# Differential Geometric Approaches to the Motion Planning Problem of the 2-link Acrobot

by

Ali Baghani

Student of
Sharif University of Technology
Visiting
Mita Laboratories
Tokyo Institute of Technology

Sotsuron dissertation submitted to the Department of
Control and Mechanical Engineering of
Tokyo Institute of Technology
2002

In the Name of God

## Abstract

In this dissertation I study the *motion planning problem* for a two link free flying robot, the *acrobot*. As governed by the conservation of momentum law, the constraint on this system is a *nonholonomic constraint*. The goal sought is to steer the acrobot from an initial posture to a final one in a specified amount of time, while it is flying in the air with a constant angular momentum. Two different approaches based on differential geometric nonlinear control theory are adopted and used and a comparison between the results is made. The first is the so called *path deformation method* while the second one is an attempt of *Nilpotentization*. Simulations are made to validate the results.

## Dedication

To my Mother

Zari Samimi

# Acknowledgements

First of all I wish to express my sincerest gratitude to the Educational Committee of the Department of Electrical Engineering, Sharif University of Technology who provided me with the opportunity to conduct this research in Japan, and in particular to Prof. M. Alavi the ex-chairman of the department and Prof. Sadooghi my supervisor in SUT. Also I am deeply grateful to Prof. A. Vafai, the chairman of the Office of International and Scientific Coop., Sharif University of Technology whose support in every step made me confident in my study.

Then I shall express my thanks to Prof. Mita Ts. who introduced this problem to me and in whose laboratory I worked and to Prof. Kurabayashi D. for his lectures. I do appreciate the foreign students staying at Mita laboratory at that time for without their intellectual help and console this work would have been impossible, among who I am the most grateful to Pavel Pratsch from Czech R., who gave me the inspiration for study of nonlinear control systems from differential geometric point of view, and provided me with extremely useful discussions. I shall thank Tang Zh. from China who provided me with the best console and friendship and J. Nicolas from France, who is always encouraging. Jurachart J. from Thailand, my tutor, has always helped me with computer problems as well as problems with the everyday life in Japan, and Yang E. from China has been a very good friend.

Then I would like to express my best thanks to Anne Patrikainen from Finland, whose friendship made the life in Japan bearable for me and also to Ole Edsberg from Norway. Being with these people was a real comfort after a day of hard work or two.

Now and forever is the time to thank my family for their love, support, encouragement, and console. I have to confess, words cannot express my deepest appreciation and gratitude to my mother, whose love and sacrifice for me never ends. The least I have to do is to dedicate this thesis to her.

Ali Baghani
2002/07/19

Table of Contents

| Chapter | | Page |
|---|---|---|
| 1 | Introduction | 6 |
| 2 | Derivation of the Dynamical Model | 8 |
| 3 | Motion Planning Using the Path Deformation Method | 12 |
| 4 | Nilpotentization | 19 |
| 5 | Conclusion | 25 |
| 6 | Bibliography | 26 |

Chapter 1

# Introduction

One of the challenging aspects of nonlinear control theory is the *motion planning problem* for systems governed by *nonholonomic constraints*. As opposed to *holonomic constraints* they are by no means easily tractable. A holonomic constraint is a one which is described by an *integrable 1-form* (first order exterior differential form) on the *state space manifold* of the system. For example $dx = 0$ defines a holonomic constraint on $R^3$. By integrating such kind of a constraint, a scalar function on the manifold is obtained, as is $f(x, y, z) = x$ in our case. The level surfaces of this function, are in fact new imbedded submanifolds which define the true state space of the system, i.e. we have an algebraic relation between the state variables of the system or equivalently *a reduction in the dimension of the state space*. To see this, if the initial state of the system of our example has $x(0) = 1$ then because of the holonomic constraint, $x(t) = 1 \quad \forall t$, so the true state space is now $R^2$. But for a nonholonomic constraint, no reduction in the dimension of the state space occurs. In fact a nonholonomic constraint is a constraint on the direction of the velocity of the system at each point of the state space, but a one which does not restrict your access to the whole possible states. The best example illustrating this concept is parking the car. When you want to park your car between two other parked cars, the moment you are beside the empty space, you cannot push your car sideways into the place, because of the nonholonomic constraint induced by your wheels. At that moment you can only go forward and backward, never the less this does not mean that you can never park your car there. You have a restriction on the direction of your motion but no restriction on the accessible points. Now if you lock the steering wheel, the constraint becomes holonomic, which means you can only go straight (assuming you have locked your steering wheel in the middle position).

The motion planning problem is now the systematic derivation of an open loop control which steers the system from an initial point in the state space to a final point, in an specified amount of time. Apart from its underlying mathematical beauty this problem is important from the practical and technological point of view. In fact automation of cargo handling using trucks and trailers, for example in airports, has been one of the hottest research issues in the field, attracting attention in the last decade.

In this dissertation I use two different methods based on differential geometry to solve the problem for a two link flying acrobot. First in chapter 2 I derive the dynamical model of the system and formulate the problem introducing the necessary notation. In chapter 3 I use the *path deformation method* which is the extension of the continuation method used in solving nonlinear equations to the path planning of dynamic systems. In chapter 4 I use the *nilpotentization method* to first derive feedback control laws, which transform my system into a simple form (namely a system whose control vector fields generate a nilpotent Lie algebra), and then use this simple form to solve the motion planning problem. The first method is quite general and can be adapted to other systems, while the second one is more problem specific. At the end of each chapter simulation results are given. Chapter 5 is devoted to the conclusion.

# Chapter 2

## Derivation of the Dynamical Model

In this chapter I will formulate the dynamical model of the system I am going to use for the purpose of control. The mechanical system under consideration is a 2-link robot where the joint between the two links is actuated (Fig.1). This robot is a simplified model of an acrobat or a diver, consisting of *the torso*, *the waist* and *the legs*. (Fig. 2)

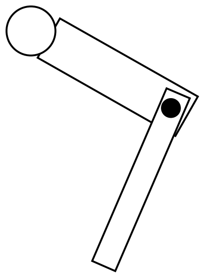


Fig.1 The Acrobot

Fig.2 The Diver

*Neglecting the friction* with the air, the law governing the motion of the acrobot while it is falling, is the *conservation of the angular momentum*. In fact the motion of the center of mass of the acrobot, is not controllable and its trajectory is a hyperbola with its parameters depending on the initial velocity and location of the acrobot. So we are not concerned with that aspect but the posture of the acrobot, i.e. the angular motion of the two links. Fix a reference frame on the center of mass (CMRF) which moves with it but its orientation is fixed with respect to a reference frame connected to the ground. Let $\theta$ denote the angle between the leg and the positive x axis of CMRF and let $\phi$

denote the angle between the torso and the leg where positive senses are counterclockwise(Fig.3).

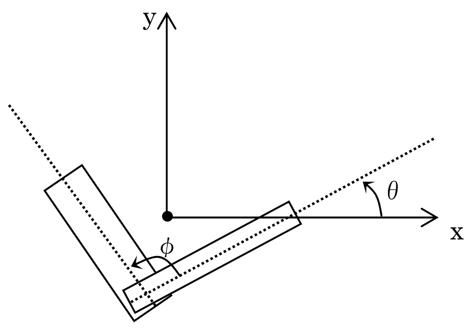


Fig.3 The angles

The total angular momentum of an n-link planar robot can be described as [1]

$$P_G = P_G^z \cdot \hat{z}$$
$$P_G^z = \sum_{i=1}^{n} J_i \dot{\theta}_i + \frac{1}{m_0} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} m_i m_j \{(x_j - x_i)(\dot{y}_j - \dot{y}_i) - (\dot{x}_j - \dot{x}_i)(y_j - y_i)\} \tag{1}$$

Where,

$J_i$ is the angular momentum of link i about its center of mass.

$\dot{\theta}_i$ is the angular velocity of link i with respect to CMRF.

$m_0$ is the total mass of the robot.

$m_i$ is the mass of link i.

$x_i, y_i$ are the Cartesian coordinates of the center of mass of link i with respect to any point, moving or fixed.

We shall use the following notation (Fig.4)

$M$ the mass of the torso.

$m$ the mass of the leg.

$J$ the moment of inertia of the torso about its center of mass.

$j$ the moment of inertia of the leg about its center of mass.

$L$ the distance from the center of mass of the torso to the joint.

$l$ the distance from the center of mass of the leg to the joint.

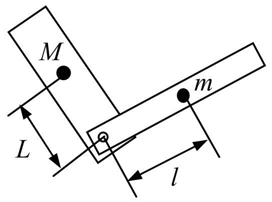


Fig. 4 The parameters

Taking the joint as the reference point for coordinates and using formula (1) we have,

$$P_G^z = j\cdot\dot{\theta} + J\cdot(\dot{\phi}+\dot{\theta}) + \frac{mM}{m+M}\{(L\cos(\theta+\phi) - l\cos(\theta))(L\cos(\theta+\phi)\cdot(\dot{\theta}+\dot{\phi}) - l\cos(\theta)\cdot\dot{\theta}) - (L\sin(\theta+\phi) - l\sin(\theta))(-L\sin(\theta+\phi)\cdot(\dot{\theta}+\dot{\phi}) + l\sin(\theta)\cdot\dot{\theta})\}$$

Since this momentum is constant, we use $P_0$ to denote it,

$$P_0 = \{j + J + \frac{mM}{m+M}(L^2 + l^2 - 2Ll\cos(\phi))\}\cdot\dot{\theta} + \{J + \frac{mM}{m+M}(L^2 - Ll\cos(\phi))\}\cdot\dot{\phi} \quad (2)$$

This is the basic equation from which the state space model of the system is derived. In this dissertation we assume that *the actuator can determine the instantaneous angular velocity between the two links.* In other words we assume that the value of $\dot{\phi}$ can be determined at each instant, or equivalently it is a control input to the system. Although this assumption neglects the intrinsic electrical and mechanical dynamics of the actuator, by deriving a continuous control function for $\dot{\phi}$ a piecewise continuous target trajectory for the *torque* of the actuator can be derived. Following this trajectory is a well understood problem for different types of practical actuators.

Now let us take $\phi$ and $\theta$ as the states and denote $\dot{\phi}$, the control input by $u$.

Using equation (2) the state space model of the system can be derived as,

$$\begin{bmatrix} \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \dfrac{P_0}{M_1 + A_1 \cos\phi} \\ 0 \end{bmatrix} + \begin{bmatrix} -\dfrac{M_2 + A_2 \cos\phi}{M_1 + A_1 \cos\phi} \\ 1 \end{bmatrix} .u \tag{3}$$

Where the parameters are given by,

$$M_1 = j + J + \frac{mM}{m+M}(L^2 + l^2)$$
$$A_1 = -2\frac{mM}{m+M}Ll$$
$$M_2 = J + \frac{mM}{m+M}L^2$$
$$A_2 = -\frac{mM}{m+M}Ll$$

The state space model (3) will be used extensively in the following chapters. As is obvious because of the presence of the cosine terms, this is a *nonlinear control system*. The goal is, given the initial condition P=[ $\theta$ (0) $\phi$ (0)]`, to find a nice control input u(t) which drives the system to state Q=[ $\theta$ (T) $\phi$ (T)]` in the specified time T. So the control task has two outputs but only one input, so this is an *underactuated* system. Moreover in the terminology of nonlinear control systems, the presence of an uncontrolled vector field in the state space equations is described by *drift.* The presence of the drift term makes the control task very difficult, because for systems with drift, even when you do not apply any input, the system is moving by its drift term, so your control inputs must cope with the internal dynamics of the system. Finally if we write equation (2) as,

$$\{j + J + \frac{mM}{m+M}(L^2 + l^2 - 2Ll\cos(\phi))\} \cdot d\theta + \{J + \frac{mM}{m+M}(L^2 - Ll\cos(\phi))\} \cdot d\phi - P_0 \cdot dt = 0$$

It can be easily checked out that this one-form is not a complete differential, so it cannot be integrated, and hence the system is *nonholonomic.*

# Chapter 3

# Motion Planning Using the Path Deformation Method

The model of the robot under consideration is

$$\Sigma : \begin{bmatrix} \dot{\theta} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \dfrac{P_0}{M_1 + A_1 \cos\varphi} \\ 0 \end{bmatrix} + \begin{bmatrix} -\dfrac{M_2 + A_2 \cos\varphi}{M_1 + A_1 \cos\varphi} \\ 1 \end{bmatrix} .u \qquad (0)$$

The goal is, given the initial condition P=[$\theta(0)$ $\varphi(0)$]`, to find a nice control input u(t) which drives the system to state Q=[$\theta$(T) $\varphi$(T)]` in the specified time T.

The Path Deformation Method is the extension of the continuation method used in solving nonlinear equations to path planning of dynamic systems. First we explain the continuation method and then its extension.

Suppose M and N are smooth manifolds and N is connected. Suppose $\Phi : M \longrightarrow N$ is a smooth map, and we want to solve the equation $\Phi$(x)=y. To do this we first choose an **arbitrary** point $\bar{x}$ and find its image $\bar{y} = \Phi(\bar{x})$. Next we define a path from $\bar{y}$ to y in N, i.e. a smooth function $\pi : [0,1] \longrightarrow N$ and then try to pull this path back to M. This means we seek for a path $\Pi : [0,1] \longrightarrow M$ such that $\Phi \text{o} \Pi = \pi$. If we are successful then clearly $\Pi$(1)=x , the solution of our equation. Although this might seem more difficult than the original problem, since we have to solve the equation $\Phi$ o $\Pi$ = $\pi$ for the unknown function $\Pi$, but in fact under certain **nondegeneracy** and **nonexplosion** conditions, this can be accomplished by solving an ordinary differential equation, which leads itself to a numerical solution for the problem.

Actually we want to have $\Phi(\Pi(\tau))=\pi(\tau)$ for $\tau\in[0,1]$, and $\Pi(0)=\bar{x}$. Differentiating the former equation we get,

$$\begin{aligned} &D\Phi(\Pi(\tau))\cdot\dot{\Pi}(\tau)=\dot{\pi}(\tau) \\ &\Pi(0)=\bar{x} \end{aligned} \tag{1}$$

It is clear that by solving this equations we can find the desired $\Pi$ (for two real smooth functions whose derivatives are the same at every point and their values are the same at some point, are equal at every point). From equation (1) the nondegeneracy condition to be met, which was mentioned earlier is apparently: rank(D$\Phi$($\Pi$(x)))=dim(N), which means that $\Phi$ must be an immersion. Obviously this is not always the case. If D$\Phi$ has singularities, then still we may define a submanifold of M which we shall call M\`, by excluding the points of singularity, and then use the restriction of $\Phi$ to M\`, but we must note that now the image under $\Phi$ of M\` is not N, it is a submanifold of it called N\`, which might not be connected anymore. In this case more work must be done to choose the initial $\bar{x}$ so as to guarantee that $\bar{y}$ is in the same connected submanifold of N\` as y. Moreover the path $\pi$ must be designed completely in N\` and not in N, in order to avoid singularities. After this discussion we may choose a right-inverse for D$\Phi$ such as the pseudoinverse

$$P(x)=D\Phi(x)^T\cdot(D\Phi(x)\cdot D\Phi(x)^T)^{-1} \tag{2}$$

and express the equation (1) as

$$\dot{\Pi}(\tau)=P(\Pi(\tau))\cdot\dot{\pi}(\tau) \tag{3}$$

$$\Pi(0)=\bar{x}$$

Under the conditions discussed above this differential equation can be solved for $\Pi$ and $\Pi(1)$ is the desired solution of our initial equation.

To extend this method to our problem, we make the following considerations,

1. Let U denote the set of admissible control inputs, in which we are seeking a solution.

In this research we take for U the set of all analytic functions on [0,T]. This set plays the role of M.

2. Let S denote the state space of our dynamic system. This set plays the role of N.
3. Define $\Phi_p^t : U \longrightarrow S$, where $\Phi_p^t(u)$ is the state reached by the system $\Sigma$ at time t when the control input u(t) is applied to it, starting from the initial state p. Now we are interested in the state reached at time T, so $\Phi_p^T$ (superscript T) plays the role of $\Phi$.

In the new formulation, our goal is to solve the equation $\Phi_p^T(u) = q$ for u. The method is exactly the same as the continuation method. The only thing left for writing down equation (1) is the differential of $\Phi_p^T$ at an input $u^0$, i.e. $D\Phi_p^T(u)\big|_{u^0}$. This differential corresponds to the variation in the final state of the system due to a variation in the control input. To calculate this differential, consider the more general differential $D\Phi_p^t(u)\big|_{u^0}$ (superscript t), which corresponds to the variation in the state of the system at time t due to a variation in the control input. Let us take the new input $u(t) = u^0(t) + s \cdot \zeta(t)$. Then define,

$$y(t) = \left| \frac{d}{ds} \Phi_p^t(u) \right|_{s=0} \tag{4}$$

Clearly y(t) is the value of $D\Phi_p^t(u)\big|_{u^0}$ in the direction of the variation $\zeta(t)$ (actually $\zeta(t)$ is a tangent vector to U at $u^0(t)$ ). We know that y(0)=0 (since $\Phi_p^0(u) = p$ which does not depend on u hence neither on s). To calculate the value of y(T), let us differentiate equation (4), with respect to time

$$\dot{y}(t) = \frac{d}{dt}\left( \frac{d}{ds} \Phi_p^t(u)\Big|_{s=0} \right)$$

$$= \frac{d}{ds}\left(\frac{d}{dt}\Phi_p^t(u)\right)\Big|_{s=0}$$

$$= \frac{d}{ds}\{f(\Phi_p^t(u)) + g(\Phi_p^t(u))\cdot u(t)\}\Big|_{s=0}$$

Where f and g are as usual,

$$f(p) = \begin{bmatrix} \dfrac{P_0}{M_1 + A_1\cos\varphi} \\ 0 \end{bmatrix}$$

$$g(p) = \begin{bmatrix} -\dfrac{M_2 + A_2\cos\varphi}{M_1 + A_1\cos\varphi} \\ 1 \end{bmatrix} \tag{5}$$

Therefore we have,

$$\dot{y}(t) = Df(\Phi_p^t(u^0 + s\zeta))\Big|_{s=0} \cdot \frac{d}{ds}\Phi_p^t(u)\Big|_{s=0}$$

$$+ Dg(\Phi_p^t(u^0 + s\zeta))\Big|_{s=0} \cdot \frac{d}{ds}\Phi_p^t(u)\Big|_{s=0} \cdot (u^0 + s\zeta)\Big|_{s=0}$$

$$+ g(\Phi_p^t(u^0 + s\zeta))\Big|_{s=0} \cdot \zeta$$

Using definition (4) we have,

$$\dot{y}(t) = Df(\Phi_p^t(u^0)) \cdot y(t) + Dg(\Phi_p^t(u^0)) \cdot y(t) \cdot u^0(t) + g((\Phi_p^t(u^0)) \cdot \zeta(t)$$

$$\therefore \dot{y}(t) = \left[Df(\Phi_p^t(u^0)) + Dg(\Phi_p^t(u^0)) \cdot u^0(t)\right] \cdot y(t) + \left[g((\Phi_p^t(u^0)) \cdot \zeta(t)\right] \tag{6}$$

This is a differential equation by which y(t) can be found (the initial condition is y(0)=0 as was shown before). If we denote the solution of this equation by Y(t) clearly the mapping which assigns to each $\zeta$ (t) the value Y(T) is the long sought $D\Phi_p^T(u)\big|_{u^0}$. (As a matter of fact this mapping assigns to each tangent vector to U at $u^0$ i.e. each $\zeta$, a tangent vector to S at $\Phi_p^T(u^0)$, namely Y(T) evaluated using that $u^0$ and $\zeta$).

Thus far the main tools have been developed. By assuming that an analytic control input exists which solves our problem, we take it's Taylor series expansion about t=0 and therefore have a basis for the tangent space to U at $u^0$, namely the $t^i$. Therefore we have

$$\zeta_i(t) = a_i t^i \tag{7}$$

$$u(t) = u^0(t) + \sum_{i=0}^{\infty} \zeta_i(t) \tag{8}$$

Using definitions (5) we have,

$$Df = \begin{bmatrix} 0 & \dfrac{-P_0 A_1 \sin\varphi}{(M_1 + A_1 \cos\varphi)^2} \\ 0 & 0 \end{bmatrix}$$

$$Dg = \begin{bmatrix} 0 & \dfrac{(A_2 M_1 - A_1 M_2)\sin\varphi}{(M_1 + A_1 \cos\varphi)^2} \\ 0 & 0 \end{bmatrix} \tag{9}$$

Now if $u^0(t)$ is applied to the system, we will have,

$$\varphi^{\#}(t) = \varphi(0) + \int_0^t u^0(\tau)d\tau \tag{10}$$

Using this $\varphi^{\#}$ as the argument in equations (9) and replacing them in equation (6), we get

$$\begin{bmatrix} \dot{y}_1(t) \\ \dot{y}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & \dfrac{-P_0 A_1 \sin\varphi^{\#} + (A_2 M_1 - A_1 M_2)\sin\varphi^{\#} u^0}{(M_1 + A_1 \cos\varphi^{\#})^2} \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} + \begin{bmatrix} -\dfrac{M_2 + A_2 \cos\varphi}{M_1 + A_1 \cos\varphi} \\ 1 \end{bmatrix} \zeta(t)$$

These equations are easily integrated,

$$y_2(T) = \int_0^T \zeta(\tau)d\tau \tag{11}$$

$$y_1(T) = \int_0^T \left(\frac{-P_0 A_1 \sin\varphi^{\#} + (A_2 M_1 - A_1 M_2)\sin\varphi^{\#} u^0}{(M_1 + A_1 \cos\varphi^{\#})^2} \int_0^\tau \zeta(\varepsilon)d\varepsilon - \frac{M_2 + A_2 \cos\varphi^{\#}}{M_1 + A_1 \cos\varphi^{\#}} \zeta(\tau)\right)d\tau$$

Now using $\zeta = \sum_{i=0}^{\infty} \zeta_i(t) = \sum_{i=0}^{\infty} a_i t^i$, we can calculate the coefficients of $a_i$ in the above formulae, so we arrive at,

$$\begin{bmatrix} y_1(T) \\ y_2(T) \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} & \cdots \\ A_{21} & A_{22} & A_{23} & A_{24} & \cdots \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \end{bmatrix} \tag{12}$$

As mentioned earlier the matrix A is exactly $D\Phi_p^T(u)\big|_{u^0}$ which can be used in an equation similar to equation (1) to find the desired control input u(t).

As a matter of fact as can be guessed from the form of equation (12) we might not need so many coefficients in $\zeta$ and two might be enough. In the simulation results also this was confirmed and the additional coefficients have little effect on the solution. So we used only two adjustable coefficients for matching equation (12), but the existence of additional conditions such as a certain value for the input at t=0 or for its derivative, forces us to assume more degrees of freedom for the input. In the below simulation 6 degrees of freedom was assumed for u(t). 5 successive points were to be passed through (6 points adding the initial state).

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| T | = | [0 | 2 | 4 | 6 | 8 | 10] |
| $\theta$ | = | [-0.5 | 0.9 | 2.3 | 3.7 | 5.1 | 6.5]* $\pi$ |
| $\phi$ | = | [-1 | -2 | -2 | -2 | -2 | -1]* $\pi$ |

A cubic Spline interpolation was used to interpolate the ( $\phi$ ,t) pairs with the condition of zero derivatives at the end points. Then the first and the second derivative of this interpolation at the 6 points of interest were used as additional conditions for the value of u(t) and its first derivative. The validity of the theory is well confirmed according to the simulation results.

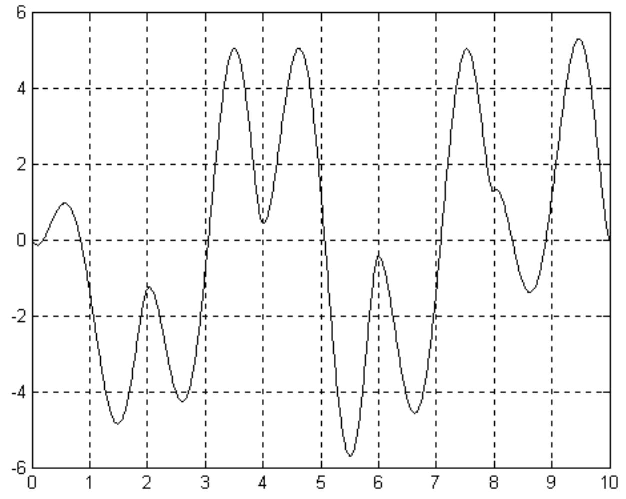


Fig.1. Input u(t)

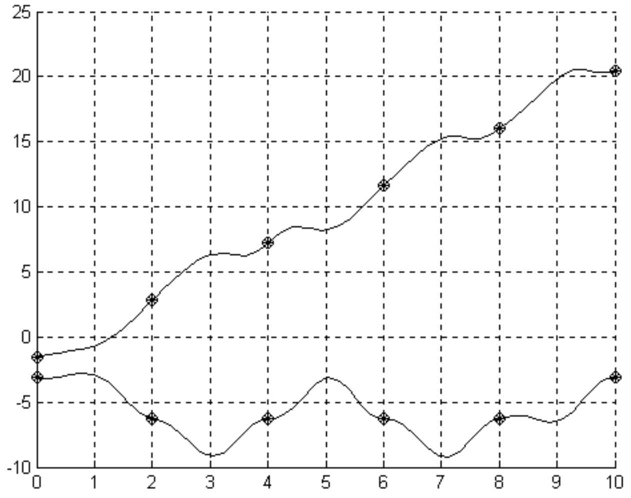


Fig.2. States

Upper : $\theta$

Lower : $\phi$

# Chapter 4

# Nilpotentization

The model of the robot under consideration is

$$\Sigma : \begin{bmatrix} \dot{\theta} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \dfrac{P_0}{M_1 + A_1 \cos\varphi} \\ 0 \end{bmatrix} + \begin{bmatrix} -\dfrac{M_2 + A_2 \cos\varphi}{M_1 + A_1 \cos\varphi} \\ 1 \end{bmatrix} .u \tag{0}$$

which can be written as,

$$\dot{Y} = Y^1 + Y^2 u \tag{1}$$

We want to replace $Y^1$ and $Y^2$ with two other vector fields $Y'^1$ and $Y'^2$ such that locally the distribution spanned by the latter is the same as the former (which means that we can transform the system through a feedback into one whose dynamics is described by $Y'^1$ and $Y'^2$), but with the property that the Lie Algebra generated by $Y'^1$ and $Y'^2$ is nilpotent. If we are able, the system thus obtained has the good property that its solution can be expressed in a simple form by a finite combination of flows of some vector fields (in fact the P-Hall basis for the Lie Algebra mentioned above) with coefficients which are computable in terms of iterated integral(s) of the input(s). Assume that we are able to do so. By construction,

$$Y'^1 = f_{11}(Y)Y^1 + f_{12}(Y)Y^2$$

$$Y'^2 = f_{21}(Y)Y^1 + f_{22}(Y)Y^2$$

$$\dot{Y} = Y'^1 v_1 + Y'^2 v_2 \tag{2}$$

The problem is that to realize the above equation we must have control over $Y^1$ i.e. the drift term in the original system. To gain this control we use the idea of reparameterization of the time, by introducing a new virtual time variable $\varepsilon$,

$$\frac{d}{d\varepsilon}\varphi = u_2$$
$$\frac{d}{d\varepsilon}t = u_1$$
$$\frac{d}{d\varepsilon}\theta = \frac{P_0}{M_1 + A_1\cos\varphi}u_1 - \frac{M_2 + A_2\cos\varphi}{M_1 + A_1\cos\varphi}u_2$$

or in matrix form,

$$\Sigma : \begin{bmatrix} \dot{\varphi} \\ \dot{t} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ \dfrac{P_0}{M_1 + A_1\cos\varphi} \end{bmatrix} .u_1 + \begin{bmatrix} 1 \\ 0 \\ -\dfrac{M_2 + A_2\cos\varphi}{M_1 + A_1\cos\varphi} \end{bmatrix} .u_2 \qquad (3)$$

The strong restriction we must pay attention to is that we cannot go back in time so although the above system is similar to a driftless system, the input $u_1$ must always be greater than zero. We will see that this restriction will turn into a big obstacle to the designing of the control input for the transformed system. Now let $Y^1$ and $Y^2$ denote the vector fields in equation 3. The method for nilpotentization is based on the construction of a local model on $R^3$ by defining two model vector fields $X^1$ and $X^2$ on $R^3$ such that the Lie Algebra generated by $X^1$ and $X^2$ is nilpotent. Then the next step is to find a diffeomorphism $\Phi : R^3 \longrightarrow M$ whose differential maps $X^1$ and $X^2$ to span$\{Y^1, Y^2\}$ in fact the images of $X^1$ and $X^2$ under $\Phi_*$ are exactly the desired $Y'^1$ and $Y'^2$. The main characteristic the model must posses for the existence of such a diffeomorphism is that Dim($D^i\{X^1, X^2\}$) = Dim($D^i\{Y^1, Y^2\}$) where $D^i$ is the derived distribution defined recursively by,

$$D^0 = span\{X^1, X^2\}$$
$$D^{i+1} = span\{D^i, [D^0, D^i]\}$$

For our case, some controllers have a singularity at $\varphi = 0$ because $[Y^1, Y^2]$

vanishes at $\varphi = 0$. We choose our region of interest a neighborhood of Y=0. In this neighborhood the Taylor expansion of the vector fields $Y^1$ and $Y^2$ prove to be a very appropriate choice. So we have

$$X^1 = \begin{bmatrix} 0 \\ 1 \\ \dfrac{P_0}{M_1 + A_1} + \dfrac{P_0.A_1}{(M_1 + A_1)^2} x_1^{\ 2} \end{bmatrix}$$

$$X^2 = \begin{bmatrix} 1 \\ 0 \\ -\dfrac{M_2 + A_2}{M_1 + A_1} + \dfrac{A_2.M_1 - A_1.M_2}{(M_1 + A_1)^2} x_1^{\ 2} \end{bmatrix}$$

Using these model vector fields and doing some cumbersome calculations one finds out that,

$$Y'^1 = \begin{bmatrix} 0 \\ \dfrac{\varphi}{\sin\varphi} \dfrac{(M_1 + A_1 \cos\varphi)^2}{(M_1 + A_1)^2} \\ \dfrac{P_0.\varphi}{\sin\varphi} \dfrac{M_1 + A_1 \cos\varphi}{(M_1 + A_1)^2} \end{bmatrix}$$

$$Y'^2 = \begin{bmatrix} 1 \\ t.\left(\dfrac{1}{\varphi} - \dfrac{2A_1 + \cos\varphi.(M_1 - A_1 \cos\varphi)}{\sin\varphi.(M_1 + A_1 \cos\varphi)}\right) \\ P_0.t.\left(\dfrac{1}{\varphi.(M_1 + A_1 \cos\varphi)} - \dfrac{2A_1 + \cos\varphi.(M_1 - A_1 \cos\varphi)}{\sin\varphi.(M_1 + A_1 \cos\varphi)^2}\right) - \dfrac{M_2 + A_2 \cos\varphi}{M_1 + A_1 \cos\varphi} \end{bmatrix}$$

To transform the original system into one described by $Y'^1$ and $Y'^2$ we have the feedback inputs,

$$u_1 = \frac{\varphi}{\sin\varphi}\frac{(M_1 + A_1\cos\varphi)^2}{(M_1 + A_1)^2}v_1 + t.\left(\frac{1}{\varphi} - \frac{2A_1 + \cos\varphi.(M_1 - A_1\cos\varphi)}{\sin\varphi.(M_1 + A_1\cos\varphi)}\right)v_2$$

$$u_2 = v_2$$

It can be verified easily that the only nonzero elements of the P-Hall basis of the Lie Algebra generated by $Y'^1$ and $Y'^2$ are $Y'^1, Y'^2, [Y'^1, Y'^2], [Y'^2, [Y'^1, Y'^2]]$ which means that if we were free to chose any $v_1$ and $v_2$, designing a local controller for the system would be very easy. But the main problem is the restriction $u_1 > 0$.

In fact by assuming parameterized forms for the control inputs $v_1$ and $v_2$, say using $m_i$ the system can easily be integrated and the result is a set of equations in $m_i$ together with an inequality in $m_i$ and $\varepsilon$ which must be satisfied for all values of $\varepsilon$ in [0,1], when the solution of the equations is substituted for $m_i$. Although the equations are very easy to solve numerically, and many solutions can be found, a solution which satisfies the inequality for all $\varepsilon \in [0,1]$ is astonishingly difficult to find. In fact the symbolic math software such as Matlab's symbolic math toolbox and Mathematica are not able to solve this problem. As a typical example of the equations and the inequality involved we solve a typical example. The data are as follows,

$$P = \begin{bmatrix} \frac{\pi}{4} & 0 & 0 \end{bmatrix}$$

$$Q = \begin{bmatrix} \frac{\pi}{4} & 2 & 1.4\cdot\pi \end{bmatrix}$$

$$v_1 = m_{10} + m_{11}\cdot\varepsilon + m_{12}\cdot\varepsilon^2 + m_{13}\cdot\varepsilon^3$$
$$v_2 = m_{20} + m_{21}\cdot\varepsilon + m_{22}\cdot\varepsilon^2 + m_{23}\cdot\varepsilon^3$$

$$l_{[Y_1,Y_2]} = 0.1\cdot(\varepsilon - 0.5)$$

Then the equations and inequality become (here the symbol *t* is used instead of $\varepsilon$),

```
m10+1/2*m11+1/3*m12+1/4*m13-8184110467507909/9007199254740992 = 0
```

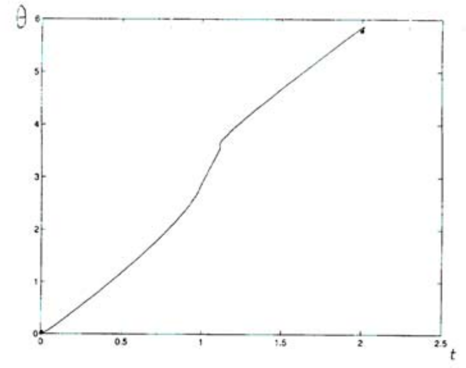
```
m20+1/2*m21+1/3*m22+1/4*m23+6283/4000 = 0
```

```
1/32*m23*m13+1/28*m22*m13+1/21*m23*m12+1/24*m21*m13+1/18*m22*m12+1/12*m23*m11+1/
20*m20*m13+1/15*m21*m12+1/10*m22*m11+1/5*m23*m10+1/12*m20*m12+1/8*m21*m11+1/4*m
22*m10+1/6*m20*m11+1/3*m21*m10+1/2*m20*m10+5477995636129085/4503599627370496 = 0
```

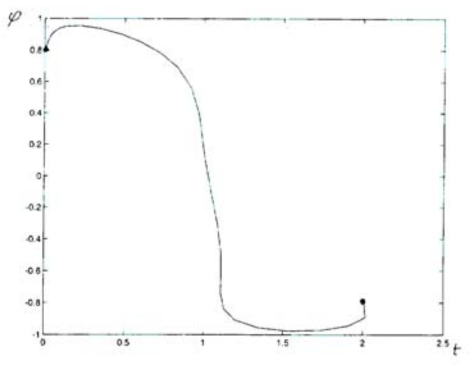
```
3/8*m20*m21*m10+4/63*m12*m20*m22+5/24*m10*m20*m23+1/9*m11*m20*m22+3/64*m11*m21*
m23+3/28*m10*m21*m23+5/144*m12*m21*m22+5/144*m20*m13*m23+5/216*m21*m13*m22+1/36
*m21*m12*m23+5/84*m11*m21*m22+3/160*m21*m13*m23+7/528*m22*m23*m13+7/360*m22*m2
3*m12+7/216*m22*m23*m11+5/56*m11*m20*m23+7/96*m22*m23*m10+4/15*m10*m20*m22+3/56
*m20*m21*m13+5/36*m10*m21*m22+1/12*m20*m21*m12+1/80*m23^2*m11+1/132*m23^2*m12+1
/42*m12*m21^2+1/48*m11*m22^2+1/64*m13*m21^2+1/192*m23^2*m13+3/20*m20*m21*m11+1/24
*m20*m13*m22+1/24*m20^2*m13+1/36*m23^2*m10+1/24*m11*m21^2+1/21*m10*m22^2+1/120*m
13*m22^2-5637776814438045/9007199254740992+1/15*m20^2*m12+1/10*m10*m21^2+5/96*m12*m
20*m23+1/81*m12*m22^2+1/8*m20^2*m11+1/3*m20^2*m10 = 0
```

```
14400/1681*(m20*t+1/2*m21*t^2+1/3*m22*t^3+1/4*m23*t^4+161/205)/sin(m20*t+1/2*m21*t^2+1/3*
m22*t^3+1/4*m23*t^4+161/205)*(4021/4920-39/82*cos(m20*t+1/2*m21*t^2+1/3*m22*t^3+1/4*
m23*t^4+161/205))^2*(m10+m11*t+m12*t^2+m13*t^3)+14400/1681*(m20*t+1/2*m21*t^2+1/3*m22*
t^3+1/4*m23*t^4+161/205)/sin(m20*t+1/2*m21*t^2+1/3*m22*t^3+1/4*m23*t^4+161/205)*(m10*t+1/
2*m11*t^2+1/3*m12*t^3+1/4*m13*t^4)*(4021/4920-39/82*cos(m20*t+1/2*m21*t^2+1/3*m22*t^3+1/
4*m23*t^4+161/205))^2*(1/(m20*t+1/2*m21*t^2+1/3*m22*t^3+1/4*m23*t^4+161/205)-(-39/41+cos(
m20*t+1/2*m21*t^2+1/3*m22*t^3+1/4*m23*t^4+161/205)*(4021/4920+39/82*cos(m20*t+1/2*m21*t^
2+1/3*m22*t^3+1/4*m23*t^4+161/205)))/sin(m20*t+1/2*m21*t^2+1/3*m22*t^3+1/4*m23*t^4+161/20
5)/(4021/4920-39/82*cos(m20*t+1/2*m21*t^2+1/3*m22*t^3+1/4*m23*t^4+161/205)))*(m20+m21*t+
m22*t^2+m23*t^3) > 0
```

In the following page the result of a typical optimization made by hand is included. Because the inequality is not satisfied for all t, we see that we have to go back in time. In fact the trajectories generated in this way are *orbit equivalent* to the trajectories of the acrobot but are not *trajectory equivalent*.



θ
0
1
2
3
4
5
6
0
0.5
1
1.5
2
2.5
t



φ
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.5
1
1.5
2
2.5
t

# Chapter 5

# Conclusion

In this dissertation I studied the problem of motion planning for a two link free flying robot, the acrobat, from the differential geometric point of view. I applied two different methods to this problem, the path deformation and the nilpotentization. As the results show, although nilpotentization, is more rigorous and mathematically rich than the path deformation, but since its theory is not very well known yet, it is not very appropriate from the practical point of view. Nevertheless if it is solved the nilpotentized system can be easily integrated, so it gives a very straightforward solution to the problem. This is not the case with systems having drift, as our case, where one finally arrives at an equality, which cannot be solved easily at least until now. In future more powerful symbolic math packages may be able to deal with these problems. On the other hand the path deformation is a good method for solving these problems, even when the system contains drift. The open loop controls generated by this method can be set to satisfy different conditions such as continuity so they are very appropriate for practical applications.

My prospect is that more work is on the way for the theory of Nilpotentization, while many practical problems might be solved using the path deformation method.

## Bibliography

[1] Bellaiche A., Laumond J.P., Chyba M.: Canonical nilpotent approximation of control systems: application to nonholonomic motion planning, *32nd IEEE Conf. on Decision and Control*: pp. 2694-2699, 1993.

[2] De Luca A., Mattone R., Oriolo G.: Stabilization of an underactuated planar 2R manipulator, (?), (1999).

[3] Godhavn J.M., Balluchi A., Crawford L.S., Sastry S.S: Control of nonholonomic systems with drift terms, *Automatica*, vol. 35: pp. 837-847, 1999.

[4] Hermes H.: Distributions having bases which generate finite dimensional Lie algebras, *Systems and Control Letters* 8: pp. 375- 380, North Holland, (1987).

[5] Hermes H., Lundell L., Sullivan D.: Nilpotent bases for distributions and control systems*, J. Diff. Eqs.*, vol. 55 no.3: pp. 385-400, 1984.

[6] Hermes H.: Distributions and the Lie algebras their bases can generate, In *Proc. Amer. Math. Soc.*, vol.106 no.2: pp. 555-565, 1989.

[7] Hermes H.: Nilpotent and high-order approximations of vector field systems, *SIAM Review*, vol. 33 no.2: pp. 238-264, 1991.

[8] Isidori A., *Nonlinear control systems 3rd edition*, Springer, 1995.

[9] Lafferriere G.., Sussmann H.J.: A differential geometric approach to motion planning, In *Nonholonomic motion planning*, Z. Li and J.F. Canny Eds., Kluwer Academic Publishers, 1992.

[10] Lafferriere G.., Sussmann H.J.: Motion planning for controllable systems without drift: a preliminary report, *Rutgers University Systems and Control Center Report*, SYCON-90-04, 1990.

[11] Montgomery R., Zhitomirskii M.: Geometric approach to Goursat flags, available

at http://orca.ucsc.edu/~rmont , (1999).

[12] Mormul P.: Geometry of Goursat flags and their singularities of codimension 2, (?), 2001.

[13] Mormul P.: Goursat flags: Classification of codimension-one singularities, *J. Dyn. Ctrl. Sys.*, vol. 6 no.3: pp. 311-330, 2000.

[14] Munthe-Kaas H., Owren B.: Computations in a free Lie algebra, (?), 1998.

[15] Mita Ts., *Introduction to Nonlinear Control-Skill Control of Robots*, Translated from Japanese, 2001.

[16] Murray R.M., Sastry S.S.: Nonholonomic motion planning steering using sinesoids, *IEEE Transactions on Automatic Control*, vol. 38 no.5: pp. 700-716, 1993.

[17] Murray R.M.: Nilpotent bases for a class of non-integrable distributions with applications to trajectory generation for nonholonomic systems*, Math. Contr. Sign. Sys.* 7: pp. 58-75, 1994.

[18] Struemper H., Krishnaprasad P.S.: Nilpotentization and motion control for under-actuated systems on matrix Lie groups, (?), (1999).

[19] Streumper H.: Motion control for nonholonomic systems on matrix Lie groups, *Ph.D. thesis*, 1998.

[20] Sussmann H.: New differential geometric methods in nonholonomic path finding, In *Systems, models and feedback: theory and applications*, A. Isidori and T.J. Tarn Eds., Birkhaüser, 1992.

[21] Sussmann H.: Local controllability and motion planning for some classes of systems with drift, (?), (1991).

[22] Sussmann H.: A product expansion for Chen series, In *Theory and Applications of Nonlinear Control Systems*, C.I. Byrnes, and E. Lindquist Eds., North-Holland, 1986, pp.

323-335.

[23] Varadarajan V.S., *Lie groups, Lie algebras and their representations*, Springer-Verlag, 1984.