flight maneuvers with increasing difficulties in section 6. A last section 7, provides preliminary elements for a theoretical interpretation.

1 Flat systems and their singularities

The first definition of flatness was given in the framework of differential algebra [19]. We prefer here to use a more flexible definition, relying on Vinogradov's notion of diffieties [8, 23], that do not restrict to algebraic systems and algebraic flat outputs. The main difference in our approach, is that diffieties are defined by fixing a derivation, which corresponds to flatness and not just the distribution generated by the associated vector field, which corresponds to orbital flatness when time scaling is allowed. See [3].

1.1 Definition

Définition 1. — *A diffiety V is an open*[1] *subset of $\mathbf{R}^I$, where I is a denumerable set, equipped with a derivation δ. All functions on a diffiety are $\mathcal{C}^\infty$ and only depend on a* finite *number of coordinates. We denote their set by $\mathcal{O}(V)$.*

In the sequel, we will be concerned with diffieties associated to a system of finitely many ordinary differential equations

$$x_i' = f_i(x, u, t), \tag{1}$$

where $x = (x_1, \dots, x_n)$ is the *state vector*, $u = (u_1, \dots, u_m)$ the *controls* and t is the time, implicitly satisfying $t' = 1$. To such a system, we associate $\mathbf{R} \times \mathbf{R}^n \times \left(\mathbf{R}^{\mathbf{N}}\right)^m$, the first copy of $\mathbf{R}$ is for t, then $\mathbf{R}^n$ for x and the last term corresponds to the controls and their derivatives. So the derivation δ, that we denote by d_t is

$$\mathrm{d}_t := \partial_t + \sum_{i=1}^{n} f_i(x, u, t)\partial_{x_i} + \sum_{j=1}^{m} \sum_{k \in \mathbf{N}} u_j^{(k+1)} \partial_{u_j^{(k)}}, \tag{2}$$

denoting $\partial/\partial x_i$ by ∂_{x_i} for simplicity. We may obviously restrict to an open subset, according to physical limitations.

Among such diffieties, is the *trivial diffiety*, which is $\mathbf{R} \times (\mathbf{R}^{\mathbf{N}})^m$, equipped with

$$\mathrm{d}_t := \partial_t + \sum_{j=1}^{m} \sum_{k \in \mathbf{N}} z_j^{(k+1)} \partial_{z_j^{(k)}},$$

1. Using the coarsest topology that makes the i^{th} projection map π_i continuous, for all $i \in I$.

which is in fact the jet space $J^\infty(\mathbf{R}, \mathbf{R}^m)$. We are now able to define flatness.

Définition 2. — *A diffiety morphism $\phi : U_{\delta_1} \mapsto V_{\delta_2}$ is such that $\phi^*\mathcal{O}(V) \subset \mathcal{O}(U)$ and, for any function g on V, $\phi^*\delta_2 g = \delta_1\phi^* g$, meaning that the mapping g is compatible with the derivations.*

The flatness domain, *is the set all* flat points, i.e. *points admitting a neighborhood isomorphic to an open subset of the trivial diffiety.*

Let ϕ be such an isomorphism defined by $z_j := Z_j(x, u, t)$, the functions Z_j are called flat outputs.

Thus, ϕ^{-1} is locally defined and provides a flat parametrization, *defined by $x_i = X_i\left(z, \ldots, z^{(r)}\right)$ and $u_j^{(k)} = U_{j,k}\left(z, \ldots, z^{(r+k+1)}\right)$.*

In many cases, the state space is not affine and can be a sphere, a circle... as we will see soon. In such cases, different charts need to be used to cover it. And flatness can impose to use more charts, each associated to a suitable flat output, in order to cover the whole flatness domain.

1.2 Singularities of flat systems

In the above definition, flat outputs are only defined on open spaces. Points where flat outputs are not defined, or the inverse mapping, are *apparent singularities* for these outputs. *Flat singularities* are the points where no flat parametrization can be defined.

The lack of a general algorithmic criterion to decide flatness makes difficult to characterize flat singularities. In a first stage of a collaboration in progress with Y.J. Kaminski and J. Lévine, we have focused on driftless systems [6] and affine systems [7] with $n-1$ controls, for which the following necessary condition, which amounts to the controllability of the linearized system, turns out to cover all the cases when the action of the control functions remain independent.

The most precise expression of this criterion requires using power series. At a given point η of a diffiety, we associate to any function F the power series: $\mathrm{j}_\eta F := \sum_{k\in\mathbf{N}} \mathrm{d}_t^k F(\eta) t^k/k!$ and consider at each point η the differential operator

$$\mathrm{d}_\eta F := \sum_{k=1}^{n} \mathrm{j}_\eta(\partial_{x_k} f_i)\mathrm{d}x_k + \sum_{j=1}^{m}\sum_{k\in\mathbf{N}} \mathrm{j}_\eta(\partial_{u_j^{(k)}} f_i)\mathrm{d}u_j^{(k)}. \tag{3}$$

THEOREM 3. — *If a diffiety defined by a differential system (1) is flat at point η, then the $\mathbf{R}[\![t]\!][\mathrm{d}_t]$-module defined the* linearized system at η $\mathrm{d}_\eta(x_i' - f_i(x,u,t))$, *that is the quotient $\mathbf{R}[\![t]\!][\mathrm{d}_t]$-module $(\mathrm{d}_\eta x, \mathrm{d}_\eta u)/(\mathrm{d}_\eta(x_i' - f_i(x,u,t)))$, is a free module.*

PROOF. — If Z is a flat output, then $\mathrm{d}_\eta Z$ is a basis of this module. Indeed, $x_i = X_i(Z)$, for $1 \leq i \leq n$ and $u_j = U_j(Z)$, for $1 \leq j \leq n$, so that $\mathrm{d}_\eta x_i = \mathrm{d}_\eta X_i(Z)$ and $\mathrm{d}_\eta u_j = \mathrm{d}_\eta U_j(Z)$. ∎

It seems that we are lacking a good reference for testing freeness of a D-module with coefficient in a power series ring. But things are easy when coefficients are constants.

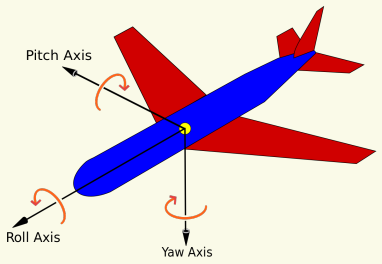
2 Aerodynamic models of aircrafts

We have used the model described by Martin [14, 15] that basically follows most textbooks. We avoid reproducing all lengthy equations to focus on their structure.

It is classical to model aircrafts using the following 12 state variables: $(x, y, z, V, \gamma, \chi, \alpha, \beta, \mu, p, q, r)$. We try to describe briefly their rough meaning. A precise understanding is not mandatory for what follows. First, (x, y, z) correspond to the coordinates of the gravity center of the aircraft, V to its speed, the flight path angle γ and the azimuth angle χ are Euler angles describing the speed vector, μ is the bank angle, corresponding to roll. Those three Euler angles define the *wind frame*, and the sideslip angle β together with the angle of attack α describe respectively the rotations with respect to the z-axis (yaw) and then y-axis (pitch) in order to go from the wind referential to the *aircraft frame*, according to the following figure.

Then, (p, q, r) is the expression of the rotation vector in the Galilean referential tangent to the aircraft referential at each time.

The controls are the following, the thrust of both engines (F_1, F_2), that we prefer to model using their sum $F = F_1 + F_2$ and a parameter $\eta := (F_1 - F_2)/(F_1 + F_2)$, and then the virtual angles δ_ℓ, δ_m and δ_n, that respectively express the positions of the ailerons, elevators and rudder. When the rudder is damaged, it is possible to some extent to use differential thrust η as a control instead of δ_n (see, *e.g.* [13]).



Thanks to Wikipedia

Angle μ corresponds to roll, β to yaw and α to pitch.

Figure 1 – Aircraft rotation axes

2.1 The shape of the equations

We can now describe the shape of the equations, dividing the state variables in 4 subsets: $\Xi_1 := \{x, y, z\}$, $\Xi_2 := \{V, \gamma, \chi\}$, $\Xi_3 := \{\alpha, \beta, \mu\}$ and $\Xi_4 := \{p, q, r\}$. We have:

$$(x', y', z') = G_1(V, \gamma, \chi); \tag{4a}$$
$$(V', \gamma', \chi') = G_2(V, \gamma, \alpha, \beta, \mu, F, [p, q, r, \delta_\ell, \delta_l, \delta_n]); \tag{4b}$$
$$(\alpha', \beta', \mu') = G_3(V, \gamma, \alpha, \beta, \mu, p, q, r); \tag{4c}$$
$$(p', q', r') = G_4(V, \gamma, \alpha, \beta, \mu, p, q, r, \delta_\ell, \delta_l, \delta_n). \tag{4d}$$

The equation (4b) actually depends on $p, q, r, \delta_\ell, \delta_l, \delta_n$, but this dependence is often neglected. With this simplification, setting $\Xi_5 := \{\delta_\ell, \delta_l, \delta_n\}$, at stage i, we can generically express the value of Ξ_{i+1}, using the derivatives Ξ_i'. At stage 2, *i.e.* for $i = 2$, we need to choose one variable ζ in the set $\Xi_3 = \{\alpha, \beta, \mu, F\}$ to form a flat output. Then, generically, x, y, z, ζ and their derivatives allow to compute the values of the state space and controls. The classical choice is $\zeta = \beta$. We now briefly investigate apparent singularities that may appear at each level of derivation, the second one being left for further investigations.

2.1.1 Stage 1

$$\frac{d}{dt}x(t) = V(t)\cos(\chi(t))\cos(\gamma(t)); \tag{5a}$$
$$\frac{d}{dt}y(t) = V(t)\sin(\chi(t))\cos(\gamma(t)); \tag{5b}$$
$$\frac{d}{dt}z(t) = -V(t)\sin(\gamma(t)). \tag{5c}$$

It is easily seen that the values of V, χ and γ, modulo π, can be computed, provided that $V\cos(\gamma) \neq 0$, which seems granted in most situations. The vanishing of V may occur with aircrafts equipped with vectorial thrust, which means a larger set of controls, that we won't consider here. This means that we assume $V > 0$, so that a single value for $(\cos(\chi), \sin(\chi))$ can be determined on the unit circle. The vanishing of $\cos(\gamma)$ can occur with loopings etc. and would require the choice of a second chart with another set of Euler angles. This issue was not investigated here.

2.1.2 Stage 3

We postpone the study of stage 2, that contains the main difficulties, to the next section. The shape of the third level equations imposes $\cos(\beta) \neq 0$. They are linear in (p, q, r), with a non vanishing determinant and so easily solved.

2.1.3 Stage 4

The case of variables (p, q, r) is easy too.
The dynamics of the angular speed matrix (p, q, r) is given by:

$$\begin{pmatrix} \frac{d}{dt}p(t) \\ \frac{d}{dt}q(t) \\ \frac{d}{dt}r(t) \end{pmatrix} = I^{-1} \begin{pmatrix} (I_{yy} - Izz)qr + I_{xz}pq + L \\ (I_{zz} - I_{xx})pr + I_{xz}(r^2 - p^2) + M \\ (I_{xx} - I_{yy})pq - I_{xz}rq + N \end{pmatrix}, \tag{6}$$

where I is the inertia matrix of the aircraft, assumed to be symmetric with respect to the xz-plane, and (L, M, N) the torque, that can obviously be computed using these equations. In general, one expects L to depend

mostly of δ_ℓ, M on δ_m, etc. and to be monotonous in the range of admissible values. Using the GNA model, they are linear in those controls, with invertible matrices.

2.2 The GNA model

The aircraft model equations involve the forces (X, Y, Z) and the torques (L, M, N) acting on the aircraft, that are given by these formulas:

$$X = F(t)\cos(\alpha+\epsilon)\cos(\beta(t)) - \frac{\rho}{2}SV(t)^2C_x - gm\sin(\gamma(t)); \tag{7a}$$

$$\begin{aligned} Y = \; & -F(t)\cos(\alpha+\epsilon)\sin(\beta(t)) + \tfrac{\rho}{2}SV(t)^2C_y \\ & +gm\cos(\gamma(t))\sin(\mu(t)); \end{aligned} \tag{7b}$$

$$Z = -F\sin(\alpha+\epsilon) - \frac{\rho}{2}SV(t)^2C_z - gm\cos(\gamma(t))\cos(\mu(t)); \tag{7c}$$

$$L = -y_p\sin(\epsilon)(F_1(t) - F_2(t)) + \frac{\rho}{2}SV(t)^2aC_l; \tag{7d}$$

$$M = \frac{\rho}{2}SV(t)^2bC_m; \tag{7e}$$

$$N = y_p\cos(\epsilon)(F_1(t) - F_2(t)) + \frac{\rho}{2}SV(t)^2aC_n. \tag{7f}$$

The angle ϵ is a small angle related to the lack of parallelism of the reactors with respect to the xy-plane of the aircraft and ρ is the air density, a and b lengths related to the aircraft characteristics.

The aerodynamic coefficients $C_x, C_y, C_z, C_l, C_m, C_n$ depend on α and β and also on the angular speeds p, q, r as well as the controls δ_l, δ_m and δ_n. To make the system flat, we need to consider that C_x, C_y and C_z only depend on α and β. In the literature, the available expressions are often partial or limited to linear approximations, as in McLean [16]. We used here the Generic Nonlinear Aerodynamic (GNA) subsonic models, given by Grauer and Morelli [4], that cover a wider range of values.

We will provide simulations with 2 aircrafts among the 8 in their database: STOL utility aircraft DHC-6 Twin Otter and the sub-scale model of a transport aircraft GTM (see [5]). Data for the F4 and F16C fighters are also available in our implementation. The GNA model for the aerodynamics functions C appearing in formulas (7a–7f) depends on 45

aerodynamic coefficients, in formulas such as:

$$\begin{array}{rcl} C_D &=& \theta_1 + \theta_2\alpha + \theta_3\alpha\tilde{q} + \theta_4\alpha\delta_m + \theta_5\alpha^2 + \theta_6\alpha^2\tilde{q} + \theta_7\delta_m + \theta_8\alpha^3 \\ & & +\theta_9\alpha^3\tilde{q} + \theta_{10}\alpha^4, \\ C_y &=& \theta_{11}\beta + \theta_{12}\tilde{p} + \theta_{13}\tilde{r} + \theta_{14}\delta_l + \theta_{15}\delta_n, \\ C_L &=& \theta_{16} + \theta_{17}\alpha + \theta_{18}\tilde{q} + \theta_{19}\delta_n + \theta_{20}\alpha\tilde{q} + \theta_{21}\alpha^2 + \theta_{22}\alpha^3 + \theta_{23}\alpha^4, \end{array} \tag{8}$$

where $\tilde{p} = ap$, $\tilde{r} = ar$, $\tilde{q} = bq$, a and b being constants related to the aircraft geometry, C_D and C_L correspond to the lift and drag coefficients. The coefficients C_x and C_z in the wind frame are then given by the formulas:

$$\begin{array}{rcl} C_x &=& \cos(\alpha)C_D + \sin(\alpha)C_L, \\ C_z &=& \cos(\alpha)C_L - \sin(\alpha)C_D. \end{array} \tag{9}$$

Grauer and Morelli also provide all the needed physical constants, but no precise data for landing conditions, flaps... To simulate landing, empirical changes were made. The starting point of this work was to be able to handle the full model, considering changes of flat outputs when singularities are met, and to question the validity of the motion planning provided by a flat simplified model, when trying to control the full model.

3 Flat outputs and their singularities

We now investigate the singularities related to the various choices of flat output, at stage two.

3.1 Classical flat outputs

Martin [14] has used the set of flat outputs: x, y, z, β. We need to explicit under which condition such a flat output is non singular. The differential equations involved at stage two are the following.

$$\frac{d}{dt}V(t) = \frac{X}{m}; \tag{10a}$$

$$\frac{d}{dt}\gamma(t) = -\frac{Y\sin(\mu(t)) + Z\cos(\mu(t))}{mV(t)}; \tag{10b}$$

$$\frac{d}{dt}\chi(t) = \frac{Y\cos(\mu(t)) - Z\sin(\mu(t))}{\cos(\gamma(t))mV(t)}. \tag{10c}$$

The first one (10a) provides the value of X. From its expression, we can express the value of F by (7a), as $\alpha + \epsilon$ is assumed to be small. We see that the two last equations depend on $\cos(\mu)Y - \sin(\mu)Z$ and $\sin(\mu)Y + \cos(\mu)Z$. We get new expressions $\hat{Y}$ and $\hat{Z}$ by substituting in them the value of F provided by (7a). We can compute locally α and μ, provided that

$$\begin{vmatrix} \frac{\partial \hat{X}}{\partial \alpha} & \frac{\partial \hat{X}}{\partial \mu} \\ \frac{\partial \hat{Y}}{\partial \alpha} & \frac{\partial \hat{Y}}{\partial \mu} \end{vmatrix} \neq 0. \tag{11}$$

This condition implies that Y and Z do not both vanish, which excludes 0-g flight for space training or some aerobatics maneuvers, but which stands in most usual flight conditions. The main interest of this choice is to be able to impose $\beta = 0$, which is almost always required.

3.2 The bank angle choice

Considering the flat output $\{x, y, z, \mu\}$, we see that we can compute the values of X, Y and Z. Again, X provides an expression of F, that may be susbsituted in Y and Z to get new expressions $\tilde{Y}$ and $\tilde{Z}$. The flat output is regular when

$$\begin{vmatrix} \frac{\partial \tilde{Z}}{\partial \alpha} & \frac{\partial \tilde{Z}}{\partial \beta} \\ \frac{\partial \tilde{Y}}{\partial \alpha} & \frac{\partial \tilde{Y}}{\partial \beta} \end{vmatrix} \neq 0. \tag{12}$$

The vanishing of this determinant may be interpreted as some kind of stalling condition. Indeed, when $\beta = 0$, it is equal by symmetry to $\partial\tilde{Z}/\partial\alpha\partial\tilde{Y}/\partial\beta$. For most aircrafts, $\partial\tilde{Y}/\partial\beta \neq 0$ seems reasonable, although it may be very small or even negative for some fighter like the F16XL with a delta wing, according to data in [4]. Then, $\partial\tilde{Z}/\partial\alpha$ means that the lift is extremal, which may be taken as a rough mathematical definition of stalling. Of course, we are here working with a simplified model that cannot reflect the irreversible changes in air flow that occurs in real stalling, but only mimics it as a maximum of the lift. In such a situation, the control δ_m that acts on α, and so on the lift, may be considered as lost. And indeed, for a straight line trajectory with constant speed equal to the stalling speed, *i.e.* with α maximal, the aircraft model is not flat according to th. 3. This means that such a flight output always works for most aircrafts, except in situations that obviously need to be avoided for safety reasons.

3.3 The thrust choice

The choice of thrust F has one main interest: to set $F = 0$ and consider the case of an aircraft having lost all its engines. See subsection 6.2. In the case of the GNA model, C_y is linear in β. If $\cos(\mu)\theta_{11} \neq 0$ (see (8)), we may express β depending on α, μ, X_1, X_2 and the aircraft parameters, using equation (10c), and then replace it by this evaluation in X and Z to get new expressions $\bar{X}$ and $\bar{Y}$. The flat outputs including F are non singular iff

$$\begin{vmatrix} \frac{\partial \bar{X}}{\partial \alpha} & \frac{\partial \bar{X}}{\partial \mu} \\ \frac{\partial \bar{Z}}{\partial \alpha} & \frac{\partial \bar{Z}}{\partial \mu} \end{vmatrix} \neq 0. \tag{13}$$

By symmetry, both $\partial\bar{X}/\partial\mu$ and $\partial\bar{Z}/\partial\mu$ vanish when $\beta = \mu = 0$, so that this choice of linearizing outputs requires non zero side-slip angle and bank angle for trajectories included in a vertical plane.

3.4 Other sets of flat outputs

Among the other possible choices for completing the set Ξ_1 in order to get flat outputs, α could work in theory but does not seem to have much specific interest. One may also consider time varying expressions, e.g. linear combinations of β and μ, to smoothly go from one choice to another, which has been implemented but did not lead to a convincing use in simulations.

4 Maple package

We describe here an experimental implementation, only designed at this stage for our own use and lacking of documentation and comments. However, the source code is made available for curious readers: `http://www.lix.polytechnique.fr/{~}ollivier/GFLAT/`. The goal was to get reliable results by minimizing the needed total amount of time, that is the time requested by numerous simulations and the time of implementation.

Four Maple packages were written. The package `GNA` implements data from Grauer and Morelli, the package `Flat_Plane_G2` implements the flat motion planning and its generalization. A package `Newton` contains a

multivariable Newton method and a package `Display_plane` deals with numerical simulations and drawing the curves that illustrate this paper.

Another important point is to be able to control long computations in order to stop them if something goes wrong. The functions were mostly used in verbose mode, displaying the index i of each new time step or intermediate numerical results during motion planning or numerical integration.

This proved important for debugging but also during the repeated trial and error sequences required to guess working parameters for the feed-back.

The general spirit was to limit ourselves to basic Maple functions: manipulation of lists, substitutions, computation with polynomials and classical functions, power series, the solve function for linear systems, and the `dsolve` numerical integrator.

4.1 Physical models. GNA

There is not much to say about this package. Our choice was to use global variables to store all the requested parameters. It has many drawbacks, including some possible protest from Maple numerical integrator, that we were able to overcome. The main advantage is to alleviate the number of arguments in functions that already require a great number of them and to make all the requested intermediate results available for the function used at next step without mistakes and omissions. There is a function for each model of aircraft that store the physical constants, with names such as `TO`, `GTM` or `F16C`. Its arguments are of the form `x = fct(t)`, `y = fct(t)`...a sequence that is stored in a global list to provide the time functions associated to the flat outputs.

We have already said that any combination ζ of β and μ can be used as a flat output. For this, the syntax

```
zeta=(f1(t)*betta+f2(t)*mu=fct(t))
```

is recognized. One may notice that `beta` and `gamma` are already used by Maple. An ugly but fast solution was to write `bbeta` and `gama` to avoid conflicts. In case of rudder failure, one can use relative thrust as control. A generic name for this control is `u_4` and one may write *e.g.* `deltan=u_4`, `eta=0` or `deltan=10*deg`, `eta=u_4`. If a non zero value is given to δ_n, it

will be used at the stage 2 (see 2.1), for better precision, instead of setting it to 0 in order to define the simplified model. Options provide models for ground effect or an expression of air density, depending on altitude. For this, the notation `_z` is used instead of `z` to avoid too early evaluation. One may also assign to `eta` a function of the time, e.g. to model an engine failure. We need then to denote the time by `_t`, again to prevent too early evaluation.

4.2 Newton operator with series

The main task of the Flat-plane model is to achieve motion planning. Following the ideas developed in section 3, this is in principle easy. We encounter two difficulties. First, computing successive derivatives of the flat outputs may lead to formulas of great size and slow computations, mostly when trying to model complicated maneuvers and long flight sequences. Second, we cannot rely at stage 2 (see 2.1) on closed form formulas for solving the equations, so that numerical approximations need to be computed.

Our choice was to compute at a given time a power series expansion of the flat outputs (x, y, z) with all terms up to t^5. At stage 2, a classical Newton method is used to compute constant terms of the series corresponding to α, β, μ or F. Then, we use a Newton method for series (see *e.g.* [1, th. 3.12 p. 70]) to compute their power series expansion modulo t^2 and then t^4, which is enough to get δ_ℓ, δ_m and δ_n as affine functions of t. Higher orders may also be computed and will be needed in sec. 5.

Unless physical considerations makes it difficult or impossible (*e.g.* near stalling conditions), the use of Newton method is in general easy, when initiated with 0 values, as most angles are small. This is no longer the case with flat outputs x, y, z and F, that require higher values of β. Then, some calibration functions (see 5.1) are used to provide suitable values to initiate the computation. Our Newton function is a memory one, so that it starts at step $i+1$ with the values of step i for better efficiency. During experiments, warning messages from the Newton function that fails to provide solution up to 10^{-3} after 20 iterations are the symptoms of a choice of trajectory that is too close to a singularity of the flat output.

4.3 Motion planning

The function `Motion_Planning` takes among its arguments a beginning time, an ending time and the number of time intervals. Its many outputs are not returned as outputs but stored in global variables. The most important is the table `TTG`. At each step time t_i, the power series expansions s_i of the controls and state variables are stored; *e.g.* for α in $\texttt{TTG}[\alpha, i, 0, 0]$. So, they can be used by functions with names such as `falpha`, ... that will compute the value of α at $t_1 \leq t \leq t_{i+1}$ using the formula: $[(t_{i+1} - t)s_i(t - t_i) + (t - t_i)s_{i+1}(t - t_{i+1})]/(t_{i+1} - t_i)$ for better precision.

An option calls Maple numerical solver to build numerical integrators for the full model (stored in `resudsolve`), using just the control functions computed with the simplified model, or completing them with feed-back functions (stored in `resudsolveB`), that are described in the next subsection. Then, the function `bouclage` that computes the feedback is also called.

An extensive use of the `subs` Maple function allows to perform rewriting tasks, replacing in the equations parameters by their values, as well as already computed state variables. A basic function `serpol` (and avatars that apply to both terms of an equality, list of equalities etc.) computes a power series expansion and convert it to a polynomial, that is easier to handle for further computations.

4.4 Design of the feed-back

The function `bouclage` that computes the feedback takes a single argument that is the fourth linearizing output: β, μ or F. It return no values, the computed results being stored in global variables.

To design the feed-back, we consider the linearized system around the trajectory planned using $\mathrm{d}x$, $\mathrm{d}y$, $\mathrm{d}z$ as flat outputs of this linear system, completed with $\mathrm{d}\beta$ or $\mathrm{d}\mu$, according to the case, or nothing with the F output. The state functions are replaced at each step i by its power series expansion at t_i. The main idea is to achieve an exponential decrease of δx, δy, ... that is the difference between values x, y, ... computed by numerical integration using the full model and the planned values $\tilde{x}$, $\tilde{y}$, ... using the flat parametrization. To be able to correct model errors, we

also need to use the integrals

$$\begin{array}{rl} I_1 &= \int_{t_0}^{t} \cos(\chi(\tau))\delta x(\tau) + \sin(\chi(\tau))\delta y(\tau)\mathrm{d}\tau; \\ I_2 &= \int_{t_0}^{t} -\sin(\chi(\tau))\delta x(\tau) + \cos(\chi(\tau))\delta y(\tau)\mathrm{d}\tau; \\ I_3 &= \int_{t_0}^{t} \delta z(\tau)\mathrm{d}\tau; \\ I_4 &= \int_{t_0}^{t} \delta\zeta(\tau)\mathrm{d}\tau, \end{array} \tag{14}$$

where t_0 is the initial time of the simulation and ζ is β or μ according to our choice of flat outputs.

The algebraic design of the feed-back relies on computations in the differential module defined by the linearized system at each step time t_i, using the analogy between the assumed "small variations" $\delta\xi = \xi - \tilde{\xi}$ and $\mathrm{d}\xi$ for any state variable ξ. Each equation P of the system is replaced by its differential $\sum_{\xi} \partial P/\partial\xi \mathrm{d}\xi$ and one substitutes to the ξ's their power series estimation $\tilde{\xi}$.

Lists of positive real values $\lambda_{i,j}$ having been given, the feed-back $\delta F = c_{1,I_1} + \sum_{\xi\in\Xi_1\cup\Xi_2\cup\Xi_3} c_{1,\xi}\delta_\xi$ is set so that $\prod_{k=1}^{3}(\mathrm{d}/\mathrm{d}t - \lambda_{1,k})I_1$ is equal to 0. In the same way, the feed-backs $\delta\delta_\ell = \sum_{\xi\in\hat{\Xi}} c_{2,\xi}\delta_\xi$, $\delta\delta_m = \sum_{\xi\in\hat{\Xi}} c_{3,\xi}\delta_\xi$ and $\delta u_4 = \sum_{\xi\in\hat{\Xi}} c_{4,\xi}\delta_\xi$, where $\hat{\Xi} = \{I_1, \ldots, I_4\} \cup \bigcup_{p=1}^{4} \Xi_p$, are computed, so that $\prod_{k=1}^{5}(\mathrm{d}/\mathrm{d}t - \lambda_{2,k})I_2$, $\prod_{k=1}^{5}(\mathrm{d}/\mathrm{d}t - \lambda_{3,k})I_3$ and $\prod_{k=1}^{3}(\mathrm{d}/\mathrm{d}t - \lambda_{4,k})I_4$ are all equal to 0.

We proceed just as for the motion planning. At each step time t_i, an the expressions for δF, $\delta\delta_\ell$, ... are computed and stored in the global array `TtF`$[i]$, `Ttdeltal`$[i]$, ... so that these results can be used by numerical functions `ftF`, `ftdeltal`, ... that achieve fast numerical computation of the feed-back during the integration.

Under good hypotheses, the I_p, $1 \leq p \leq 4$ tend to a constant value, or a slowly varying value, so that their derivatives are 0, or small, just as the δx, δy, δz and $\delta\zeta$. Troubles appear with fast maneuvers and also with aircrafts like the Twin Otter with generous controls surfaces, generating greater thrusts. Too big values for the $\lambda_{i,j}$ can create instabilities, two small values do not manage to keep close to the planned trajectory. Choices where made with trial and errors, that sometimes required many interrupted simulations.

The choice of F as a flat output just requires minor changes. We only need to use I_1, I_2 and I_3 and compute the feed-backs $\delta\delta_\ell$ ... so that $\prod_{k=1}^{5}(\mathrm{d}/\mathrm{d}t - \lambda_{p,k})I_p$, for $1 \leq p \leq 3$.

5 Generalized flatness

5.1 Calibration functions

When the torsion and the curvature of the trajectory are constants, the values of the controls F, δ_l, δ_m and δ_m are constant too. It is then possible to compute them, just knowing V, γ, χ' and β, even for the full model. They are solutions of a non-linear system, that may be solved using Newton method. Indeed, looking at the set of equations (4c), (4d) and the equations (10a) and (10b), we see that for such trajectories, the derivatives in the left members are equal to 0. On may add equation (10c), for which the left member χ' is a constant. We have then 9 equations between the 13 unknowns in $\{V, \gamma, \chi', F\} \cup \Xi_3 \cup \Xi_4 \cup \Xi_5$. Generically, we need to fix 4 values to have local expressions of the 13 others. We have implemented such functions to compute the angle of attack α, depending of V, or to compute stalling speed. They most of the time only depend of 2 arguments, instead of 4, when assuming $\gamma = \chi' = 0$, or just one, when assuming also $\beta = 0$.

5.2 From calibration to time varying controls

When the control functions are not constant, it remains possible to evaluate their values with the full system. The basic idea is to recompute the trajectory planned with the simplified system, using the values obtained for $p, q, r, \delta_\ell, \delta_m, \delta_n$, instead of 0. The process can then be iterated, and we can describe it in the general setting of an almost chained system, such as

$$\begin{array}{rl} (Z'_h, X'_h) = & G_h(Z_1, \dots Z_{h+1}, X_1, \dots, X_{h+1}) \\ & +H_h(X_{h+2}, \dots, X_{h+\ell_h}), 1 \leq h \leq r, \end{array} \tag{15}$$

with the $\ell_h \geq 1, 1 \leq h \leq r$. By convention, $\ell_h = 1$ means that $H_h = 0$. The X_h form a partition of X, the Z_h a partition of Z and $X \cup Z$ is the set of both state variables and controls, the distinction being more physical than mathematical. We assume that $\sharp X_h + \sharp Z_h = \sharp X_{h+1}$, $\sharp Z = m$, the number of controls and $\sharp X_1 = 0$, where $\sharp X_p$ denotes the cardinal of X_p.

If one neglects H, or replace in H its arguments by any known value $\hat{X}$, the variables in Z are assumed to be flat outputs for the system. This assumption means that setting $Z_{h,i} = \zeta_{h,i}(t)$, one can at time t_0 replace $Z_{h,i}$ in the equations (15) by a power series development of $\zeta_{h,i}$ at order $\kappa -$

$h+1$ and compute power series solutions $\tilde{X}_h$ at order $\kappa - h + 1$. This is assumed to be implemented in a function $\texttt{FlatParametrization}(t_0, \kappa, \zeta, \hat{X})$. Using any guessed value $\hat{X}^{[-1]}$, with $X_h^{[-1]}$ known at order $\kappa - h + \ell_h$, we can compute an approximation of the state and control

$$\hat{X}^{[0]} := \texttt{FlatParametrization}(t_0, \kappa_0, \zeta, \hat{X}^{[-1]}),$$

where each set $\hat{X}_h$ is computed at order $\kappa_0 - h + 1$.

This may be iterated J times, using $X^{[0]}, X^{[1]}, \ldots$ instead of the guessed value $X^{[-1]}$, as described by the following process, where the input v denotes the guessed initial value, ζ any vector of m functions, J a non-negative integer and e the wanted order for the output. The order of the output decreases of $L := \max_{h=1}^{r} \ell_h - 1$ at each iteration.

$\texttt{GeneralizedFlatParametrization}(v,\ \zeta,\ J,\ e)$
$\hat{X}^{[-1]} := v$ *(Guessed values)*;
$L := \max_{h=1}^{r} \ell_h - 1$;
$\kappa_0 := e + r + JL$;
for j **from** 0 **to** J **do**
    $\hat{X}^{[j]} := \texttt{FlatParametrization}(t_0, \kappa_j, \zeta, \hat{X}^{[j-1]})$,
    $\kappa_{i+1} := \kappa_i - L$;
od;
$\texttt{return}\ \hat{X}^{[J]}$;

Returning to the plane model, we have $\sharp X_1 = 0$, $\sharp X_2 = \sharp X_3 = 3$ and $\sharp X_4 = \sharp X_5 = 4$, adding $F^{(p-3)}$ to Ξ_p, for $p = 4, 5$, for consistency with (15). Furthermore, we have $Z_1 = \{x, y, z\}$ and $Z_3 = \{\xi\} \in \{\alpha, \beta, \mu, F\}$, with $X_3 = \{\alpha, \beta, \mu, F\} \setminus \{\xi\}$.

The only term H is H_2, that depends of the state variables p, q, r in X_4 and the controls δ_l, δ_m and δ_n in X_5. So, $L = 2$ in our case. This means that with J iterations, we need to start computations with series of order $5 + 2J$ in oder to get the controls δ in X_5 at order 1.

All the unavoidable accessory tinkerings in the real implementation would be tedious to detail, but basically, implementing generalized flat parametrization is an easy task, as we just have to increase the orders of a known integer and to implement a loop that iterates the core of the `Motion_Planning` function. At iteration j, the series corresponding, *e.g.*, to α is stored in $\texttt{TTG}[\alpha, i, j, 0]$.

We do not investigate more deeply here the question of the convergence of this process, beyond the fact that the H_h are assumed to be

"small" and that a limited number of iterations provide good results in the following examples, all computed with $J = 4$.

6 Examples

Designing a trajectory that matches actual practice and aircrafts possibilities by looking at flight instructions books and pilots forums sure helps. We did not try to use tricks to reduce computation time in order to get better precision.

6.1 Single engine

We model here a Twin Otter that loses an engine, whose power gradually decreases. We go from equal thrust to total extinction of starboard engine, setting the value of $\eta = (F_1 - F_2)/(F_1 + F_2)$, as in equation (16) below. The distance of the engines to the plane of symmetry of the aircraft has been evaluated to 9.2ft.

The rudder must compensate the torque created by a dissymmetric thrust. With the full model, the rudder also creates a thrust, that must be compensated by a variation of β or μ. With $\beta = 0$ or $\mu = 0$, the trajectory planned by the simplified model is the same. Using here the feed-back for β, μ will change.

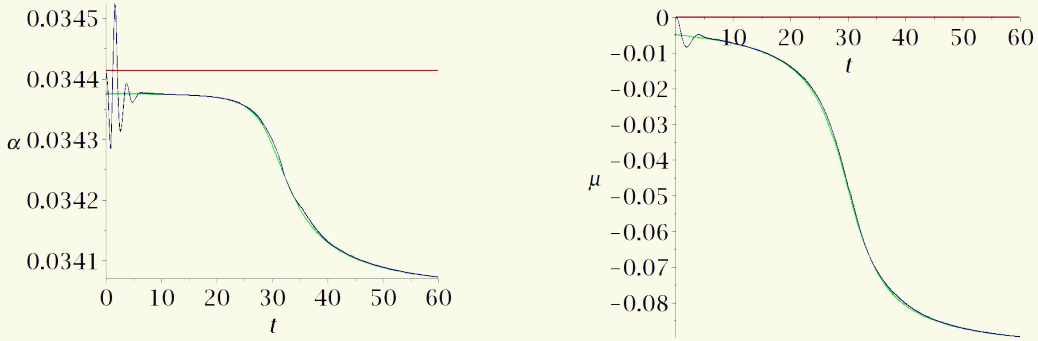
$$\begin{aligned} x &= 140\text{kts}t; \quad y = 0; \quad z = 0; \quad \mu = 0; \\ \eta &= .5 + \frac{\arctan\frac{t-30.}{5.}}{\pi} \end{aligned} \tag{16}$$

The Twin Otter has generous control surfaces, making it highly manoeuvrable, but meaning a higher contribution of the δ_l, δ_m, δ_n to C_x, C_y and C_z. We borrow with some adaptations the values of the $\lambda_{i,j}$ suggested by Martin [14]: $\lambda_{1,1} = 1.$, $\lambda_{1,2} = 2.$, $\lambda_{1,3} = 3.$, $\lambda_{2,1} = 1.$, $\lambda_{2,2} = 1.$, $\lambda_{2,3} = 1.$, $\lambda_{2,4} = 2.$, $\lambda_{2,5} = 3.$, $\lambda_{3,1} = 1.5$, $\lambda_{3,2} = 1.5$, $\lambda_{3,3} = 1.5$, $\lambda_{3,4} = 3.$, $\lambda_{4,5} = 4.$, $\lambda_{4,1} = 1.$, $\lambda_{4,2} = 2.$, $\lambda_{4,3} = 3$.

The variations of μ remains little, in accordance with the reported ability of the T-O to fly with a single engine (Lecarme [9]).

We see that the integrated curves converge to the curves planned by generalized flatness, after initial oscillations, which already shows that this prediction is meaningful. The total computation time for the flat and

Figure 2 – Twin Otter loosing one engine, with $\beta = 0$.



The flatness planned curve is in red, the integration with feed-back in darkblue and the generalized flatness curve in green.

generalized parametrization is 1279sec. The numerical simulation takes 76sec.

6.2 Forward slip

This maneuver may be used for emergency landing, when an aircraft that has lost all engines comes near the landing strip too high or too fast. A way to decrease speed and altitude is to increase β and μ in opposite ways, creating deceleration when aerobrakes are unusable. It is in general used for small aircrafts, but there is a successful example of an emergency landing with an airliner, at the former air force basis of Gimli, Manitoba, in 1983 [12]. Here we used a calibration function to guess initial values and non zero values for the controls, close to the mean speed and flight path angle of our trajectory.

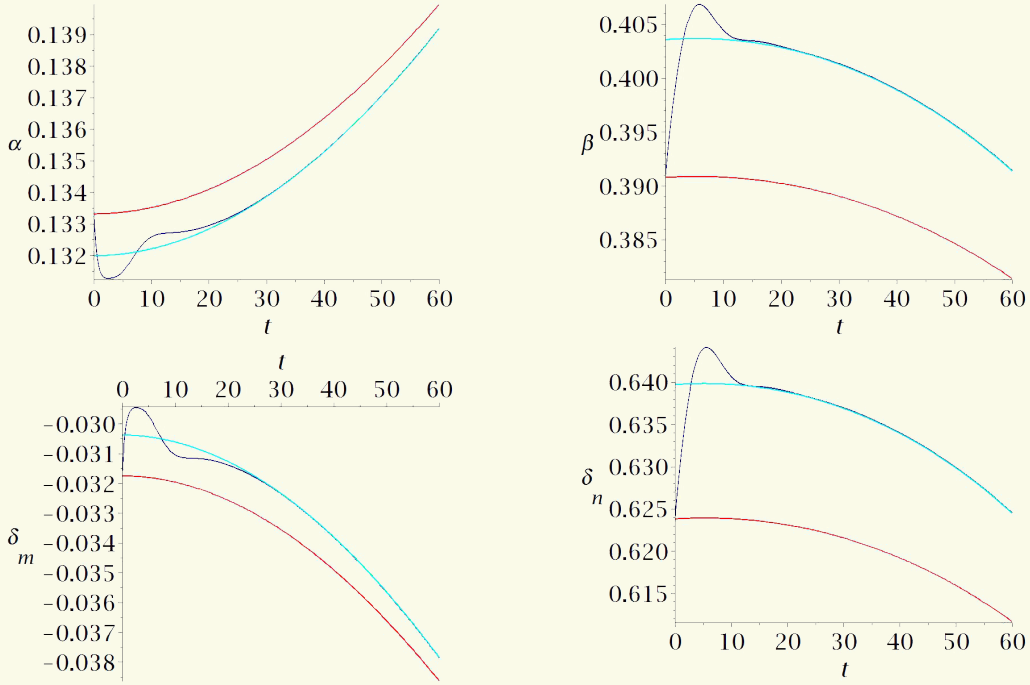
The following table shows constant values for straight line trajectories, depending on α and β, for both the real and the simplified models with $(p, q, r, \delta_l, \delta_m, \delta_n) = (0, 0, 0, 0, 0, 0)$.

Model	α	β	γ	μ	V	δ_l	δ_m	δ_n
Simple	0.15	0.	-0.1187	0.	29.8996	0.	0.	0.
Real	0.15	0.	-0.1190	0.	30.3053	0.	-0.0490	0.
Simple	0.15	0.2	-0.1650	0.2409	29.3672	0.	0.	0.
Real	0.15	0.2	-0.1470	0.1345	30.1114	-0.1880	-0.0490	0.3305
Simple	0.15	0.35	-0.2508	0.3899	28.4019	0.	0.	0.
Real	0.15	0.35	-0.2027	.2250	29.7171	-0.3316	-0.0490	0.5690

For our simulation, we have chosen $\alpha = 0.15$ and $\beta = 0.35$ as reference values to set the controls. To fix ideas, the speed values for such a 0.055 scale model must be divided by $0.055^{0.5}$ to get full scale values, which means 456.1709km/h for the total speed. Here are the flat output trajectories and feed-back parameters.

$$\begin{array}{ll} x & = 29.10852587t + 50\sin(t/60.); \\ y & = 60\cos(t/100. + 2.); \\ z & = -1000 + 5.983293200t + 70\sin(t/70.)); \\ \lambda_{i,j} & = 0.5 \end{array} \tag{17}$$

Figure 3 – Forward slip with the GTM



The flatness planned curve is in red, the integration with feed-back in dark blue and the generalized flatness curves in green. The curve in cyan is the integration with the generalized flatness planned controls and without feed-back.

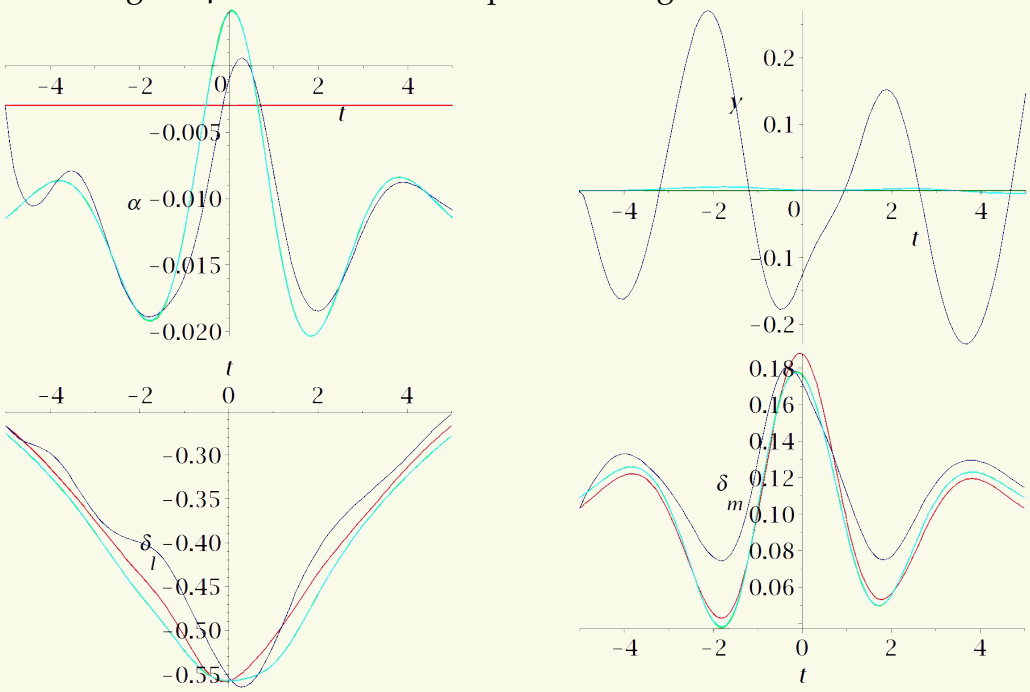
Again, the feed-back allows the integrated value to converge to the curve planned by generalized flatness with good precision, after initial oscillations. The curves δ_l and δ_m actually show $\delta_m + \delta\delta_m$ and $\delta_n + \delta\delta_n$,

including feed-back. We have included here the integration of the general system, with we initial values and control coming from generalized flatness. The coincidence is so good that the generalized flatness planned curves in green are covered by the curve in cyan provided by the integration.

6.3 Aileron roll and parabolic flight

Here, we investigate a limit case with rapid changes. The trajectory is parabolic with acceleration g, so the flat outputs with β is unusable. We use μ, setting $\mu = \pi/2t$. A fighter would have been more credible, but we could only make the feed-back work with the GTM. The horizontal speed is 100km/h.

Figure 4 – Aileron roll and parabolic flight with the GTM.



The flatness planned curve is in red, the integration with feed-back in dark blue and the generalized flatness curve in green. The curve in cyan is the integration with the generalized flatness planned controls and without feed-back.

We see that the feed-back permits to follow the generalized flatness planned curve, but things are moving too fast to keep always the two curves close. The integration in cyan with the generalized flatness planned control, without feed-back, remains very close to the prediction, which confirms that the generalized flatness parametrization is a good approximation of a solution of the real system. *E.g.*, a small discrepancy of about 0.5cm, is observed for y at $t = 5.$, one of the only state function for which the curve in green appears bellow the cyan one. The computation time is 647sec for the motion planning and 402sec for the simulation.

To better appreaciate the convergence of the generalized flatness loop, we have computed the values for the controls F, δ_l, δ_m and δ_n at $t = -1.9$ a time for which the differences with the plain flatness values are much appreciable. They are given in the table bellow.

	$J=0$	$J=1$	$J=2$	$J=3$	$J=4$	$J=5$	$J=6$	$J=7$
F	-2.36	8.40	8.56	8.610	8.624	8.628	8.6304	8.6309
δ_l	-0.44	-0.45	-0.462	-0.4642	-0.4647	-0.4648	-0.46493	-0.464918
δ_m	0.04	0.04	0.039	0.0389	0.0387	0.03872	0.038730	0.038731
δ_n	0.05	0.07	0.085	0.0871	0.087	0.08800	0.087997	0.0880978

The theoretical study of convergence is of course of a great interest, but it is known that such a property is not mandatory for applications. *E.g.*, some divergent series, using smallest term trunctation, can provide accurate and fast computations. See [18].

7 Generalized flatness from the theoretical standpoint

The flat parametrization only involves a finite number of derivatives, which is the basis of all known necessary conditions of flatness (see [22, 21, 17]). We have seen that our motion planning is a limit that potentially involves an infinite number of derivatives, as the evaluation for the controls δ at step $j+1$ depends on the second derivative of their evaluation at step j. This gives some credibility to a folkloric conjecture, claiming that *all controllable systems are flat if functions of an infinite number of derivatives are allowed*. We propose some elements of interpretation in the linear case.

We may indeed consider the simple system $x' = y + \epsilon y'$. When ϵ is 0, x is a flat output. For $\epsilon > 0$, we may choose $\zeta_\epsilon := x - \epsilon y$. However,

we can keep x as a *generalized flat output*. Indeed, one may write $y = \sum_{i\in\mathbf{N}}(-1)^i\epsilon^i(\mathrm{d}/\mathrm{d}t)^i x$. This series will converge if x is analytic with a convergence radius greater that $1/\epsilon$. Moreover, if there exists a linear operator L in $\mathbf{R}[\mathrm{d}/\mathrm{d}t]$ such that $Lx = 0$ and $1 + \epsilon\mathrm{d}/\mathrm{d}t$, as well as $\mathrm{d}/\mathrm{d}t$, are not a factors of L, then there exists M and N such that $ML + N(1 + \epsilon\mathrm{d}/\mathrm{d}t) = 1$, so that $y = Nx'$. Taking for L the sequence $(\mathrm{d}/\mathrm{d}t)^i$, the sum that gives the value of y becomes trivially finite. This situation is close to our considerations about calibration in subsec. 5.1. But this can work also with any operator $\prod_{i=1}^{k}(\mathrm{d}/\mathrm{d}t - \lambda_i)^i$, such as those that we met for designing feed-backs in subsec. 4.4.

Conclusion

We have seen how computer algebra may help to investigate the validity of some simplifications required to reduce to a flat model. Although we could rely on very classical algorithmic tools, some investment have been required to work out for our experiments an implementation with acceptable computation times. One also need a joint use of symbolic and numeric computations.

A slight modification of the code used with the simplified flat model have made possible the direct computation of an accurate motion planning for the original non flat system, an observation that cannot be a mere artefact and so requires a theoretical explanation.

One cannot predict if this notion of generalized flatness will have actual applications. The theoretical difficulties are also unkown, but the unanswered problems related to flatness show that limited theoretical knowledge is not an obstacle to applicability, as long as computations are fast and results reliable. The complexity of the model used here could justify some optimism for computational success with much simpler examples, such as the car with two deported trailers, known not to be flat [20].

Those investigations include an algorithmic aspect. *E.g.*, one may ask whether is it possible to compute the generalized parametrization in a faster way, using some kind of Newton method, which could also help to investigate the convergence of the process. So, even if the applicability should be limited, computational issues may remain of some interest.

Thanks To Yirmeyahu J. Kaminski, Jean Lévine and anonymous referees for their patience, rereading and suggestions.

References

[1] Alin Bostan, Frédéric Chyzak, Marc Giusti, Romain Lebreton, Grégoire Lecerf, Bruno Salvy, and Éric Schost, *Algorithmes efficaces en calcul formel*, Frédéric Chyzak (auto-édit.), Palaiseau, September 2017 (french), 686 pages. Printed by CreateSpace. Also available in electronic version.

[2] M. Fliess, J. Lévine, Ph. Martin, and P. Rouchon, *Flatness and defect of non-linear systems: introduction theory and examples*, Int. Journal of Control **61** (1995), no. 6, 1327–1361.

[3] ———, *A Lie-Bäcklund approach to equivalence and flatness of nonlinear systems*, IEEE Trans. Automatic Control **44** (1999), no. 5, 922–937.

[4] Jared A. Grauer and Eugene A. Morelli, *A generic nonlinear aerodynamic model for aircraft*, AIAA Atmospheric Flight Mechanics Conference, AIAA, 2014.

[5] Richard M. Hueschen, *Development of the transport class model (tcm) aircraft simulation from a sub-scale generic transport model (gtm) simulation*, Tech. Report NASA/TM–2011-217169, NASA, 2011.

[6] Y. Kaminski, J. Lévine, and F. Ollivier, *Intrinsic and apparent singularities in differentially flat systems, and application to global motion planning*, Systems & Control Letters **113** (2018), 117–124.

[7] ———, *On singularities of flat affine systems with n states and $n-1$ controls*, International Journal of Robust and Nonlinear Control **30** (2020), no. 9, 3547–3565.

[8] V.V. Krasil'shchik, V.V. Lychagin, and A.M. Vinogradov, *Geometry of jet spaces and nonlinear partial differential equations*, Gordon and Breach, New York, 1986.

[9] J. Lecarme, *Lignes de vol, le de havilland dhc-6 twin otter*, Aviation Magazine (1966), no. 449.

[10] J. Lévine, *Analysis and control of nonlinear systems: A flatness-based approach*, Mathematical Engineering, Springer, Dordrecht, Heidelberg, London, New-York, 2009.

[11] ———, *On necessary and sufficient conditions for differential flatness*, Applicable Algebra in Engineering, Communication and Computing **22** (2011), no. 1, 47–90.

[12] George H. Lockwood, *Final report of the board of inquiry into air canada boeing 767 c-gaun accident — gimli, manitoba, july 23, 1983*, Tech. report, Minister of Supply and Services Canada, 1985.

[13] Long K. Lu and Kamran Turkoglu, *Adaptive differential thrust methodology for lateral/directional stability of an aircraft with a completely damaged vertical stabilizer*, International Journal of Aerospace Engineering **218** (2018).

[14] P. Martin, *Contribution à l'étude des systèmes différentiellement plats*, Ph.D. thesis, Ecole Nationale Supérieure des Mines de Paris, Paris, France, 1992.

[15] Philippe Martin, *Aircraft control using flatness*, CESA'96 - Symposium on Control, Optimization and Supervision (Lille, France), IMACS/IEEE-SMC Multiconference, 1996, pp. 194–1999.

[16] Donald McLean, *Automated flight control systems*, Prentice Hall, New York, 1990.

[17] François Ollivier, *Une réponse négative au problème de lüroth différentiel en dimension 2*, C. R. Acad. Sci. Paris **327** (1998), no. 10, 881–886.

[18] J.P. Ramis, *Séries divergentes et théories asymptotiques*, Société Mathématique de France, Marseille, 1993.

[19] J.F. Ritt, *Differential algebra*, American Mathematical Society, Providence, Rhodes Island, 1950.

[20] P. Rouchon, M. Fliess, J. Levine, and P. Martin, *Flatness, motion planning and trailer systems*, Proceedings of 32nd IEEE Conference on Decision and Control, IEEE, 1993, pp. 2700–2705 vol.3.

[21] Pierre Rouchon, *Necessary condition and genericity of dynamic feedback linearization*, Journal of Mathematical Systems Estimation and Control **4** (1994), no. 2, 1–14.

[22] Willem M. Sluis, *A necessary condition for dynamic feedback linearization*, Systems & Control Letters **21** (1993), 277–283.

[23] Victor V. Zharinov, *Geometrical aspects of partial differential equations*, Series on Soviet and East European Mathematics, World Scientific, Singapore, 1992.