Numerical Search for Local (Partial) Differential Flatness

Carmelo Sferrazza, Diego Pardo and Jonas Buchli

Abstract— **Differential flatness is a property of certain systems that greatly simplifies the generation of optimal and dynamically feasible trajectories. Using a differentially flat model, there is no need to integrate the system dynamics to retrieve the states and the constraints of the optimization problem are simpler. Recently, the concept of partial differential flatness has been introduced covering a broader class of systems. In particular, it allows to reduce the need for integration by limiting it to a subset of the states. However, finding an analytical expression for the (partial) differential flatness requires the manipulation of the equations of motion in a very specific manner such that a series of properties are fulfilled. In general, finding such analytical model is not straightforward nor compatible with algorithmic models. In order to tackle this problem, in this paper we present a numerical method to find a (partially) differentially flat model of a system around a collection of states and inputs trajectories. We present results on three underactuated nonlinear systems (cart-pole, planar ballbot and a 3D quadrotor). As use case examples, we show online trajectory re-planning tasks. The validity of the trajectories obtained with the locally flat models is verified by forward integrating the original equations of motion together with an optimal stabilizer.**

I. Introduction

Planning dynamically feasible and optimal trajectories in nonlinear systems is challenging. Recently, numerical trajectory optimization has received considerable attention in robotics [15], [16], [17]. However, solving a nonlinear optimization problem requires a significant amount of iterations (and therefore time). Most of the computational complexity comes from the constraints corresponding to the differential equations describing the dynamics of the system [18]. Conversely, for a small class of systems, known as differentially flat, the trajectory optimization problem is reduced to a linearly constrained program. Specifically, the differential flatness property makes possible to avoid the integration of the dynamics in order to retrieve the states and inputs trajectories. This is very useful in the case of online re-planning, where convergence time is critical.

Roughly speaking, a system is differentially flat if it is possible to find a flat output of the same dimension of the input vector, such that all the states and inputs can be expressed as a function of this flat output and its derivatives. Clearly this is a very restrictive property, and therefore most underactuated systems are not differentially flat. In order to tackle this problem, a broader class of systems, known as partially differentially flat, have been recently introduced [1].

Partial differential flatness requires only a partition of the states, as well as the inputs, to be a function of the flat output and its derivatives. The only further requirement is that the remaining states must be given by chains of integrators, where the derivatives of the highest order elements of each chain is a function of the flat output and its derivatives. Formal definitions of both types of differential flatness are shown in Section II.

The concepts of differential flatness and feedback linearization [1] are highly interconnected. However, differential flatness is a structural property and it does not imply the linearization of the system. Such structure can be exploited for designing control and trajectory planning algorithms [19]. There are many examples in the literature where differential flatness has been proven to be very effective for both optimal trajectories generation and tracking control [3], [9]. Moreover, criteria for proving differential flatness have been shown for different systems [6]. In [7] a catalog of some differentially flat mechanical systems has been provided. In [5] a direct characterization of the system using matrices expressing the states and inputs as a function of a linear flat output and its time derivatives has been proposed. In [4] it has been shown how the flat outputs can be retrieved in uniformly controllable linear time-varying systems, as a linear combination of the states. In [1] sufficient conditions have been shown in order to prove partial differential flatness. Recently the concept of differential flatness and partial differential flatness have been proven and exploited in different mechanical systems on principal bundles [8]. These examples apply to a small class of systems and usually provide only sufficient conditions for proving or excluding (partial) differential flatness. However, there is still the problem of finding analytical expressions for the flat output and the functions relating it to the states and inputs in more general cases.

In this paper we present a method based on a numerical procedure to discover local (partial) differential flatness. Given a set of inputs and states trajectories, this method finds a mapping between the flat output and the system states and inputs. The method is formulated as an optimization problem that returns a local flat model that can then be used for online trajectory re-planning. This is very useful for systems where finding the analytical flat model is not straightforward and for numerical models (i.e. derived from algorithms [2]) where discovering analytical (partial) differential flatness is not evident.

Here we apply the method to three systems (cart-pole,

This research has been funded through a Swiss National Science Foundation Professorship award to Jonas Buchli and by the Swiss National Centre of Competence in Research Robotics (NCCR Robotics).

Carmelo Sferrazza carmelos@student.ethz.ch, Diego Pardo depardo@ethz.ch and Jonas Buchli buchlij@ethz.ch are with the Agile Dexterous Robotics Lab at the Institute of Robotics and Intelligent Systems, ETH Zürich, Switzerland.

planar ballbot and 3D quadrotor). As use cases we present problems where the flat model is used for online optimal trajectory re-planning. Moreover, in order to verify the feasibility of the results, we show the forward integration of the original equations of motion using the planned trajectories together with an optimal feedback stabilizer. Results show that the re-planning process may be executed online using the numerical flat model.

The rest of the paper is structured as follows. Section II presents the formal definitions of (partial) differential flatness, introducing the mathematical notation for the rest of the discussion. Section III introduces the numerical search for (partial) differential flatness, formulated as an optimization problem. Section IV presents three examples where the method has been applied, and where the results of the numerical search are used in an online re-planning scenario. Finally, Section V lists the benefits and drawbacks of the approach and discusses future extensions, while Section VI presents the conclusions.

II. Definitions

In this section we show the formal definitions for Differential Flatness and Partial Differential Flatness. For a complete derivation see [1], [3].

A. Differential Flatness

The dynamics of a given system is represented by a set of differential equations,

$$\dot{x}(t) = f(x(t), u(t)) \tag{1}$$

where $x \in \mathbb{R}^n$ represents the states of the system and $u \in \mathbb{R}^m$ the vector of control inputs. The system is differentially flat if there exist "flat" outputs $y \in \mathbb{R}^m$, such that all the states and inputs can be expressed as a function of the flat outputs and a finite number of their derivatives. Formally, the system is differentially flat if there exist functions ξ, ψ, χ, and finite integers p, q such that:

$$\begin{aligned} y &= \xi(x, u, \dot{u}, \cdots, u^{(p)}) \\ x &= \psi(y, \dot{y}, \cdots, y^{(q)}) \\ u &= \chi(y, \dot{y}, \cdots, y^{(q)}) \end{aligned} \tag{2}$$

where the apices $^{(j)}$ indicate the j-th derivatives. This model is equivalent to (1) and can be used to efficiently generate optimal trajectories.

B. Partial Differential Flatness

A system is partially differentially flat if a partition of the states exists,

$$x = \begin{bmatrix} x_r \\ x_{ur} \end{bmatrix} \quad x_r \in \mathbb{R}^{n_r}, \quad x_{ur} \in \mathbb{R}^{n_{ur}}, \quad n_{ur} = n - n_r, \quad n_r \leq n$$

such that the retrievable states, x_r, and all the inputs can be expressed as a function of the flat outputs, $y \in \mathbb{R}^m$, and a finite number of their derivatives. Additionally, the dynamics of the unretrievable states, x_{ur}, must be given by one or more chains of integrators. Finally, the derivative of the highest order elements of each of these chains must be expressed as a function of the flat outputs and their derivatives.

Formally, the system is partially differentially flat if there exist functions ξ, ψ, χ, finite integers p, q, l, matrices A_i and b_i for $i = 1, \cdots, l$, such that,

$$\begin{aligned} y &= \xi(x_r, u, \dot{u}, \cdots, u^{(p)}) \\ x_r &= \psi(y, \dot{y}, \cdots, y^{(q)}) \\ u &= \chi(y, \dot{y}, \cdots, y^{(q)}) \end{aligned} \tag{3}$$

and,

$$\dot{x}_{ur} = \begin{bmatrix} A_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & A_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & A_l \end{bmatrix} x_{ur} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_l \end{bmatrix} \tag{4}$$

where l is the number of chains of integrators, and A_i and b_i are given by,

$$A_i = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ 0 & \mathbf{0} \end{bmatrix}, \quad b_i = \begin{bmatrix} \mathbf{0} \\ h_i(y, \dot{y}, \cdots, y^{(q)}) \end{bmatrix}$$

where h_i is a smooth scalar function. The dimensions of the identity ($\mathbf{I}$) and zero ($\mathbf{0}$) matrices/vector depend on the length of each chain of integrators.

For $n_r = n$, the definition of partial differential flatness is equivalent to the one of differential flatness. Thus, the former encompasses a larger class of systems.

III. Numerical Search

Finding an analytical expression for the flat outputs is not always evident. Even for relatively simple system dynamics, the functions relating the flat outputs with the states and the inputs are often hard to derive.

In this section, a numerical approach to prove local (partial) differential flatness is described. The main assumption of this method is that a collection of feasible states and inputs trajectories is available. Additionally, we assume that the (partially) differentially flat model can be approximated using a parametric model based on a set of basis functions.

A. Local Differential Flatness

Let us approximate ξ in (2) as a linear combination of certain basis functions of the states, inputs and a finite number of derivatives of the latter, i.e.,

$$y \cong \sum_{i=1}^{N} \rho_i \cdot \phi_i(x, u, \dot{u}, \cdots, u^{(p)}), \tag{5}$$

where N represents the number of basis functions, $\rho_i \in \mathbb{R}^m$ is a weighting vector and ϕ_i describes a scalar basis function. Thus, the function describing the flat outputs can be expressed in a matrix form as,

$$y = P \cdot \Phi(x, u, \dot{u}, \cdots, u^{(p)}),$$

where,

$$P = \begin{bmatrix} \rho_1 & \cdots & \rho_N \end{bmatrix} \in \mathbb{R}^{m \times N} \quad , \quad \Phi = \begin{bmatrix} \phi_1 \\ \vdots \\ \phi_N \end{bmatrix} \in \mathbb{R}^N.$$

Let us also approximate ψ and χ as a linear combination of certain scalar basis functions of the flat output and a defined finite number of its derivatives. i.e.,

$$\begin{aligned} x &\approx \sum_{i=1}^{L} \kappa_i \cdot \theta_i(y, \dot{y}, \cdots, y^{(q)}) \\ u &\approx \sum_{i=1}^{L} \omega_i \cdot \theta_i(y, \dot{y}, \cdots, y^{(q)}). \end{aligned} \tag{6}$$

where $\kappa_i \in \mathbb{R}^n$ and $\omega_i \in \mathbb{R}^m$. This relation can also be expressed in matrix form,

$$x = K \cdot \Theta(\cdot) \quad , \quad u = \Omega \cdot \Theta(\cdot),$$

where $K \in \mathbb{R}^{n \times L}$, $\Omega \in \mathbb{R}^{m \times L}$ are weight matrices and the basis functions are gathered in $\Theta \in \mathbb{R}^L$. In general, given a sufficiently rich set of basis functions $\{\phi_i\}, \{\theta_i\}$, a set of matrices $\{P, K, \Omega\}$ exists such that the approximations presented above are valid within certain tolerance.

B. Optimization-based model search

Based on the previous assumptions, we propose to find the approximated model using an optimization approach. It consists in a feasibility problem where the decision variables are the matrices $\{P, K, \Omega\}$. Given sets of feasible trajectories, $x^*(t), u^*(t)$, and candidate basis functions, equations (5) and (6) must hold and therefore they can be used as constraints. The resulting feasibility problem is stated as follows,

$$\begin{aligned} &\text{find} && P, K, \Omega \\ &\text{subject to} && x^* = K \cdot \Theta(y, \dot{y}, \cdots, y^{(q)}) \\ &&& u^* = \Omega \cdot \Theta(y, \dot{y}, \cdots, y^{(q)}) \\ &&& y = P \cdot \Phi(x^*, u^*, \dot{u}^*, \cdots, u^{(p)*}). \end{aligned} \tag{7}$$

If the feasibility problem provides a non-zero solution for the decision variables, it can be stated that the system is locally differentially flat with flat output $y = P \cdot \Phi(x, u, \dot{u}, \cdots, u^{(p)})$ around x^*, u^*. Note that time dependency has been dropped for notation simplicity.

There are different alternatives to translate the sample trajectories into the constraints as well as to approximate the derivatives of y and u. Both these issues are discussed later in Section III-D.

C. Local Partial Differential Flatness

The same approach and assumptions can be used to prove partial differential flatness on a given system. Using a slightly modified notation, it is straightforward to approximate y, x_r and u,

$$y = P_r \cdot \Phi_r(\cdot) \quad , \quad x_r = K_r \cdot \Theta(\cdot) \quad , \quad u = \Omega_r \cdot \Theta(\cdot),$$

where $\Phi_r \in \mathbb{R}^N$ is a vector of scalar basis functions of the retrievable states, the inputs and a finite number of derivatives of the latter. $P_r \in \mathbb{R}^{m \times N}$, $K_r \in \mathbb{R}^{n_r \times L}$ and $\Omega_r \in \mathbb{R}^{m \times L}$ are weight matrices.

Moreover, the states are assumed to be given by a set of $\frac{n}{2}$ chains of integrators of length two, i.e. $x = [q \ \dot{q}]^T$, where q and $\dot{q}$ are called *positions* and *velocities* respectively. Thus, $q_{ur} \in \mathbb{R}^{\frac{n_{ur}}{2}}$ groups the positions related to the unretrievable states. Importantly, the partition between retrievable and unretrievable states cannot break these chains. Therefore,

$$\ddot{q}_{ur} = \begin{bmatrix} h_1(y, \dot{y}, \cdots, y^{(q)}) & \cdots & h_{\frac{n_{ur}}{2}}(y, \dot{y}, \cdots, y^{(q)}) \end{bmatrix}^T,$$

where h_i can be approximated as a linear combination of the same basis functions defined in Θ, i.e.

$$\ddot{q}_{ur} \approx \sum_{i=1}^{M} \gamma_i \cdot \theta_i(y, \dot{y}, \cdots, y^{(q)}) \tag{8}$$

where $\gamma_i \in \mathbb{R}^{\frac{n_{ur}}{2}}$. This relation can also be expressed in matrix form,

$$\ddot{q}_{ur} = \Gamma \cdot \Theta(\cdot),$$

where $\Gamma \in \mathbb{R}^{\frac{n_{ur}}{2} \times L}$ is a weighting matrix.

Following the approach presented in Section III-B, finding the partially differentially flat model is written as a feasibility problem where the constraints are built from a collection of feasible trajectories.

$$\begin{aligned} &\text{find} && P_r, \Omega_r, K_r, \Gamma \\ &\text{subject to} && u^* = \Omega_r \cdot \Theta(y, \dot{y}, \cdots, y^{(q)}) \\ &&& x_r^* = K_r \cdot \Theta(y, \dot{y}, \cdots, y^{(q)}) \\ &&& \ddot{q}_{ur}^* = \Gamma \cdot \Theta(y, \dot{y}, \cdots, y^{(q)}) \\ &&& y = P_r \cdot \Phi_r(x_r^*, u^*, \dot{u}^*, \cdots, u^{(p)*}). \end{aligned} \tag{9}$$

The partition of the states has to be determined beforehand using different combinations of the chains of integrators for the unretrievable and the retrievable states.

If the feasibility problem provides a non-zero solution for the matrices P_r, K_r, Ω_r, Γ it can be stated that the system is locally partially differentially flat around the collection of trajectories used to write the constraints.

D. Solving for Local (Partial) Differential Flatness

The method presented above requires to write the constraints in (7) and (9) using a set of sample trajectories. The following approaches may be applied to complete the formulation:

1) Discretization: Trajectories can be sampled in time. Thus, each point of the discretized trajectory (x_k, u_k) provides a set of constraints.

2) Parametrization: The sample trajectories can be fitted into a parametric (e.g., polynomial) model of time, i.e. $x^*(t) = g_D(t)$, $u^*(t) = f_D(t)$. Choosing such parametric models as basis functions (e.g, $\Phi_i = [x_i]$ and $\Theta_i = [y_i, \dot{y}_i]$), the equality constraints on the trajectories are guaranteed by the equalities on the resulting coefficients. This parametrization approach also provides the benefit of obtaining an analytical expression for the derivatives of y and u.

Indeed, for the discretization approach, finite differences can be used for numerical differentiation of y and u. However, for the special case where the basis functions Φ (Φ_r)

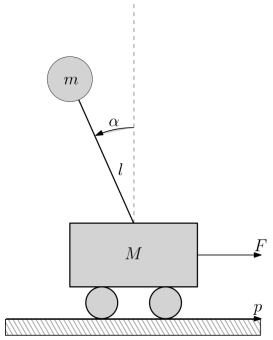

Fig. 1. Cart and Pole system. A force F is applied to the cart to move it while balancing the pole.

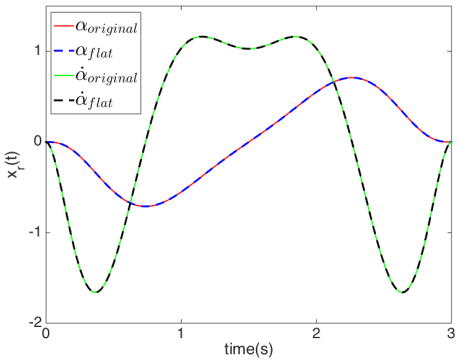

Fig. 2. Cart-Pole. Plot representing how the constraint on the retrievable states ($x_r^*(t) = K_r \cdot \Theta(\cdot)$) in the numerical search is satisfied. Here $\alpha_{original}$ and $\dot{\alpha}_{original}$ correspond to the ones of the first given trajectory $x_r^*(t)$, while α_{flat} and $\dot{\alpha}_{flat}$ are derived from the matrices product $K_r \cdot \Theta(\cdot)$ (i.e. from the numerical flat model).

are explicitly given by the (retrievable) states, and those in Θ are explicitly given by the flat output and derivatives, the sample trajectories u can be fitted into a parametric model of time such that its analytical differentiation would provide better derivatives. The same curve fitting can also be applied to the (retrievable) states, since in this case linearity allows to analytically obtain the derivatives of y (i.e. using $\Phi = x$ implies $\dot{y} = P \cdot \dot{x}$ and so on).

The results presented in this paper were obtained using the discretization approach. Moreover, since in our examples the mentioned conditions on Φ (Φ_r) and Θ apply, time polynomials of a certain degree were used to fit the sample trajectories of u and x (x_r), obtaining a better differentiation.

Finally, the feasibility problem can be solved using a non-linear programming (NLP) solver. Solving such a numerical program can be resource consuming. However, this process is executed offline and the resulting numerical flat model can be subsequently used for online trajectory generation.

IV. Results

The numerical search for local partial differential flatness was applied to three underactuated systems, i.e., "Cart-Pole", "Planar Ballbot" and a 3D quadrotor. Three feasible trajectories were used to define the constraints of the optimization problem for each system. The feasibility problems were solved using IPOPT [11], a nonlinear programming solver based on the Interior Point Method.

A. Cart-Pole

The Cart-Pole system is shown in Fig. 1. An inverted pendulum is mounted on the top of the cart. Actuated wheels allowing horizontal movement of the cart. M is the mass of the cart, m is the mass of the ball at the extreme of the pole, l is the distance between the cart-pole attachment point and the ball. α is the angle between the pole and the vertical line, p is the cart position coordinate on the plane, and F is the horizontal force applied to the cart. The state vector is defined as $x = \begin{bmatrix} p & \dot{p} & \alpha & \dot{\alpha} \end{bmatrix}^T$.

This system has been analytically proven to be partially differentially flat [1] with flat output α. The functions connecting this flat output with the input and the states are highly nonlinear although in this case they are not difficult to find, therefore this model is used here to validate the numerical approach in a very simple and controlled example.

The unretrievable states have been chosen as $x_{ur} = \begin{bmatrix} p & \dot{p} \end{bmatrix}^T$, the retrievable states as $x_r = \begin{bmatrix} \alpha & \dot{\alpha} \end{bmatrix}^T$, and the basis functions vectors

$$\Phi_r = \begin{bmatrix} \alpha & \dot{\alpha} \end{bmatrix}^T \quad , \quad \Theta = \begin{bmatrix} y & \dot{y} & \ddot{y} \end{bmatrix}^T .$$

After solving the feasibility problem the numerical flat model is available. Fig. 2 shows the pole angle (α) and angular velocity ($\dot{\alpha}$) comparing the original sample trajectory with the one provided by the local flat model, i.e.,

$$x_r = K_r \cdot \begin{bmatrix} y \\ \dot{y} \\ \ddot{y} \end{bmatrix} \quad , \quad y = P_r \cdot \begin{bmatrix} \alpha \\ \dot{\alpha} \end{bmatrix} .$$

It can be seen that the model is accurate and that it reproduces the dynamics of the system correctly.

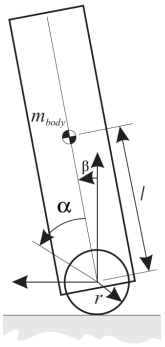

Fig. 3. Ballbot from [3]. A torque τ is applied between the body and the ball, making the system to lean. This allows the ball to move on the horizontal plane while balancing the body.

B. Planar Ballbot

The Planar Ballbot is shown in Fig. 3. It can be seen as a particular form of an inverted pendulum, and it can be modeled as a rigid rectangle mounted on a circular wheel. An input torque is applied between the body and the ball,

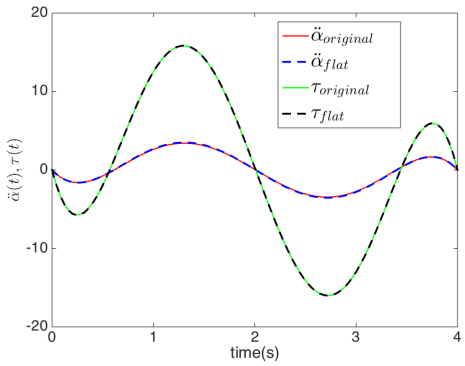

Fig. 4. Ballbot. Input τ and acceleration $\ddot{\alpha}$ trajectories. Verification of the constraints $u^*(t) = \Omega_r \cdot \Theta(\cdot), \ddot{q}^*_{ur}(t) = \Gamma \cdot \Theta(\cdot)$. Here $\tau_{original}$ and $\ddot{\alpha}_{original}$ correspond to the original trajectories, while τ_{flat} and $\ddot{\alpha}_{flat}$ correspond to $\Omega_r \cdot \Theta(\cdot)$ and $\Gamma \cdot \Theta(\cdot)$ respectively (i.e. derived from the numerical flat model).

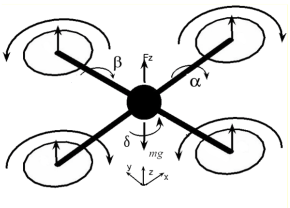

Fig. 5. Quadrotor schematics from [14]. The four inputs, given by the torques about each axis and the total trust in the z direction, make the quadrotor to move in the 3D space.

making the robot to lean. This causes the ball to move, rolling on the horizontal plane. m_{ball} is the mass of the ball, r is the radius of the ball, I_{ball} is the moment of inertia of the ball, α is the angle travelled by the ball on the horizontal plane, m_{body} is the mass of the body, I_{body} is the moment of inertia of the body, β is the lean angle and τ is the input torque. The state vector can be defined as $x = \begin{bmatrix}\alpha & \dot{\alpha} & \beta & \dot{\beta}\end{bmatrix}^T$.

The unretrievable states have been chosen as $x_{ur} = \begin{bmatrix}\alpha & \dot{\alpha}\end{bmatrix}^T$, the retrievable states as $x_r = \begin{bmatrix}\beta & \dot{\beta}\end{bmatrix}^T$ and the basis functions vectors

$$\Phi_r = \begin{bmatrix}\beta & \dot{\beta}\end{bmatrix}^T \quad , \quad \Theta = \begin{bmatrix}y & \dot{y} & \ddot{y}\end{bmatrix}^T.$$

It is important to remember that the (partial) differential flat model also allows to reconstruct the input as function of the flat outputs and derivatives. In Fig. 4 we show how the input torque τ is computed using the model and the corresponding outputs for a sample trajectory. Moreover the original and derived angular acceleration $\ddot{\alpha}$ are also shown.

C. Quadrotor

The Quadrotor model used for this paper is the one derived in [12]. p_x, p_y and p_z are the coordinates of the system in the space, α, β and δ are the Euler angles that represents the 3D roll, pitch and yaw rotations. m is the mass of the quadrotor, τ_α, τ_β and τ_δ are the input torques about each axis. F_z is the total trust in the z direction. The state vector can be defined as $x = \begin{bmatrix}p_x & p_y & p_z & \alpha & \beta & \delta & \dot{p}_x & \dot{p}_y & \dot{p}_z & \dot{\alpha} & \dot{\beta} & \dot{\delta}\end{bmatrix}^T$, and the input vector as $u = \begin{bmatrix}F_z & \tau_\alpha & \tau_\beta & \tau_\delta\end{bmatrix}$.

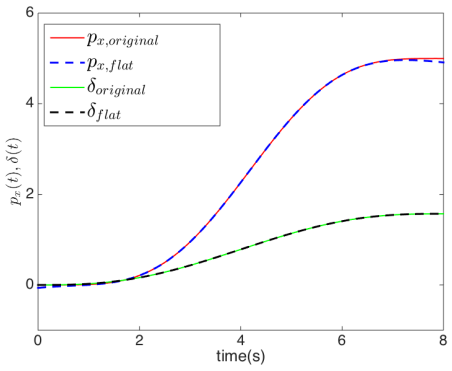

Fig. 6. Quadrotor. Plot representing how the constraints on δ and p_x in the numerical search are satisfied. Here $\delta_{original}$ and $p_{x,original}$ correspond to the ones of the first given trajectory, while δ_{flat} and $p_{x,flat}$ are derived from the numerical flat model).

The quadrotor has been proven to be differentially flat [13], [14] with analytical flat output $y = \begin{bmatrix}p_x & p_y & p_z & \delta\end{bmatrix}^T$. In this paper we test the method to derive the local differential flatness on a larger system (12 states).

The basis functions vectors have been chosen as,

$$\Phi = \begin{bmatrix}p_x & p_y & p_z & \delta\end{bmatrix}^T \quad , \quad \Theta = \begin{bmatrix}y & \dot{y} & \ddot{y} & y^{(3)} & y^{(4)}\end{bmatrix}^T.$$

In this case the numerical search has also produced a differentially flat model. Fig. 6 shows the angle δ and the position coordinate p_x given by one of the feasible trajectories as well as the ones provided by the numerical flat model.

D. Re-planning Tasks

In this section we show the use of the numerical flat models obtained above in a re-planning task. These simulation results were obtained using a standard laptop computer with 2.2Ghz Intel Core i7 (quad core) processor with 8GB 1600Mhz DDR3 RAM. The re-planning optimization problems have also been solved using IPOPT [11].

The re-planning task consists in using the local (partial) differential flat model to plan a new trajectory at certain time during the execution of one of the original trajectories. The time at which the re-planning occurs is denoted as t_{rp}.

In practice in our simulation environment we integrate forward the equation of motion following the original trajectory together with a stabilizer controller. At $t = t_{rp}$ we stop the simulation and use the flat model for re-planning, we then continue the simulation given the new state and control trajectories. We measure the time required for re-planning and it is being reported in Table I.

1) Cart-pole: The re-planning task implemented in the cart-pole corresponds to an 'emergency stop' behavior. The original trajectory starts at $p = 0$ and ends at $p = 12m$ and it has a duration of 8s. The trajectory ends in a up-right equilibrium state of the pendulum. At $t = t_{rp}$ an emergency stop is required, and an optimization problem is solved in

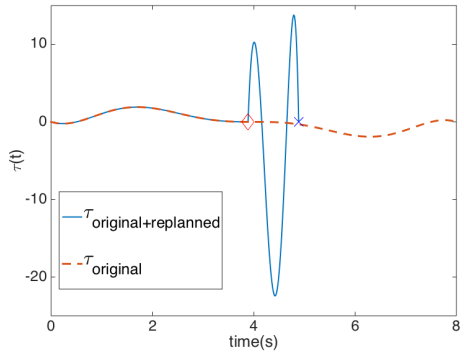

Fig. 7. Cart-pole. Comparison of the input τ in the original trajectory and in the one including the emergency brake asked at t_{rp} (indicated by the red diamond). The system aggressively try to stop as soon as possible, and it does it in $1s$ (at the time indicated by the blue cross)..

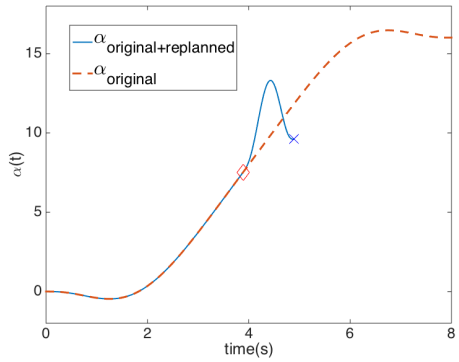

Fig. 8. Ballbot. Comparison of the ball angle α in the original trajectory and in the one including the emergency brake asked at t_{rp} (indicated by the red diamond). The system manages to stop in $1s$ (at the time indicated by the blue cross) in an earlier position than the original trajectory.

order to take the system to the up-right equilibrium position in minimum time. The resulting input trajectory is shown in Fig. 7, together with the original input trajectory. The complete resulting behavior is shown in the video accompanying this paper. It can be seen how in the re-planned trajectory a big effort is made by the system in order to stop at an equilibrium state as soon as possible.

2) Planar ballbot: A similar emergency stop behavior is requested for the planar ballbot. Fig.8 shows the original trajectory with a duration of 8s. The re-planned trajectory takes the system to the upright stable position at $t = t_{rp} + 1$s. The complete behavior is also shown in the attached video.

3) Quadrotor: Here we present a detour behavior as re-planning task. The original trajectory for the quadrotor consists in a go-to task from the initial position (0,0,0) to a still position at (5,0,0) in $8s$. At a t_{rp} the robot is asked to change the final destination to a new set of coordinates (5,2,0) in the minimum time. As shown in Fig. 9, the new task is fullfilled in $t = 4.47s$. Simulation of the re-planning task is shown in the multimedia material related to this paper.

Table I summarizes the time required for solving the numerical search and the consequent re-planning experiments

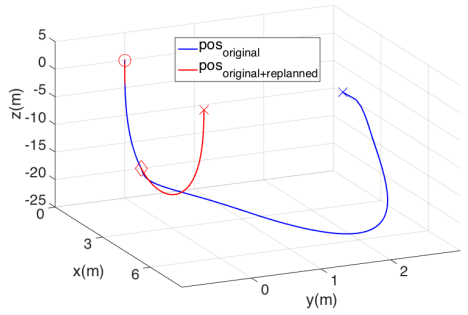

Fig. 9. Quadrotor. Comparison between the original trajectory (in red), and the trajectory including the replanning (in blue) in the 3D space. The starting point is indicated by the circle, while the ending points are indicated by the two crosses. The diamond shows the instant on which the replanning is made.

TABLE I

TIME CONSUMPTION SUMMARY

System	Numerical search	Replanning
Cart-Pole	68.89s	0.02s
Planar Ballbot	20.86s	0.03s
Quadrotor	16min 45s	0.5s

for the three systems. These results demonstrate that it is possible to perform online planning using the local (partial) differential flatness model derived in this paper. As expected, the numerical search requires more time (especially on the quadrotor) to be solved. Moreover, it is important to remark the numerical search results are strongly dependent on the given trajectories.

All the trajectories shown above have been simulated through the forward integration of (1). A Time-Varying Linear Quadratic Regulator [10] (TVLQR) has been used as stabilizer. The results of these simulations are shown in the video.

V. DISCUSSION

A key benefit of the numerical search is the simplification in the derivation of the flat model avoiding the manipulation of the equations of motion. The use of basis function to approximate the flat model simplifies the formulation in the parameters space. Such parametrization makes the optimization time smaller in an online re-planning task.

On the other hand, this method requires partitioning the states beforehand. This is a design decision and it can be guided by the knowledge of the dynamics of the system. However results in [1] introduces a lemma that, providing sufficient conditions for partial differential flatness, proves how to partition the states when certain conditions on some of the actuated states are verified.

Another critical aspect of the approach is the dependency on sets of feasible trajectories and sufficiently rich basis functions. Trajectories can be generated using methods that are more time expensive, like standard numerical optimization approaches [17], [18]. Regarding the selection of an

effective set of basis functions, knowledge of the dynamics of the system can be exploited, focusing initial guesses on simple functions. In the examples provided in this paper a very simple vector of basis functions has been used, simplifying the implementation. However, the method is still very sensible to this choice.

For the case of the quadrotor, the trajectories used for obtaining the model do not include the highly nonlinear take-off and landing regimes as it requires much more elaborated basis functions. However, using the local model demonstrated to be effective for feasible solutions in the re-planing stage. In the same example the re-planning time shown in Table I is greater than the first two examples, but it is important to recognize that the implementation has not been optimized for reducing convergence time.

Regarding further extensions, providing feasible trajectories spanning a wider part of the dynamics and a richer set of basis functions, together with a proper method to select them, would result in more consistent approximations, making this approach even more general. Moreover, a more efficient implementation would further reduce convergence time for the optimization problems.

VI. Conclusion

We have introduced a numerical method in order to find an approximated (partially) differentially flat model given some feasible inputs and states trajectories. The method has been successfully proven on three underactuated examples, where the system dynamics has been correctly approximated. Moreover, the new models have been used in online re-planning tasks, resulting in trajectories that have been shown to be stabilizable simulating the original equations of motion.

References

[1] S. Ramasamy, G. Wu, and K. Sreenath, Dynamically feasible motion planning through partial differential flatness. In Robotics: Science and Systems Conference (RSS), 2014.

[2] R. Featherstone, Rigid Body Dynamics Algorithms. Springer, 2008.

[3] M. Shomin and R. Hollis, Differentially flat trajectory generation for a dynamically stable mobile robot. In IEEE Internation Conference on Robotics and Automation (ICRA), 2013.

[4] H. Sira-Ramirez and S. K. Agrawal, Differentially flat systems. CRC Press, 2004.

[5] J. Lvine and D. V. Nguyen, Flat output characterization for linear systems using polynomial matrices. Systems & control letters, 2003.

[6] G. G. Rigatos, Nonlinear Control and Filtering Using Differential Flatness Approaches: Applications to Electromechanical Systems. Springer, 2015.

[7] R. M. Murray, M. Rathinam and W. Sluis, Differential Flatness of Mechanical Control Systems: A Catalog of Prototype Systems. Proceedings of the 1995 ASME International Congress and Exposition, 1995.

[8] T. Dear, S. D. Kelly, M. Travers and H. Choset, Motion Planning and Differential Flatness of Mechanical Systems on Principal Bundles. ASME 2015 Dynamic Systems and Control Conference, 2015.

[9] C. P. Tang, P. T. Miller, V. N. Krovi, J. C. Ryu, and S. K. Agrawal, Differential-Flatness-Based Planning and Control of a Wheeled Mobile Manipulator: Theory and Experiment. IEEE/ASME Transactions on Mechatronics, 2011.

[10] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, Lqr-trees: Feedback motion planning via sums-of-squares verification. International Journal of Robotics Research, 2010.

[11] A. Wachter and L.T. Biegler, On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Mathematical Programming, 2006.

[12] A. Di Cesare, K. Gustafson, and P. Lindenfeler, Design Optimization of a Quad-Rotor Capable of Autonomous Flight. BS Report, Aerospace and Mechanical Dept., Worchester Polytechnic Institute, Worchester, MA, 2009.

[13] S. Formentin and M. Lovera, Flatness-based control of a quadrotor helicopter via feedforward linearization. IEEE Conference on Decision and Control and European Control Conference (CDC-ECC), 2011.

[14] I. D. Cowling, O. A. Yakimenko, J. F. Whidborne, and A. K. Cooke, A Prototype of an Autonomous Controller for a Quadrotor UAV. Control Conference (ECC), 2007.

[15] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, Whole-body model-predictive control applied to the hrp-2 humanoid robot. IEEE-RSJ Int. Conference on Intelligent Robots and Systmes (IROS) , 2015.

[16] Michael Posa, Cecilia Cantu, and Russ Tedrake, A direct method for trajectory optimization of rigid bodies through contact. International Journal of Robotics Research, 2014.

[17] D. Pardo, L. Moller, M. Neunert, A. Winkler, and J. Buchli, Evaluating direct transcription and nonlinear optimization methods for robot motion planning. IEEE Robotics and Automation Letters, 2016.

[18] John T. Betts, Survey of Numerical Methods for Trajectory Optimization. Journal of Guidance, Control and Dynamics, 1998.

[19] A. Banos, F. Lamnabhi-Lagarrigue, F. J. Montoya. Advances in the Control of Nonlinear Systems. Springer, 2001.