Control of a Rotary Inverted Pendulum using Symbolic Methods

Scientific work to obtain the degree
B.Sc. Engineering Science

Munich School of Engineering
Technische Universität München

Supervised by Prof.Dr.Majid.Zamani
Mr.Pushpak Jagtap,Mr.Mahmoud Khaled.
Assistant Professorship of Hybrid Control Systems

Submitted by Feki Mohamed Amine
Matriculation Number: 03677907

Filed on 28.01.2019 in München

Erklärung

Ich versichere hiermit, dass ich die von mir eingereichte Abschlussarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ort, Datum, Unterschrift

Acknowledgments

First, I would like to thank Prof.Zamani for offering me the chance to write my thesis under the supervision of his chair, and offering me a project that combines both theoretical and practical interesting aspects.

I am very grateful for my Pushpak Jagtap, for his clear explanations, for the detailed answers to my questions and for his assistance during some technical issues. I would also like to thank Mahmoud Khaled, who has assisted me in the final stages of this project, and provided me with great advice.

I should not forget to thank my family, particularly my parents, whose continuous support was not only of a financial order, but also moral during difficult phases of my studies. Without their contributions, I would never have made it so far.

Abstract

Symbolic methods are among the emerging control techniques that have been drawing the attention in the control theory research community. Their main advantage resides in enabling the synthesis of a correct-by-construction controller that enforce complex specifications on a system. The classes of systems that mathematically admit a symbolic abstraction has been consistently growing, including nonlinear systems without stability assumption.

In this thesis, we are interested in testing symbolic methods on a non-linear, unstable system. The choice went to the rotary inverted pendulum, for which we designed an abstract controller that ensures an invariance, also called safety, specification. Experiments were conducted on the real plant to test the performance of the refined controller.

Introduction

Control theory is a powerful branch of mathematics, devoted to the study of dynamical systems with inputs, in order to derive adequate control strategies to obtain the desired behavior of a system.

Control problems are omnipresent in an uncountable number of engineering applications and science fields. There has been a strong correlation between the industrial development and the rise of control theory.

While control strategies for linear-time-invariant systems are mature and well-established, those related to non-linear systems, i.e. systems which do not obey the superposition principle, are still a current center of research effort.

Since most real-world application, for example, applications within the field of robotics or aerospace, exhibits a non-linear response, non-linear control holds great importance to the industry.

Some techniques, such as feedback linearization, could be sometimes deployed to a certain class of non-linear system, but they still lack generality. The need to resort to exhaustive numerical simulations of the plant is still indispensable to design a controller for many complex nonlinear systems, for instance, an airplane.

In this thesis, we investigate an innovative control method, called the symbolic method, which possess a great potential to become the future solution for many advanced control problems, including the nonlinear ones.

For the practical demonstration of the performance and efficiency of this method, we choose to work with a rotary inverted pendulum. This choice is well-founded, first, the dynamics of the pendulum is complex, rich, highly nonlinear and unstable. Thus this pendulum is representative of many real-world applications, for instance robots or drones. Second, the inverted pendulums are the gold standard in control theory community to validate emerging control strategies and test their performances, we wished to keep this tradition.

The objectives of this thesis are to derive a dynamical model for the pendulum, choose the proper abstraction method for the implementation, and synthesize an abstract controller to enforce an invariance, alternatively called safety, specification.

We finally verify the correctness of the approach by refining the abstract the controller to the real plant and conducting some experiments.

Outline:

The plan followed in this thesis is similar to the order of steps we carried on in practice. The first two chapters deal about some generalities and serves to further motivate this thesis. The intermediate ones are concerned with important aspects of the theory. The last three chapters are devoted to the implementation and experiments.

Chapter 1 introduces the concept of inverted pendulums, their main properties and the position they hold in both control theory education and research, as well as their relevance for the real-world applications.

Chapter 2 deals with symbolic methods in a general manner, the idea behind them, and the main advantages they could offer. We also briefly introduce the concept of linear temporal logic (LTL) used to define complex specifications.

Chapter 3 is concerned with the precise derivation of a mathematical model that describes the rotary inverted pendulum. We derive the mechanical equations with the Lagrangian formulation. We set all the equations in the useful state-space representation form and we define the electromechanical coupling between the pendulum and the Servo-base.

In **Chapter 4** we discuss the theoretical aspects of symbolic methods for the particular case of nonlinear systems without stability assumptions. The main focus lies on the mathematical construction of an abstraction of the real system. We present the two possible alternatives found in the literature, the first being based on the approximate alternating simulation relation, the second being founded on the feedback refinement relation. The purpose of introducing the two methods is to compare them from a practical point of view, we justify why the feedback refinement relation is better suited to implement a controller to the pendulum.

In **chapter 5** we introduce the implementation tool SCOTS. we illustrate the computational work-flow in this software. We define which parameters and entries are user-dependent in order to automatically synthesize the controller. We also carry on the basic implementation steps, for instance, the computation of growth bound.

We dedicate **chapter 6** to describe the main difficulties we encountered during the implementation in relation to the choice of design parameters and the curse of dimensionality. We describe the cause of those difficulties and the approaches we tried out. We justify why we had to consider a reduction of the dimension of the dynamics, which was possible and leads to find a controller.

In **chapter 7** we simulate the closed-loop behavior of the controller on Matlab. Then, we refine the abstract controller to the real plant that we have assembled. we verify the correctness of all the previous step we carried on and the performance of the implemented controller.

Chapter 1: Generalities about inverted pendulums.

1)Definition of an inverted pendulum:

the main distinction between a simple pendulum and an inverted pendulum is the position of the center of mass with respect to the pivot. Inverted pendulums are characterized through a center of mass located above the pivot. As a consequence, they are inherently unstable. This means that an inverted pendulum will fall down from the upwards position in the absence of a control signal. On the other hand, the simple pendulum is stable.

2)Types of inverted pendulums:

There are many variations of the inverted pendulum. Most classical examples include the cart inverted pendulum [fig 1.1], inertia wheel inverted pendulum, and rotary inverted pendulum that we will test in this thesis.

There exist theoretically infinitely many geometrical configurations of inverted pendulums, which could be generated by adding an extra number of links and joint. The typical example is the double rotary inverted pendulum [fig 1.2]

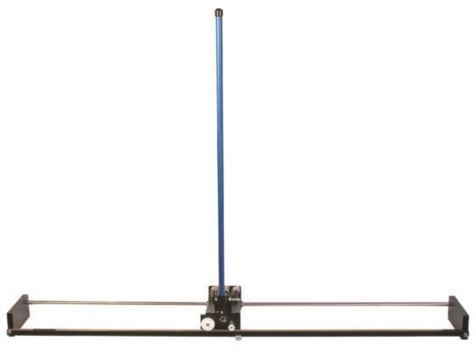


Figure 1.1: inverted pendulum on a cart

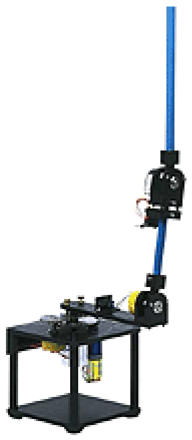


Figure 1.2 rotary double inverted pendulum

3) inverted pendulums and control theory:

3.1) an educational tool in control theory:

Inverted pendulums have been a classical, yet valuable example, to introduce students to some basic concepts of control theory in universities. Furthermore, many labs have been designed with the purpose of demonstrating this theory. In general, basic labs introduce simple control strategies such as state-feedback control and pole placement.

3.2) Interesting properties of inverted pendulums:

inverted pendulums exhibit a certain number of properties, that justify why they are relevant to control theory and robotics research.

- Non-linear: in contrast to their simple geometrical configuration, inverted pendulums exhibits a complex, rich, highly nonlinear dynamics. This nonlinearity is explained physically by the action or inertial forces that act on the system, such as centrifugal and Coriolis forces. As a consequence, they are ideal research systems in the field of non-linear control theory.
- Under-actuated: this means that the number of their degree of freedom is higher than the number of control actuators (generally there is a unique control input). this makes the control task harder, particularly for pendulums with multiple links. Under-actuated systems are interesting in robotics since we want to use the minimal number of actuators to reduce cost, as well as the system energy consumption and weight.
- Potentially-perturbed: it is possible to add some stochastic parameters to the dynamics, for example, a perturbation torque. therefore, inverted pendulums could be studied in the field of stochastic control theory.

Another interesting point is that we have the possibility to extend further the dimension of inverted pendulum state-space, by assembling further link and joint, and use this fact to study high dimensional systems.

3.3) A benchmark for verification and performance analysis:

Inverted pendulums are the gold standard for many control theorists to test and validate their algorithms and control strategies.it is also used to test the performance and effectiveness of unconventional and new control strategies. Because it is hard to make a full account of all the control strategies that have been tented on inverted pendulum, we only mention, for illustration purpose some of them, notably strategies that have emerged from the artificial intelligence field: Fuzzy logic control (Roose, SamerYahya, & Al-Rizzo, 2017) (Mladenov, Tsenov, Ekonomou, & Harkiolakis, 2009), neuronal networks control (Mladenov, Tsenov, Ekonomou, & Harkiolakis, 2009) and genetic algorithms (Metni, 2009) .

4)inverted pendulums and real-world applications:

Inverted pendulums are not only a mere theoretical tool for education and research, but also an inspiration source to many industrial applications, particularly those which shares similar dynamics with them. (Boubaker, 2014)

Possibilities of applications field includes:

Robotics: the examples are numerous, but the most interesting case is the one when we are wishing to stabilize a humanoid robot. From a dynamical point of view, an inverted pendulum and a humanoid robot have the similar instability property and a similar form of the reigning differential equations. Thus the task of stabilizing a robot could be cast to the pendulum case. [fig 1.3]

Aerospace: the problem of stabilizing a rocket at the launching moment is similar to stabilizing a pendulum.

Transport: an example of a commercial application Is the Segway, which is a mobile wheeled inverted pendulum. [fig 1.4]

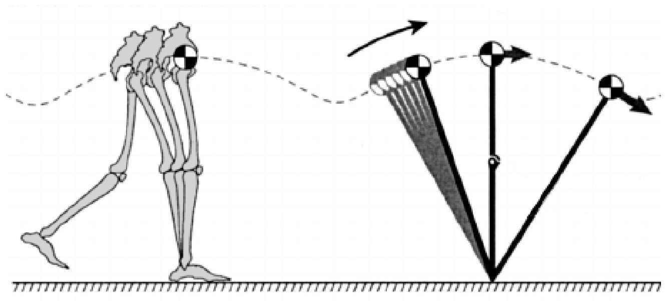


Figure 1.3: inverted pendulum as a model for a walking robot.

Figure 1.4: Segway an example of application inspired by inverted pendulums

Chapter 2: Generalities about symbolic methods

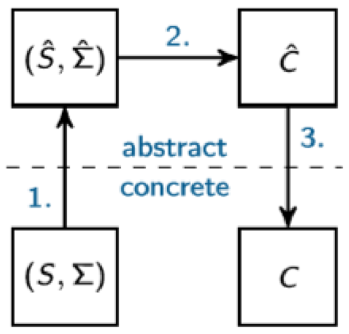
1)the main idea of symbolic methods:

Independently of which kind of systems we are dealing with, a symbolic method for control consists schematically of the following [fig 2.1]:

Step 1: from the original control problem, composed of a system and a specification, we build an abstracted control problem. Abstraction could be understood as an equivalent mathematical description of the control problem, that is ideally more simple than the original one and offer much easier manipulation. In case of a space-continuous system, that has infinite and uncountable states, an abstraction of that system could be another system that has finite and countable states. Construction of abstractions is not easy from a mathematical point of view and will depend on the nature of that system. Intuitively, we are interested whether it is possible to exploit some convenient dynamical properties of the system so that it would be possible to represent some aggregates or collections of infinite and continuous states in one unique discrete state, called symbol.

Step2: we solve an auxiliary control problem by synthesizing an abstract controller that enforces the abstract specification on the abstract system. This process is generally automated in practical implementations, and uses the advantage of the finite nature of the abstraction, to deploy techniques and algorithms that have been priory developed by computer-scientists and automata theorist for finite state machines. Finite state machines are very useful to model many phenomena and applications such as digital circuits. therefore, they have been extensively researched in the last century and a large number of algorithms has been developed, which symbolic methods could exploit at this level.

Step3: Once an abstract controller is successfully obtained, the controller is refined into the original system, also called the plant. The refinement process will generally depend on the mathematical procedure used to construct the abstraction.



$$\begin{tikzcd}
\boxed{(\hat{S},\hat{\Sigma})} \arrow[r, "2."] & \boxed{\hat{C}} \arrow[d, "3."] \\
\boxed{(S,\Sigma)} \arrow[u, "1."] & \boxed{C}
\end{tikzcd}$$

Figure 2.1: general schemas of a symbolic method and the relation between the abstract and concrete domain

2)LTL specifications:

By specification we mean the behavior that we want the system to exhibit, or the behavior it should not exhibit. There exist many different manners to express a specification. A simple kind of specification is to force all the trajectories of a system to converge to an equilibrium point (stability specification) or to follow a reference trajectory (reference-tracking specification).

A more interesting, yet more complex, kind of specifications are those expressed in the form of linear-temporal-logic(LTL). LTL has been largely used in computer-science and software-engineering.

A basic example of LTL includes reachability specifications. For an atomic proposition which describes a system-state or a collection of system-states, basic LTL specifications include reachability denoted as $\lozenge\phi$, and invariance denoted as $\square\phi$.

Furthermore, it is possible to combine basic specifications with each other, and with the help of logical conjunctions ($\neg, \land, \lor$) to form more complex specifications. For instance:

- $\neg \lozenge\phi$: avoid specification.
- $\square \lozenge\phi$: reach and stay.

Thus, LTL could be used to precisely formulate specifications for complex systems, for instance, a system that controls the autonomous-driving of a car, and ensures avoiding collisions, or to plan a complex motion of a robot in an environment with obstacles. [fig 2.2]

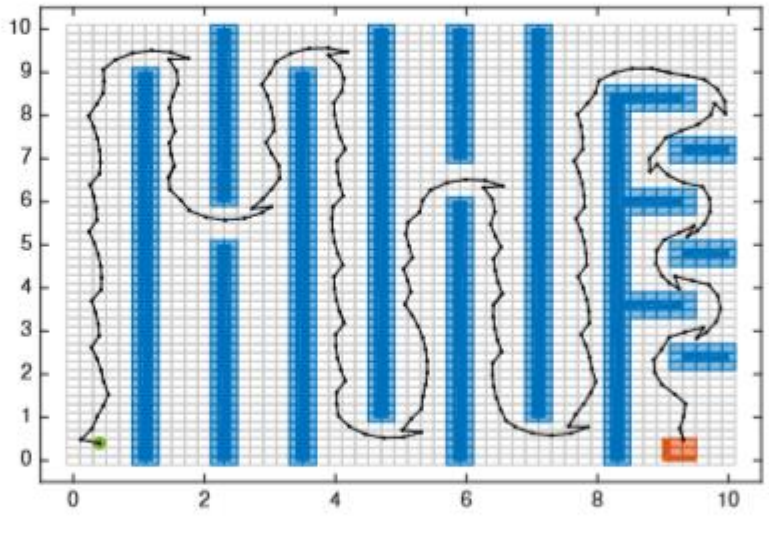


Figure 2.2: an example of reach and avoid specification, the robot(green) should reach the red region while avoiding the obstacles(blue)

.

3)Advantages of Symbolic methods:

3.1) enforcing complex specifications in LTL:

While classical control techniques are very well suited to enforce simple specifications such as reference tracking and stabilizing (PID controller), symbolic methods are naturally able to support highly complex specifications expressed in LTL. This advantage is inherent to the fact that symbolic methods are based on abstracting the system as a finite state machine, for which a well-developed theory by computer scientists and game-automata theorist already exists, and from which we could benefit to try to enforce those complex specifications.

3.2) a correct-by-construction controller:

Control problems are in the heart of many engineer's tasks. The engineer is faced with the task to design a controller that enforces certain specifications or constraints on a system, which could be continuous, discrete or combination of both (hybrid).
The difficulty to find a solution will depend both on the complexity of system dynamics and the complexity of the specifications.
In order to guarantee the synthesis of a controller that reliably conform to the specification and work properly in safety-critical environments (example: a surgical robot), the engineer should resort to formal verification techniques.
There exist two approaches, in the first one, the verification phase is executed after the modeling and the design phase of the controller. If the controller fails however to succeed the verifications, an iterative process should take place, the engineer should identify the problem and restart the modeling and design phase. The second approach consists of merging the design and verification phases in a single process, this is known as the correct-by-construction approach.
In the context of cyber-physical systems, where the applications tend to be more sophisticated, with intricate dynamics and complex interactions between software and hardware components resulting in the need of highly complex specifications, resorting to the design-then-verification approach seems not appealing, since it will result in a very long iterative process and as a consequence higher design costs. On the other hand, a correct-by-construction approach is ideally suited for this task.
Symbolic methods are among the methods that guarantee a correct-by construction controller, resulting in a smoother design process and saving tremendous costs.

3.3) applicability to a large class of systems:

Although symbolic methods are relatively recent concept, the class of systems admitting a symbolic method has been consistently growing, including linear systems, affine systems, affine switched systems, incrementally input-to-state-stable nonlinear system, non-linear systems without stability assumptions, stochastic systems to name few.
This applicability of symbolic methods to many systems, in particular, those which frequently are modeled in real-life applications make them interesting from an industrial point of view.

3.4) robustness under uncertainties:

Symbolic methods could be adapted to tolerate uncertainties in the system, such as those due to perturbations or measurement errors (Reissig, Rungger, & Alexander, 2015). this is very relevant since models used in engineer practice are frequently prone to uncertainties.

4) Comments:

In this thesis, we deal only with deterministic non-linear, unstable systems, through the example of the rotary inverted pendulum, therefore we will not be able to verify all the above-mentioned advantages. We will only check the correctness-by-construction property and the enforcement of an LTL-specifications.

Possible LTL-specifications for a rotary inverted pendulum could be invariance (stabilize the pendulum upwards), reachability(swing-up) and reach and stay (swing-up then stabilize).it is not easy to imagine more complex specifications because of the simple geometry of the plant. We choose mainly to deal with invariance specification, the procedure is similar for the other ones but will need eventually higher computation and memory performance than what we could access to.

Chapter 3: Dynamics of the rotary inverted pendulum.

1) Overview:

In this part, we want to derive a model that mathematically describes the rotary inverted pendulum and its base [fig 3.1]. we preferred to make the least possible assumptions and simplifications while deriving the model, in order to reduce modeling uncertainties with respect to the real plant.

For mechanical systems, different techniques that yields the equation of motions exists, the most two well-known are Newton-Euler and Lagrange formulation. Although both will lead to the exact same result under identical assumptions, we preferred to follow the latter since it is more powerful when dealing with systems composed of links and joints, such as robot arm-manipulators, but also pendulums. Lagrange formulation could also be used by some software to automatically derive the equations of motion for the general case of a system with n link and joints, for example, a n-rotary inverted pendulum (Sarnovsky & Jadlovska, 2013).

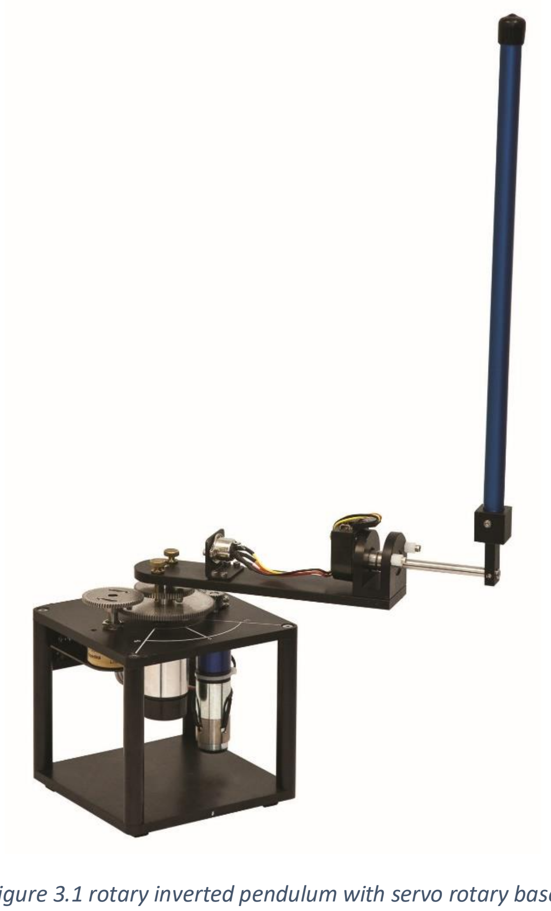


Figure 3.1 rotary inverted pendulum with servo rotary base

2) Lagrange formulation:

Given a system with n degree of rotation freedoms, composed of n joints and links, with no translation motion. $\mathrm{l_k}$ is the distance between the kth –joint and the center of mass of kth – link ,whereas $\mathrm{L_k}$ is the full link length. we denote $\mathrm{m_k}$ its mass, $\mathrm{r_k}$,$\mathrm{v_k}$,ω_k respectively the position, translation velocity and angular velocity vectors of its center of mass, $\mathrm{J_k}$ the corresponding inertia tensor,$\mathrm{b_k}$ the viscous-damping coefficient and $\boldsymbol{\tau}_\mathrm{k}$ the torque acting on it.[fig 3.2]

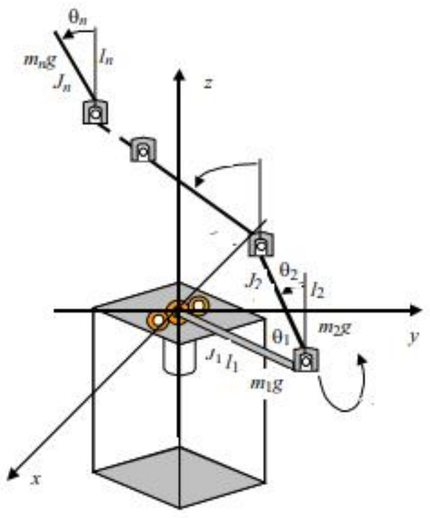


Figure 3.2 rotary n-inverted pendulum

The total kinetic energy, resulting from the motion of all the links is denoted as T.

$$\mathrm{T} = \sum_{\mathbf{k=1}}^{\mathbf{n}} \left(\tfrac{1}{2}\mathbf{m_k v_k^t v_k} + \tfrac{1}{2}\boldsymbol{\omega}_\mathbf{k}^\mathbf{t}\mathbf{J_k}\boldsymbol{\omega}_\mathbf{k} \right) \quad (3.1)$$

The total potential energy V is given by:

$$\mathrm{V} = -\sum_{\mathbf{k=1}}^{\mathbf{n}} \mathbf{m_k r_k\, g} \qquad (3.2)$$

Where **g** *is the gravity acceleration vector.*

The Lagrangian, which expresses the total energy of the system and serves to derive the n set of differential equations, is defined as:

$$\boldsymbol{\mathcal{L}} = \mathrm{T} - \mathrm{V}$$

This derivation is achieved by solving n time the following equations (3.3):

$$\frac{\mathbf{d}}{\mathbf{dt}}\frac{\boldsymbol{\partial\mathcal{L}}}{\boldsymbol{\partial\dot{\theta}}_\mathbf{k}} - \frac{\boldsymbol{\partial\mathcal{L}}}{\boldsymbol{\partial\theta}_\mathbf{k}} = \boldsymbol{\tau}_\mathbf{k} - \mathbf{b_k}\dot{\boldsymbol{\theta}}_\mathbf{k}$$

With $\boldsymbol{\theta} = (\boldsymbol{\theta_1}\ \boldsymbol{\theta_2} \ldots.\boldsymbol{\theta_k} \ldots\ldots \boldsymbol{\theta_n})$ denotes the general coordinates vector, describing the rotation angles of the system. We can see how general this approach to derive equations for any high dimensional multiple links joint systems. Next, we only consider the case n=2 of our rotary simple inverted pendulum.

3) Kinematics:

Now we are interested in developing the expression of the Lagrangian as a function of the general coordinates. For the purpose, we have first to equip each link with a reference framework (x_n , y_k , z_k) for k=1,2.we also define the reference (x_0 , y_0 , z_0) fixed to the base. The direction of references is defined according to the Denavit-Hartenberg convention.

Since we need to relate different references to each other, we have to define the change of coordinates matrices, which are in this case rotations matrices R_{10} and R_{21}.

$$R_{10}=\begin{pmatrix} \cos(\boldsymbol{\theta_1}) & \sin(\boldsymbol{\theta_1}) & 0 \\ -\sin(\boldsymbol{\theta_1}) & \cos(\boldsymbol{\theta_1}) & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad R_{21}=\begin{pmatrix} 0 & \sin(\theta_2) & -\cos(\boldsymbol{\theta_2}) \\ 0 & \cos(\boldsymbol{\theta_2}) & \sin(\boldsymbol{\theta_2}) \\ 1 & 0 & 0 \end{pmatrix}$$

we obtain the following relations between the references:

$$(x_0 , y_0 , z_0) \xrightarrow{R_{10}} (x_1 , y_1 , z_1) \xrightarrow{R_{21}} (x_2 , y_2 , z_2)$$

Now we want the expression of $v_1, v_2, \omega_1, \omega_2$.

$\omega_1 = (0 \quad 0 \quad \dot{\theta}_2)^t$

$\omega_2 = (0 \quad 0 \quad \dot{\theta}_2)^t + R_{21}\omega_1 = (-\cos(\theta_2)\dot{\theta}_1 \quad \sin(\theta_2)\dot{\theta}_1 \quad \dot{\theta}_2)^t$

$v_1 = \omega_1 \times (l_1 \quad 0 \quad 0)^t = (0 \quad l_1\dot{\theta}_1 \quad 0)^t$

$v_2 = \omega_2 \times (l_2 \quad 0 \quad 0)^t + R_{21}(\omega_1 \times (L_1 \quad 0 \quad 0)^t) = (L_1 sin(\theta_2)\dot{\theta}_1 \quad L_1\cos(\theta_2)\dot{\theta}_1 + l_2\,\dot{\theta}_2 \quad -\dot{\theta}_1 l_2\, sin(\theta_2))^t$

The next tables summarize the kinematics expressions needed to derive the equations of motion

Table 1 kinematics expressions

velocities	Expression
ω_1	$(0 \quad 0 \quad \dot{\theta}_1)^t$
ω_2	$(-\cos(\theta_2)\dot{\theta}_1 \quad \sin(\theta_2)\dot{\theta}_1 \quad \dot{\theta}_2)^t$
v_1	$(0 \quad l_1\dot{\theta}_1 \quad 0)^t$
v_2	$(L_1 sin(\theta_2)\dot{\theta}_1 \quad L_1\cos(\theta_2)\dot{\theta}_1 + l_2\,\dot{\theta}_2 \quad -\dot{\theta}_1 l_2\, sin(\theta_2))^t$

4)mechanical equations of motions:

4.1) inertial tensor assumptions:

since the reference axis coincides with the principle axis of each link, the inertial tensor $\mathbf{J_k}$ has a diagonal form:

$$\mathbf{J_k} = \begin{pmatrix} \mathrm{J_{kxx}} & 0 & 0 \\ 0 & \mathrm{J_{kyy}} & 0 \\ 0 & 0 & \mathrm{J_{kzz}} \end{pmatrix}$$

The only simplifications we took in the whole derivation is to consider that $\mathrm{J_{kxx}}$=0 , which is physically a reasonable approximation, furthermore by a symmetry argument $\mathbf{J_k}$ could be written as:

$$\mathbf{J_k} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \mathrm{J_k} & 0 \\ 0 & 0 & \mathrm{J_k} \end{pmatrix}$$

4.2) coupled equations of motion:

Using equation (3.1) and (3.2) we obtain:

$\mathrm{T}=\frac{1}{2}\dot{\theta}_1{}^2(\mathrm{m_1 l_1{}^2}+\mathrm{J_1} + \mathrm{m_2 L_2{}^2} + \sin(\theta_2)^2\left(\mathrm{m_2 l_2{}^2} + \mathrm{J_2}\right)) + \frac{1}{2}\dot{\theta}_2{}^2\left(\mathrm{m_2 l_2{}^2} + \mathrm{J_2}\right) + \mathrm{m_2 l_2 L_1}\cos(\theta_2)\,\dot{\theta}_1\dot{\theta}_2$

$\mathrm{V}=\mathrm{g m_2 l_2}(1 - \cos(\theta_2))$

A careful employ of (3.3) will result in the following expressions:

$\ddot{\theta}_1(\mathrm{J_1} + \mathrm{m_1 l_1{}^2} + \mathrm{m_2 L_2{}^2} + \sin(\theta_2)^2\left(\mathrm{m_2 l_2{}^2} + \mathrm{J_2}\right))+ \ddot{\theta}_2\mathrm{m_2 l_2 L_1}\cos(\theta_2) - \mathrm{m_2 l_2 L_1}\sin(\theta_2)\,\dot{\theta}_2{}^2 + \sin(2\theta_2)\,\dot{\theta}_1\dot{\theta}_2\left(\mathrm{m_2 l_2{}^2} + \mathrm{J_2}\right)= \tau_1\text{-}\mathrm{b_1}\dot{\theta}_1$ (3.4)

$\ddot{\theta}_1\mathrm{m_2 l_2 L_1}\cos(\theta_2) + \ddot{\theta}_2\left(\mathrm{m_2 l_2{}^2} + \mathrm{J_2}\right) - \frac{1}{2}\left(\mathrm{m_2 l_2{}^2} + \mathrm{J_2}\right)\sin(2\theta_2)\,\dot{\theta}_1{}^2+\mathrm{g m_2 l_2}\sin(\theta_2)=\tau_2\text{-}\mathrm{b_2}\dot{\theta}_2$ (3.5)

We are interested in having the most compact form of those equations to ease the implementations later, we could obtain a slightly better expression by introducing the following variables, which could be interpreted as the inertia terms with respect to the pivots (Steiner's theorem)

$$\hat{J}_1 = \mathrm{J_1} + \mathrm{m_1 l_1{}^2}$$

$$\hat{J}_2 = \mathrm{J_2} + \mathrm{m_2 l_2{}^2}$$

We further define:

$$\hat{J}_0 = \hat{J}_1 + \mathrm{m_2 L_2{}^2}$$

$$\mathrm{c} = \mathrm{m_2 l_2 L_1}$$

equations (3.4) and (3.5) could be written as a shorter form as:

$$\boxed{\begin{gathered} \ddot{\theta}_1(\hat{J}_0 + \sin(\theta_2)^2\hat{J}_2)+\mathrm{c}\cdot\cos(\theta_2)\ddot{\theta}_2 - \mathrm{c}\cdot\sin(\theta_2)\,\dot{\theta}_2{}^2 + \hat{J}_2\sin(2\theta_2)\,\dot{\theta}_1\dot{\theta}_2= \tau_1\text{-}\mathrm{b_1}\dot{\theta}_1 \\ \ddot{\theta}_1\mathrm{c}\cdot\cos(\theta_2) + \ddot{\theta}_2\hat{J}_2 - \frac{1}{2}\hat{J}_2\sin(2\theta_2)\,\dot{\theta}_1{}^2+\mathrm{g m_2 l_2}\sin(\theta_2)=\tau_2\text{-}\mathrm{b_2}\dot{\theta}_2 \end{gathered}}$$

4.3) Matrix form:

it is possible to write the above equations in a matrix form, this form has two advantages, it enables us to better understand the dynamics and interpret how the non-linearity appears, second, it will help us to decouple the equations to obtain the needed state-space representation form.

we note $\tau = (\tau_1 \quad \tau_2)$, $\theta = (\theta_1 \quad \theta_2)$, $\dot{\theta} = (\dot{\theta}_1 \quad \dot{\theta}_2)$, $\ddot{\theta} = (\ddot{\theta}_1 \quad \ddot{\theta}_2)$, and introduces the matrix and vector functions:

$$M(\theta,\dot{\theta})=\begin{pmatrix} \hat{J}_0 + \sin(\theta_2)^2 \hat{J}_2 & c \cdot \cos(\theta_2) \\ c \cdot \cos(\theta_2) & \hat{J}_2 \end{pmatrix}$$

$$C(\theta,\dot{\theta})=\begin{pmatrix} b_1 + \hat{J}_2 \sin(2\theta_2)\, \dot{\theta}_2 & -c \cdot \sin(\theta_2)\, \dot{\theta}_2{}^2 \\ -\frac{1}{2}\hat{J}_2 \sin(2\theta_2)\, \dot{\theta}_1 & b_2 \end{pmatrix}$$

$$B(\theta) = (0 \quad g m_2 l_2 \sin(\theta_2))$$

An elegant representation of equations of motions in an oscillator form is:

$$M(\theta,\dot{\theta})\ddot{\theta} + C(\theta,\dot{\theta})\, \dot{\theta} + B(\theta) = \tau$$

This equation has a physical interpretation and accounts for the sources of nonlinearities in the equations. $M(\theta,\dot{\theta})$ is a symmetrical matrix called the mass matrix and accounts for the inertial terms, $C(\theta,\dot{\theta})$ is the matrix that describes frictions as well as Coriolis and centrifugal forces, the $B(\theta)$ summarize the effect of potential and gravitational forces.

4.4) Uncoupled equations of motion:

the equation that has been derived so far are still in a coupled form, but since we will later need a state-space representation of the equations, we need to uncouple them.

We could do that with Symbolic Math Toolbox™ but a more elegant approach is to use the matrix form, which will result in less lengthy expressions.

We have $\det(M(\theta,\dot{\theta})) = \hat{J}_0\hat{J}_2 + \hat{J}_2{}^2 \sin(\theta_2)^2 - c^2 \cos(\theta_2)^2$

We assume it will never be zero, then $M(\theta,\dot{\theta})$ is invertible.

$$\ddot{\theta} = M^{-1}(\theta,\dot{\theta})(\tau - C(\theta,\dot{\theta})\dot{\theta} - B(\theta))$$

After some computations, we arrive to the separated form:

$$\ddot{\theta}_1 = \frac{-\widehat{J_2} b_1 \dot{\theta}_1 + c\cos(\theta_2)\, b_2 \dot{\theta}_2 - \widehat{J_2^2}\sin(2\theta_2)\dot{\theta}_1\dot{\theta}_2 - (1/2)\widehat{J_2} c \widehat{\cos(\theta_2)\sin}(2\theta_2)\dot{\theta}_1^2}{\hat{J}_0\hat{J}_2 + \hat{J}_2{}^2 \sin(\theta_2)^2 - c^2\cos(\theta_2)^2} +$$

$$\frac{\widehat{J_2} c \sin(\theta_2)\dot{\theta}_2^2 + \widehat{J_2}\tau_1 - c\cos(\theta_2)\tau_2 + (1/2) m_2^2 l_2^2 L_1 \sin(2\theta_2) \mathrm{g}}{\hat{J}_0\hat{J}_2 + \hat{J}_2{}^2 \sin(\theta_2)^2 - c^2\cos(\theta_2)^2}$$

$$\ddot{\theta}_2 = \frac{c\cos(\theta_2) b_1 \dot{\theta}_1 - b_2(\widehat{J_0} + \widehat{J_2}\ \sin^2(\theta_2))\, \dot{\theta}_2 + c\,\widehat{J_2}\cos(\theta_2)\sin(2\theta_2)\dot{\theta}_1\dot{\theta}_2 - (1/2)\sin(2\theta_2)\left[\widehat{J_0}\widehat{J_2} + \hat{J}_2^2\sin^2(\theta_2)\right]\dot{\theta}_1^2}{\hat{J}_0\hat{J}_2 + \hat{J}_2{}^2 \sin(\theta_2)^2 - c^2\cos(\theta_2)^2}$$
$$+ \frac{(\widehat{J_0} + \widehat{J_2}\sin^2(\theta_2)\tau_2 - m_2 l_2 \sin(\theta_2)(\widehat{J_0} + \widehat{J_2}\sin^2(\theta_2))\mathrm{g} - (1/2)c^2\sin(2\theta_2)\dot{\theta}_2^2 - c\cos(\theta_2)\tau_1}{\hat{J}_0\hat{J}_2 + \hat{J}_2{}^2 \sin(\theta_2)^2 - c^2\cos(\theta_2)^2}$$

5) State-space representation:

From the last two equations it is straightforward to write all the system dynamics in a state-space representation form:

$$\dot{\xi}(t) = f(\xi(t), \boldsymbol{\tau}) \quad (3.8)$$

Where $\xi(t) = (\theta_1 \quad \theta_2 \quad \dot{\theta}_1 \quad \dot{\theta}_2)$ with $f\colon \mathbb{R}^4 \times \mathbb{R}^2 \to \mathbb{R}^4$.

Now, we need to discuss the inputs. In the case of the plant we are working on, there exist only one torque acting on the system, because the pendulum is underactuated, so we have either to consider $\tau_2 = 0$, or to model τ_2 as a perturbation torque, so we can treat the system as non-deterministic. Our choice here was to not bring any perturbation on the model we will work on, although symbolic methods could handle this.

Another point is related to τ_1.in the real plant we cannot control the system directly through a torque, but rather through a voltage or a current. The expression for τ_1 is given by the manufacturer (Quanser), in this case:

$$\tau_1 = \frac{\eta_g\, K_g\, \eta_m\, k_{\,t}(V_m - K_g\, k_{\,m}\, \dot{\theta}_1)}{R_m} \quad (3.9)$$

So that our concrete control input is V_m.

Injecting the equation (3.9) in (3.8) gives us a slightly different form, that we are going to use in SCOTS:

$$\dot{\xi}(t) = \tilde{f}(\xi(t), V_m) \qquad (3.9)$$

with $\tilde{f}\colon \mathbb{R}^4 \times \mathbb{R} \to \mathbb{R}^4$.

Chapter 4: Symbolic methods for deterministic non-linear systems without stability assumptions

1) Overview:

The theory behind Symbolic methods is very profound and is based on results from different disciplines, essentially control theory and computer science. Therefore, a full account of this theory is beyond the scope of this thesis. We will rather focus on aspects that could be helpful for the practical implementation and to solve our problem. In this context, we are dealing with a non-linear, unstable, control system. Thus, it will be convenient to only focus on Symbolic methods for non-linear systems, where no stability condition (like incremental input –to-state stability (δ-ISS)), is required. The greatest attention will be centered on the procedure constructing a finite-state abstraction from the original continuous system. Once an abstraction is correctly constructed, the rest of steps, for instance solving the control problem on the abstract domain, is standard, straightforward and fully automated in most Software. Refining the abstract controller to the real plant is also unproblematic, and is granted to work if the abstraction is correctly constructed.

To sum up, we will only concentrate on mathematically generating an abstraction, also called symbolic model, for very general non-linear, unperturbed systems.

Checking the literature, we found two approaches compatible with our situation. The first approach is based on Approximate(-alternating-) simulation relations, and requires only the very mild assumption of the incremental forward completeness δ-FC, which is very relaxed condition compared to δ-ISS, and is verified by many physical systems including the pendulum (Zamani, Pola, Mazo JR, & Tabuada, 2011). The second approach relies on a different type of relation, called feedback refinement relation, and is inspired by the idea of over-approximating the attainable sets (reachable sets) by computing a growth bound (Reissig, Rungger, & Alexander, 2015).

before we mention globally how those approaches work, we want to cite the foundation that allows us mathematically, not only to describe the same control problem in different manners but also to relate those different descriptions to each other.

2) A versatile mathematical description of control systems:

2.1) Definition of a transition system:

In order to construct mathematical relations between different descriptions of the same dynamical phenomena, we need a versatile mathematical description that could be used to uniformly treat both the finite-state and the infinite state cases. this is allowed by the definition of transition systems.

Definition 1: (Tabuada, 2009) A transition system S is a sextuple (X , X_0, U , $\rightarrow$, Y , H) consisting of:

- A set of finite or infinite state alphabets X;
- A set of initial states $\mathrm{X}_0 \subseteq \mathrm{X}$;
- A set of inputs U;
- A transition relation $\rightarrow \ \subseteq \mathrm{X} \times U \times \mathrm{X}$;
- A set of output Y;
- An output map $H : \mathrm{X} \rightarrow Y$;

this definition is very general, we could adopt a simpler one that is also applicable to many control problems, in which the outputs and internal states coincides and no initial states are considered.

Definition 2: A simple transition system S' is a triple (X , U , $\rightarrow$) consisting of::

- A set of infinite or finite states X;
- A set of inputs U;
- A transition relation $\rightarrow \ \subseteq \mathrm{X} \times U \times \mathrm{X}$;

Remark: the transitions could be alternatively described through a map valued function

$F : \mathrm{X} \times U \rightrightarrows \mathrm{X}$.

We can see that the definition of a transition system is intuitive in the case of the discrete, finite states systems, where they could be thought as finite state machines. A graphical representation is also possible, if we know the set of states, inputs and transitions [Figure 4.1]. More intriguing however, is how this definition applicable to continuous control system.

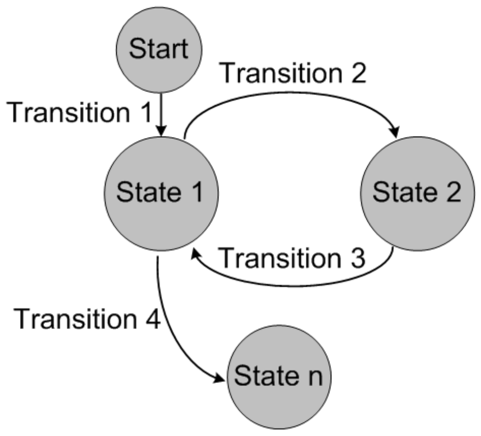


Figure 4.1: a possible graphical representation of finite-states transition system

2.2) Continuous control systems as transition systems:

2.2.1) Definition of a continuous control system:

Definition 3: (Tabuada, 2009) A continuous control system is a triplet $\Sigma = (\mathbb{R}^n, \mathcal{U}, f)$ consisting of:

- The state space $\mathbb{R}^n$.
- A set of input curve $\mathcal{U}$,whose elements are essentially bounded piecewise .
- A smooth continuous map $f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$

A continuously differentiable curve $\xi :]a, b[\rightarrow \mathbb{R}^n$ is said to be to be trajectory or solution of Σ if there is $u \epsilon \mathcal{U}$ so that $\frac{d}{dt}\xi = f(\xi, u)$ holds for almost every t ϵ $]a, b[$.

2.2.2) Interpretation of continuous control system as a transition system:

We could interpret a time-space continuous system as a transition system in the following way:

(Tabuada, 2009) Given a system $\Sigma = (\mathbb{R}^n, \mathcal{U}, f)$,it is possible to describe Σ by a transition system $S = (\mathrm{X}, \mathcal{U}, \rightarrow)$,where:

- $\mathrm{X} = \mathbb{R}^n$
- $\mathrm{U} = \mathcal{U}$
- $x \overset{v}{\rightarrow} x'$ if there exists $v \in \mathcal{U} : [0, \tau] \rightarrow \mathbb{R}^m$ and $\xi_{xv} : [0, \tau] \rightarrow \mathbb{R}^n$

  Satisfying $\frac{d}{dt}\xi_{xv} = f(\xi, v)$ with $\xi_{xv}(0) = x$ And $\xi_{xv}(\tau) = x'$,$\tau \in \mathbb{R}^+$.

Σ is therefore described by a transition system with infinite, uncountable state and input sets. In this context, transitions highlights the dynamical evolution of the system trajectories that starting at a given initial condition and evolve under some input signal and for some time horizon τ. A transition depends therefore on both input and time.

2.2.3) Interpretation of a Sampled system as a transition system:

a sampled control system is the time-discrete, space-continuous version of a time-continuous, space-continuous control system.

(Tabuada, 2009) For a fixed $\tau \in \mathbb{R}^+$, and a continuous control system $\Sigma = (\mathbb{R}^n, \mathcal{U}, f)$ we define the sampled transition system $S_\tau = (\mathrm{X}_\tau, \mathcal{U}_\tau, \underset{\tau}{\rightarrow})$ associated with Σ as:

- $\mathrm{X}_\tau = \mathbb{R}^n$
- $\mathcal{U}_\tau = \{\mathcal{X} \in C \mid dom\mathcal{X} = [0, \tau]\}$

- $x \overset{\mathcal{X}}{\rightarrow} x'$ if there exist $\mathcal{X} \in \mathcal{U}_\tau$,and a trajectory $\xi_{x\mathcal{X}}: [0,\tau]$ of Σ Satisfying $\frac{d}{dt}\xi_{x\mathcal{X}} = f(\xi_{x\mathcal{X}},\mathcal{X})$,with $\xi_{x\mathcal{X}}(\tau) = x'$ and $\xi_{x\mathcal{X}}(0) = x$.

2.2.4) Comments:

- The main difference between 2.2.2 and 2.2.3, is that in case of a sampled system, the time horizon τ is fixed. In a practical context, τ could correspond to the rate or the clock frequency, at which the controller is applying a new different control signal each time.
- We can see the clear advantage of the definition of a transition system**,** which allows us to describe both time-space continuous control system and time-discrete space-continuous control systems (and eventually time-discrete, space-discrete control systems). This unifying mathematical framework enables us also to define mathematical relations between different descriptions of the systems, example of such relations are Approximate (alternating) simulation relation and feedback refinement relation that we will define in next parts.
- Thank to such relations it could be possible to substitute a control problem on an infinite time-space continuous domain, with an auxiliary, more simple control problem on a finite discrete space (equivalent to a finite-state).

3) Abstraction for non-linear systems based on approximate simulation relations:

3.1) Approximate simulation relations:

Definition 4: (Tabuada, 2009) let $S_a = (X_a, U_a, \underset{a}{\rightarrow})$ and $S_b = (X_b, U_b, \underset{b}{\rightarrow})$ be metric systems equipped with a metric d, with $X_a = X_b$, consider a precision $\varepsilon \in \mathbb{R}^+$.

A relation $R \subseteq X_a \times X_b$ is said to be ε-approximate simulation relation from S_a to S_b to if all those conditions are satisfied.

(i) $\forall\, x_a \in X_a, \exists\, x_b \in X_b$ with $(x_a, x_b) \in R$,
(ii) $\forall\, (x_a, x_b) \in \mathbb{R}$, $d(x_a, x_b) \leq \varepsilon$
(iii) $\forall\, (x_a, x_b) \in \mathbb{R}$, $x_a \overset{u_a}{\rightarrow} x'_a \Rightarrow \exists\, u_b, x_b \overset{u_b}{\rightarrow} x'_b \wedge (x'_a, x'_b) \in R$

We say that S_a is approximately simulated by S_b or that S_b approximately simulate S_a denoted by :

$S_a \preceq^{\varepsilon}_{AS} S_b$ if there exists an ε-approximate simulation relation from S_a to S_b .

Intuitively, this relation means that for every state x_a in S_a, we could attach to it a state x_b in S_b,where those two states should be close enough to each other (up to ε-precision) .if from this state x_a , there is an input u_a that drive the trajectory to a point x'_a, the system b has to find an input u_b which drive the trajectory starting from x_b to a position x'_b that is close enough to the position x'_a (up to ε- precision) . Therefore, S_b should always be able to reproduce or simulate the behavior of system S_a. In terms of precision, this simulation is not exact or perfect, but subject to an error ε. Thus, we call this relation approximate simulation.

Remark: approximate alternate simulation relation is a more general variant than approximate simulation, which applies to nondeterministic system. the notation is: $S_a \preceq^{\varepsilon}_{AAS} S_b$

3.2) Incremental forward completeness and lyapunov characterization

3.2.1) Definition and a simple example:

3.2.1.1) Definition of incremental forward completeness.

Definition 4: (Zamani, Pola, Mazo JR, & Tabuada, 2011) A control system is incrementally forward complete (δ-FC) if it is forward complete and there exist continuous function B : $\mathbb{R}_0^+ \times \mathbb{R}_0^+ \to \mathbb{R}_0^+$ and Y : $\mathbb{R}_0^+ \times \mathbb{R}_0^+ \to \mathbb{R}_0^+$ such that for every $s \in \mathbb{R}^+$, the functions $B(\cdot, s)$ and $Y(\cdot, s)$ belong to the function class κ_∞ ,that for any $x, x' \in \mathbb{R}^n$, any $\tau \in \mathbb{R}^+$, and any v, v', where $v, v' : [0, \tau[\to U$ the following condition is satisfied for all $t \in [0, \tau]$.

$$\|\xi_{xv}(t) - \xi_{x'v'}(t)\| \leq B(\|x - x'\|, t) + Y(\|v - v'\|_\infty, t)$$

The intuitive interpretation of this definition is that for any two arbitrary trajectories, one that starts in x and evolve under an input $v(t)$, and the second starts in x' and evolve under $v'(t)$, the distance between the instantaneous position of both trajectory at any arbitrary time $t \in [0, \tau[$ is bounded by two functions, one function captures the deviation or mismatch between the initial conditions and the second describes the mismatch in terms of inputs. Through this definition we require that trajectories which start close to each other and evolve under close input signal should remain to a certain degree close to each other.

3.2.1.1) A simple example for linear-time-invariant systems:

a particular simple example for could be taken for an LTI control system in the form of:

$$\dot{\xi} = A\xi + B\xi \text{ }, \xi(t) \in \mathbb{R}^n, v(t) \in U \subseteq \mathbb{R}^m$$

This system admit as a solution for the initial condition $x_0 \in \mathbb{R}^n$ and input $v_0(t) \in U$ the trajectory $\xi_{x_0 v_0}$ given as:

$$\xi_{x_0 v_0}(t) = e^{At}x_0 + \int_0^t e^{A(t-Z)}\, Bv_0(Z)dZ = e^{At}x_0 + \int_0^t e^{At}\, Bv_0(Z-t)dZ$$

If we take two different trajectories ξ_{xv} , $\xi_{x'v'}$starting from different initial positons $x, x'$and that evolve under different input signals v, v', we could compute the distance between them as:

$$\|\xi_{xv}(t) - \xi_{x'v'}(t)\| = \left\| e^{At}x + \int_0^t e^{A(t-Z)}\, Bv(Z)dZ - e^{At}x' + \int_0^t e^{A(t-Z)}\, Bv'dZ \right\|$$

By applying the triangular inequality, then considering norms and integrals inequalities, we deduce,

$$\|\xi_{xv}(t) - \xi_{x'v'}(t)\| \leq \|e^{At}(x - x')\| + \left\|\int_0^t e^{At}\, B(v(Z - t) - v'(Z - t)dZ\right\|$$
$$\leq \|e^{At}\|\|x_0 - x'\| + (\int_0^t \|e^{AZ}B\|dZ)\,\|v - v'\|_\infty$$

The choice of $B(\,r, t\,) = \|e^{At}\|\, r$ and $Y(\,r, t\,) = \int_0^t(\|e^{As}B\|\; ds)\; r$ satisfies **definition 5** and the system is δ-FC

3.2.2) Characterization of δ-FC in term of Lyapunov function:

Definition 6: (Zamani, Pola, Mazo JR, & Tabuada, 2011) consider a control system $\Sigma = (\mathbb{R}^n\,, \mathcal{U}\,, f)$ and a smooth function $V :\ \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_0^+$. Function V is called a δ-FC Lyapunov function for Σ if there exist κ_∞ functions $\underline{\alpha}, \overline{\alpha}\,, \sigma$ and $k \in \mathbb{R}$ such that:

(i) for any $x, x' \in\ \mathbb{R}^n, \underline{\alpha}(\|x -\ x'\| \leq\ \ V(x -\ x') \leq \overline{\alpha}\,(\|x -\ x'\|)$;

(ii) for any $x, x' \in\ \mathbb{R}^n$and for any $u, u' \in U$
$$\frac{\partial V}{\partial x} f(x, u) + \frac{\partial V}{\partial x'} f(x', u') \leq\ k. V(x, x') + \sigma(\|u - u'\|)$$

Intuitively $V(x, x')$ could be though as an energy dissipation function, and has a close definition to Lyapunov functions used to investigate stability of Dynamical systems.

It could be proven that a system Σ is a δ-FC, if it admit a δ-FC Lyapanov function. Moreover the functions B and Y in **definition 5** could be analytically given as:

$$B(r, t) = \alpha^{-1}(2e^{kt}\,\overline{\alpha}\,(r)), Y(r, t) = \alpha^{-1}(2\left(\frac{e^{kt} - 1}{k}\right)\sigma(r))$$

3.3) Symbolic Models for a δ-FC control system:

Definition of a grid cover: for a compact set $A \subseteq\ \mathbb{R}^n$, $A = \prod_{i=1}^n [c_i, d_i]$with $c_i \leq, d_i$ .and for a positive constant $\eta \leq \hat{\eta}$ where $\hat{\eta} = \min\{|d_1 - c_1|, \dots\dots |d_n - c_n|\}$ we define as a grid cover to A the set $[A]_\eta$ expressed as:

$$[A]_\eta = \{a \in A \mid a_i = k_i\eta, k_i \in \mathbb{Z}, i = 1, \dots\dots, n\}$$

Now consider δ-FC continuous control system $\Sigma = (\mathbb{R}^n\,, \mathcal{U}\,, f)$ with a compact input set $U = \prod_{i=1}^m [a_i, b_i]\ \subseteq\ \mathbb{R}^m$

Given a sapling time $\tau \in \mathbb{R}^+$,we define $S_\tau(\Sigma) = (\mathrm{X}_\tau\,, U_\tau\,, \underset{\tau}{\rightarrow})$ as a transition system associated Σ :

- $\mathrm{X}_\tau = \mathbb{R}^n$
- $U_\tau = \{\, v_\tau : [0, \tau[\, \to U \mid v_\tau(t) = v_\tau(0) \in U, \forall\, t \in [0, \tau[\,\}$
- $x_\tau \overset{v_\tau}{\rightarrow} x'_\tau$ if there is a trajectory $\xi_{x_\tau v_\tau}: [0, \tau] \to \mathbb{R}^n$ Satisfying $\frac{d}{dt}\,\xi_{x_\tau v_\tau} = f(\xi_{x_\tau v_\tau}, v_\tau)$ with $\xi_{x_\tau v_\tau}(\tau) = x'_\tau$ and $\xi_{x_\tau v_\tau}(0) = x_\tau$

$S_\tau(\Sigma)$ describes therefore the sampled behavior of control system Σ under a sequence of piecewise constant inputs.

For a quadruple of positive parameters $q = \{\tau, \eta, \mu, \theta\}$ We define the system $S_q(\Sigma) = \left(X_q, U_q, \underset{q}{\rightarrow}\right)$ as

- $X_q = [\mathbb{R}^n]_\eta$;
- $U_q = [U_\tau]_\mu$;
- $x_q \overset{uq}{\rightarrow} x'_q$ if $\left\|\xi_{x_q u_q}(\tau) - x'_q\right\| \le B(\theta, \tau) + Y(\mu, \tau) + \eta$, $\xi_{x_q u_q}: [0, \tau] \to \mathbb{R}^n$ Satisfying $\frac{d}{dt}\xi_{x_q u_q} = f(\xi_{x_q u_q}, u_q)$ with $\xi_{x_q u_q}(\tau) = x'_q$ and $\xi_{x_q u_q}(0) = x_q$.

Theorem 1: (Zamani, Pola, Mazo JR, & Tabuada, 2011) Let $\Sigma = (\mathbb{R}^n, \mathcal{U}, f)$ be a δ-FC control system with a compact input set $U = \prod_{i=1}^{m}[a_i, b_i] \subseteq \mathbb{R}^m$. For any precision choice $\varepsilon \in \mathbb{R}^+$, any choice $q = \{\tau, \eta, \mu, \theta\}$ of quantization parameters satisfying:

- $\mu \le \hat{\mu}$, $\hat{\mu} = \min(|b_1 - a_1|, \dots \dots . |b_n - a_n|)$
- $\eta \le \varepsilon \le \theta$

we have $S_\tau(\Sigma) \preceq_{AS}^{\varepsilon} S_q(\Sigma)$

comments: theorem 1 enables us to construct an abstraction $S_q(\Sigma)$ which approximately simulate $S_\tau(\Sigma)$**.** In contrast to $S_\tau(\Sigma)$, $S_q(\Sigma)$ has discrete, countable states ,which is a desired property for any practical implementation, yet $S_q(\Sigma)$ still has an infinite number of states, which represent an inconvenient. this problem is inherent to the fact that we are considering an unbounded state-space domain $[\mathbb{R}^n]_\eta$. Fortunately, many physical systems (including the pendulum) gives the possibility of restricting the state space to a compact bounded domain. In this case it would be possible to construct an abstraction with countable and finite number of state. the procedure, however, is slightly different than what we have described in this part, and uses the more general variant of approximate simulation relation, which is the approximate alternate simulation relation (ASS).

In practice to construct an abstraction, we need to define a bounded domain D of our physical variables, choose a proper precision ε and suitable discretization parameters $q = \{\tau, \eta, \mu, \theta\}$ which respect the conditions of **theorem 1**. We generate space-state symbols by determining the grid points contained inside the domain D, and the input symbols by determining the grid points inside U.

Then we should compute all the possible transitions of the system, as described above. for the computation of transitions, we need an analytical expression the functions $B(s, t)$ and $Y(s, t)$ obtained either directly, or through a δ-FC Lyapunov candidate function.

While this is very easy for LTI system as we showed in example **3.2.1.1**, determining the expression of $B(s, t)$ and $Y(s, t)$ is a burden task for complex non-linear system, particularly in the case of our rotatory inverted pendulum (unless we consider a linearized version of the dynamics).

This shows some limitation of this type of abstraction, because it is not easy to express $B(s,t)$ and $Y(s,t)$ analytically or to find a good candidate δ-FC Lyapanov function in the most general case.

4) Abstraction for non-linear systems based on feedback refinement relation:

In this part, we use the alternative notation of a simple transition system S , $S = (X, U, F)$ instead of the notation $S = (X, U, \rightarrow)$, where $F : \mathrm{X} \times U \rightrightarrows \mathrm{X}$ is a map-valued function, which for each state and input associate the set of the future states.

We note $U_S(x) = \{u \in U \mid F(x,u) \neq \emptyset\}$ as the set of admissible inputs for a state x .

We define $\varphi(t, x_0, u)$,called nominal solution, as the unique trajectory that solve the initial value problem $x = f(x,u), x(0) = x_0$ under constant input u.

4.1) Definition of feedback refinement relationship:

(Reissig, Rungger, & Alexander, 2015)let S_1 and S_2 be simple systems, $S_i = (X_i\,, U_i\,, F_i), i = \{1,2\}$ and assume $U_1 \subseteq U_2$.

a strict relation $Q \subseteq X_1 \times X_2$ is a feedback refinement relation from S_1 to S_2 if the following holds for all $(x_1, x_2) \in Q$

(i) $U_{S_2}(x_2) \subseteq U_{S_1}(x_1)$;

(ii) $u \in U_{S_2}(x_2) \Rightarrow Q(F_1(x_1,u)) \subseteq F_2(x_2,u)$

In this case, we note $S_1 \preceq_Q S_2$

4.2) Definition of growth bound:

(Reissig, Rungger, & Alexander, 2015) Let $\Sigma = (\mathbb{R}^n\,, \mathcal{U}\,, f)$ be a continuous control system with input set U, Consider the sets $K \subseteq \mathbb{R}^n, U' \subseteq U$ and the sampling $\tau > 0$. A map $\beta : \mathbb{R}_+^n \times U' \rightarrow \mathbb{R}_+^n$ is a growth bound on K, U'associated with τ if the following hold

(i) $\beta(r,u) \geq \beta(r',u)$ wherever $r \geq r'$ and $u \in U'$.

(ii) $[0,\tau] \times K \times U' \subseteq dom\ \varphi$,and if ξ is a solution of $\frac{d}{dt}\xi = f(\xi\,, u$
On $[0,\tau]$ with input $u \in U'$, initial condition $\xi(0) = \ \xi_0, and\ p \in K$ then:
$|\xi(\tau) - \varphi(\tau, p, u)| \leq \beta(|\ \xi_0 - p|, u)$ holds.

Remark: it should be stressed out that $\leq$ and $\geq$ are vector component comparators in this definition.

Roughly speaking, β quantifies the effect of a mismatch between two different initial conditions on the distance of their corresponding trajectory after a simulation time of τ. Since β is an increasing map with respect to the distance argument, a bigger mismatch in initial conditions, should result in a more conservative, larger estimation of the deviation of the solutions.

4.3) Analytical expression of growth bound:

it is possible to find an analytical expression of the growth bound defined above, by using a Lipchitz–type estimation. We consider the expression of the growth bound for a non-linear, unperturbed systems.

(Reissig, Rungger, & Alexander, 2015) For $\Sigma = (\mathbb{R}^n , \mathcal{U} , f)$, Let τ > 0, $U' \subseteq U$ and assume in addition that $f(\cdot, u)$ is continuously differentiable for every $u \in U'$. Furthermore, let $K \subseteq K' \subseteq \mathbb{R}^n$ with K' being convex, so that for any $u \in U'$, any $\tau' \in [0, \tau]$ and any solution ξ on $[0, \tau]$ of Σ with input u and $\xi(0) \in K'$,for all $\tau \in [0, \tau']$ $we\ have, \xi(t) \in K'\ for\ all\ t \in [0, \tau']$. Lastly, let the parametrized matrix: $L: U' \to \mathbb{R}^{n \times n}$ satisfy

$$L_{i,j}(u) \geq \begin{cases} D_i f_i(x, u), & if\ i = j \\ |D_i f_i(x, u)|, & otherwise \end{cases}$$

For all $x \in K'$ and all $u \in U'$. the map β given by $\beta(r, u) = e^{L(u)\tau} r$ Is a growth bound on K, U' associated with τ and Σ .

Remark: we could see that β is the solution of the differential equation

$$\dot{r} = L(u)r$$

This will give us the practical possibility to compute β with numerical schemes (like Rungga-Kutta method), if the direct evaluation of the matrix exponential turns out to be computationally more expansive.

4.4) Construction of an abstraction based on feedback refinement relation:

a control system $\Sigma\ (\mathbb{R}^n , \mathcal{U} , f)$,and let $S_1 = (X_1 , U_1 , F_1)$ be the corresponding sampled system.

We consider a system $S_2 = (X_2 , U_2 , F_2)$, which state alphabets X_2 are defined as a cover of the state alphabet X_1 . The elements of this cover are non-empty, closed unbounded hyper intervals i.e.:

$\forall\, x_2 \in X_2 \Rightarrow x_2 = [\![a, b]\!]$,for $a, b \in (\mathbb{R} \cup \{\pm\infty\})^n, a \leq b$.

we work with a set $\tilde{X}_2 \subseteq X_2$ of compact bound cells. $\tilde{X}_2$ are called real quantize symbols, Whereas $X_2 \setminus \tilde{X}_2$ are called overflow symbols.

We call S_2 an abstraction of S_1 based on $\tilde{X}_2$ and β if :

(i) X_2 is a cover of X_1 by no empty, closed by hyper intervals and every element $x_2 \in \tilde{X}_2$ is compact
(ii) $U_2 \subseteq U_1$
(iii) For $x_2 \in \tilde{X}_2, x_2' \in X_2$ and $u \in U_2$ we have

$$(\varphi(\tau, c, u)) + [[r', r']]) \cap x_2' \neq \emptyset \Rightarrow x_2' \in F_2(u_2, u)$$

Where $[[a, b]] = x_2, c = \frac{b+a}{2}$. $r = \frac{b-a}{2}$ and $r' = \beta(r, u)$;

(iv) $F_2(x_2, u)$ where $x_2 \in X_2 \setminus \tilde{X}_2$, $u \in U_2$

(Reissig, Rungger, & Alexander, 2015)**Theorem:** $if\ \ \beta$ is a growth-bound, then $Q: \{(x_1, x_2) \mid x_1 \in x_2\}$ is feedback refinement relation between $S_1\ and\ S_2$,denoted $S_1 \preceq_Q S_2$.

Remarks:

- In practice, $\tilde{X}_2$ are characterized through a uniform grid $\eta\mathbb{Z}^n$=$\{c \in \mathbb{R}^n \mid \exists\ \mathrm{k} \in \mathbb{Z}^n, \forall\ i \in [1;\mathrm{n}], c_i = \mathrm{k}_i\eta_i\}$ with grid discretization parameter $\boldsymbol{\eta} \in \mathbb{R}_+^n$
- We could interpret c the old cell center, r the old cell radius, $\varphi(\tau, c, u)$ the new cell center, and r' as the new cell center.it is possible to visualize transitions for two dimensional case in [figure 4.1].

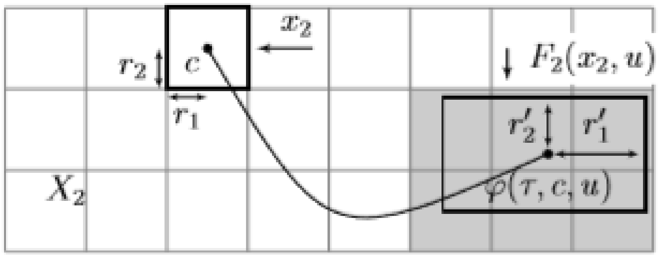


Figure 4.1 visualization of transitions in the abstract domain. From an original cell with center c and radius r, we compute the new center and new radius and determine the intersections to determine the transitions.

5) Comparisons and discussion:

We have presented two strategies from the literature to construct valid abstraction or symbolic models, based on different mathematical relations. Details and rigorous proofs concerning δ-FC systems and their abstractions could be found in (Zamani, Pola, Mazo JR, & Tabuada, 2011), those concerning feedback relations are exhaustively presented in (Reissig, Rungger, & Alexander, 2015):

Our ultimate goal behind this chapter is not only to have a global description of abstractions, but to guide the choice of the method to be used for the coming practical implementation

Although both approaches share a lot of similarities, employing them in practical context will result in a different implementation difficulty level. We summarize the most important distinctions between the two approaches from a practical point of view.

	approximate simulation relation based abstraction	Feedback refinement based abstraction
Space discretization parameters	η	η_1,η_2 ,…, η_n
Influence of system dimension on the number Space discretization parameters	No influence	Grows up with higher dimension
Computation of transition function	Allowed directly through two function B and Y,or indirectly through one δ-FC Lyapunov function.	Allowed through a growth bound β
Analytical expressions of the Characterizing functions	Does not exist in the general case, we needs to try many candidate functions	Exist, and is possible through a Lipchitz matrix estimation

It turns out, that although feedback refinement relation will result in more design parameters for high dimensional system(since we need to make an independent space discretization step in each direction), It is still more advantageous because an analytical expression for the growth bound β does exist and is straightforward to compute. On the other side characterizing analytically the functions B and Y ,or finding a δ-FC Lyapunov function is generally a complicated task and has no success guarantees.

Based on this argument, we chose to adapt the feedback refinement relation for our future implementation.

another argument that further motivate the choice of feedback refinement based implementation:

is related to the abstract controller refinement step, which is very simple and uses only a static quantizer.

.

Chapter 5: Basic implementations in SCOTS

1) Overview:

In this thesis, we are aiming at designing a controller to enforce a safety specification on a rotatory inverted pendulum. First, we have derived the dynamics, which turns out to be highly nonlinear, then we diseased the two possible abstraction approaches found in the literature. We ended by choosing the feedback refinement relation based abstractions the implementation. The best accessible free tool that we could use in this case was SCOTS.

2) Presentation of SCOTS:

SCOTS is an open source software, developed by the professorship of hybrid control systems and is addressed to researches in the area of formal method of cyber physical systems. SCOTS provides a basic implementation for the construction of discrete abstraction of non-linear, possibly perturbed, and prone to measurement-errors systems according to a feedback refinement relation.

Furthermore, this tool contains tow algorithms (minimal and maximal fixed point) that serves for the synthesis of controller, mainly subject to invariance or reachability specifications. However, it still possible to extend it with other external tools to enforce more complex specification. (Rungger, 2017)

SCOTS code is implemented in C_{++} ,in a header-only style, and provide additionally a Matlab interface to simulate the close loop of the controller. It comes up with various examples to be tested and simulated.

There exists two version of scots, based on different data structures. SCOTSv01 uses a Binary decision diagram (BDD) while SCOTSv02 is based on sparse matrices. Because sparse matrices offer faster computation, we chose to work with SCOTSv02.

3) Control problem formulation:

We are interested in designing a controller that enforce an invariance specification to stabilize the pendulum in the upward, unstable position.

In chapter 3 we have derived the ODE describing the pendulum and formulate the in a state-space representation:

$$\xi(t) = f(\xi(t), u) \quad (1)$$

Where $f: \mathbb{R}^4 \times U \to \mathbb{R}^4$ and $U \subseteq \mathbb{R}$

Given a time horizon τ, we define a solution of (1) on $[0, \tau]$ under constant input $u \in U$ as the absolutely continuous function $\xi: [0, \tau] \to \mathbb{R}^4$ that satisfies (1).

The sampled behavior of the pendulum could be casted as a simple system

$$S_1 := (X_1, U_1, F_1)$$

With state alphabet $X_1: = \mathbb{R}^4$, $U_1 := U$ and the transition function $F_1(x, u) := \{ x' \mid \exists_{\xi}$ is a solution of (1) with $\xi(0) = x \wedge \xi(\tau)$=$x'$.$\}$

Because $f(\cdot, u)$ is smooth map, a solution does always exist, so that F_1(x,u) contains exactly one element (deterministic).

We define $(X_1)^\infty := U_{T\in Z\geq 0\cup\{\infty\}} X_1{}^{[0,T[}$ as the sequence of all possible output sequences.
An invariance specification associated with $\mathrm{Z}_1 \subseteq X_1$ for the system S_1 is given as

$$\Sigma_1 := \{\mathrm{x}_1 \in X_1^\infty \mid \forall_{[0,\infty[}: \mathrm{x(t)} \in Z_1\}$$

Together (S_1, Σ_1) constitutes a control problem.

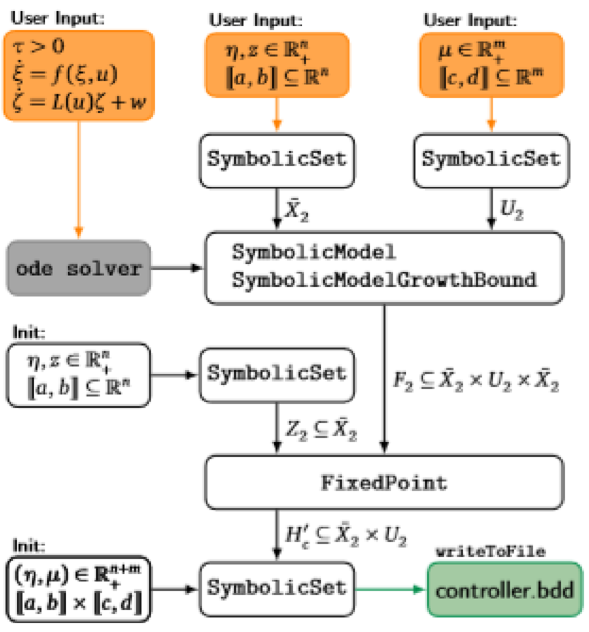
The basic idea in SCOTS is to substitute the original control problem with an abstract system (S_2, Σ_2),then solve it as an auxiliary control problem by finding a controller C, and finally refine the abstract controller to the real plant.

The enabling theorem that ensures the correctness of this procedure in SCOTS is the following:

(Reissig, Rungger, & Alexander, 2015) Given two control problem (S_i, Σ_i), $i \in \{1,2\}$. Suppose that Q is a feedback refiment relation from S_1 to S_2 and Σ_2 is an abstract specification of Σ_1 .if C solves the control problem (S_2, Σ_2), then C o Q solves the control problem.

4) SCOTS algorithmic flow and user-dependent parameters:

Understanding the work flow of SCOTS might be helpful for a successful implementation of a controller, but more important is to figure out the user-dependent design parameters, and where to include them in the code. Figure 5.1 summarize the software work-flow and the user-dependent parameters.



User Input:
$\tau > 0$
$\dot{\xi} = f(\xi, u)$
$\dot{\zeta} = L(u)\zeta + w$
User Input:
$\eta, z \in \mathbb{R}^n_+$
$[\![a,b]\!] \subseteq \mathbb{R}^n$
User Input:
$\mu \in \mathbb{R}^m_+$
$[\![c,d]\!] \subseteq \mathbb{R}^m$
SymbolicSet
SymbolicSet
$\bar{X}_2$
U_2
ode solver
SymbolicModel
SymbolicModelGrowthBound
Init:
$\eta, z \in \mathbb{R}^n_+$
$[\![a,b]\!] \subseteq \mathbb{R}^n$
SymbolicSet
$F_2 \subseteq \bar{X}_2 \times U_2 \times \bar{X}_2$
$Z_2 \subseteq \bar{X}_2$
FixedPoint
$H_c' \subseteq \bar{X}_2 \times U_2$
writeToFile
Init:
$(\eta, \mu) \in \mathbb{R}^{n+m}_+$
$[\![a,b]\!] \times [\![c,d]\!]$
SymbolicSet
controller.bdd

Figure 5.1 algorithmic work-flow in SCOTS and user-dependent entries

In practice, our task will resume to the following three steps:

- Enter the ODE, with the numerical values of physical parameters.
- Compact a Growth bond in terms of a Lipchitz matrix $L(\mathrm{x}, \mathrm{u})$.

- Estimate different domains and discretization parameters.

While the first two steps are relatively straightforward, the main difficulty lies within the third one. Since we are dealing with 4-dimensional system, the number of parameters we need to estimate is relatively-high:

- We have to select 8 variables describing the compact set $Z \subseteq \mathbb{R}^4$ that defines an invariance specification. Although a specification is normally set to be freely chosen, we still need to select one that is physically achievable. If we go for a very tight and strict specification, this could result in no solution, on the other hand, a specification defined on a large domain is not very interesting in practice.
- Another 8 variables which stand for the compact set $X \subseteq \mathbb{R}^4$ describing the space-state domain, we have also to be careful with that choice, we have to choose a sufficiently large domain a solution could exists, but avoid the choice of excessively large domain which could lead to computational problems.
- 4 parameters related to the discretization of the state space domain. Here, a careful choice is also needed because large discretization could result in finding no controller while fine discretization could lead to an extreme computational effort.
- determine a suitable compact interval $U \subseteq \mathbb{R}$ that describes the input and a suitable input discretization μ .
- Finally set a good time horizon τ, that is compatible with the reactivity of the system. This choice is also critical, very large time horizon τ could result in an insolvable control problem, whereas ,small time horizon requires a choice of a fine space discretization.(there exist a trade-off between space discretization and time discretization)

In total, we have to estimate 24 parameters, which should be suitable to solve the control problem with a reasonable computational time and without exceeding the memory limitations.

Because of the complex balance we have to set while choosing 23 parameters, and the existence of a trade-off between certain parameters like sampling time and space discretization. The task of finding a controller is similar to optimization task under constraints.

But what makes the task a real challenge is the absence of a well-established theory to guide the choice of parameters in the general case. Therefore, there is need to count on some intuition and self-initiative.

For example, based on a simple symmetry argument related to the geometry of the pendulum, we could think of a possible reduction of design parameters to 15, by taking Z, and X and U to be symmetrical hyper-intervals around some origin point.

Since the estimation of parameters was the hardest part in this thesis, we have devoted a full chapter for that purpose. In the rest of this chapter, we will focus on basic and straightforward steps of the implementation.

5) Basic implementation steps:

5.1) Inserting the dynamics into SCOTS:

First we precise the dimension of the state-space and the input space [fig 5.2].

```
/* state space dim */
const int state_dim=4;
/* input space dim */
const int input_dim=1;
```

Figure 5.2 code fragment for precising the dimension dimensions

Then we add the ODE in the code [fig 5.3], and the numerical values of physical parameters [fig 5.4].

```
auto system_post = [](state_type &x, const input_type &u) noexcept {
  /* the ode describing the pendulum */
  auto rhs =[](state_type& xx,  const state_type &x, const input_type &u)
noexcept {

      const double det=J0*J2+J2*J2*sin(x[1])*sin(x[1])-
c*cos(x[1])*c*cos(x[1]);
      xx[0]=x[2];
      xx[1]=x[3];
      xx[2]=(1/det)*(-J2*b1*x[2]+c*cos(x[1])*b2*x[3]-
J2*J2*sin(2*x[1])*x[2]*x[3]-
0.5*J2*c*cos(x[1])*sin(2*x[1])*x[2]*x[2]+J2*c*sin(x[1])*x[3]*x[3]
             +g*0.5*c*c*sin(2*x[1])/L1+J2*(q1*u[0]-q2*x[2]));
      xx[3]=(1/det)*(x[2]*c*cos(x[1])*b1-
x[3]*b2*(J0+J2*sin(x[1])*sin(x[1]))+x[2]*x[3]*c*J2*cos(x[1])*sin(2*x[1])-
0.5*x[2]*x[2]*sin(2*x[1])*(J0*J2+J2*J2*sin(x[1])*sin(x[1]))-
             0.5*x[3]*x[3]*c*c*sin(2*x[1])-
g*m2*l2*sin(x[1])*(J0+J2*sin(x[1])*sin(x[1]))-c*cos(x[1])*(q1*u[0]-q2*x[2]));

     };
  scots::runge_kutta_fixed4(rhs,x,u,state_dim,tau,5);
```

Figure 5.3 the ODE describing the pendulum

```
/* parameters for system dynamics */

const double g=9.81;   const double jj1=9.982910141666664*0.0001;const double j2=0.001198730801458;

const double m1=0.2570;const double m2=0.127;  const double L1=0.2159000;const double L2=0.336550;

 const double l1=0.061912500000000;

const double l2=0.155575000000000;  const double b1=0.002400000000000;const double b2=0.002400000000000;
const double Rm=2.600000000000000; const double Ng=0.900000000000000;

const doubleNm=0.690000000000000;const double Kg=70; const double Kt=0.007682969729280;

const double Km=0.007677634454753; const double q1=Ng*Kg*Nm*Kt/Rm;

const double q2=Ng*Kg*Kg*Km*Nm*Kt/Rm; const double J1=jj1+m1*l1*l1;

const double J2=j2+m2*l2*l2;const double J0=J1+m2*L1*L1;const double c=m2*L1*l2;
```

fig 5.4 numerical values of the physical parameters of the pendulum

5.2) Computation of the growth-bound:

We have to determine the growth bond defined in 4.2, and whose analytical expressions is given in 4.3. This step, though not hard, should be carried with a lot of attention if the derivations are done manually. Alternatively, we could deploy Symbolic Math Toolbox™.

We have manually computed $L_{11}(\mathrm{x},\mathrm{u}) = |D_1 f_1(x,u)|,\dots, L_{44}(\mathrm{x},\mathrm{u}) = D_4 f_4(\mathrm{x},\mathrm{u})$.

Since the resulting expression are so lengthy, we preferred to insert them directly in lambda reserved the computation of growth bound in SCOTS [figure 5.6]

We provide also the general form of the resulting L matrix.

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & L_{32}(\mathrm{x},\mathrm{u}) & L_{33}(\mathrm{x},\mathrm{u}) & L_{34}(\mathrm{x},\mathrm{u}) \\ 0 & L_{42}(\mathrm{x},\mathrm{u}) & L_{43}(\mathrm{x},\mathrm{u}) & L_{44}(\mathrm{x},\mathrm{u}) \end{pmatrix}$$

The expressions obtained for L_{ij} are still however a function of x and u, whereas in definition, it is required to only depend on u.to achieve this we need to determine a priori-enclosure $K' \subseteq \mathbb{R}^4$, then every term L_{ij} (x,u) should be maximized over K' with respect to x. A solution would be to resort to some interval arithmetic tool like vnode-lp.

However, we encountered difficulties while trying to install those tools. Fortunately, my supervisors suggested that is possible to skip the computation of the a priori-enclosure and use the advantages that SCOTSv02 offers by allowing the Lipchitz matrix to depend also on the variable x [figure 5.5]

This alternative will even result in a less conservative over approximation of the attainable sets, because an adapted growth-bond for each cell will be computed.as consequence, we obtain less transitions in the abstraction and this will reduce the use of memory.

However, a drawback of this alternative is computational, because instead of determining once a unique growth bound that will work for all the cells, the computation, through the Rungga-Kutta scheme, should be carried for every cell within the abstract domain.

```
auto radius_post = [](state_type &r, const state_type&x, const
input_type &u) noexcept {
  /* the ode for the growth bound */
  auto rhs =[x](state_type& rr,  const state_type &r, const input_type
&u) noexcept {
```

Figure 5.5: dependence of the growth bound lambda on the cell center x in scotsv02

```
auto radius_post = [](state_type &r, const state_type&x, const input_type
&u) noexcept {
  /* the ode for the growth bound */
  auto rhs =[x](state_type& rr,  const state_type &r, const input_type
&u) noexcept {

const double detr=J0*J2+J2*J2*sin(x[1])*sin(x[1])-
c*cos(x[1])*c*cos(x[1]);
const double f3=-J2*b1*x[2]+c*cos(x[1])*b2*x[3]-
J2*J2*sin(2*x[1])*x[2]*x[3]-
0.5*J2*c*cos(x[1])*sin(2*x[1])*x[2]*x[2]+J2*c*sin(x[1])*x[3]*x[3]
+g*0.5*c*c*sin(2*x[1])/L1+J2*(q1*u[0]-q2*x[2]);
const double f4=x[2]*c*cos(x[1])*b1-
x[3]*b2*(J0+J2*sin(x[1])*sin(x[1]))+x[2]*x[3]*c*J2*cos(x[1])*sin(2*x[1])-
0.5*x[2]*x[2]*sin(2*x[1])*(J0*J2+J2*J2*sin(x[1])*sin(x[1]))-
               0.5*x[3]*x[3]*c*c*sin(2*x[1])-
g*m2*l2*sin(x[1])*(J0+J2*sin(x[1])*sin(x[1]))-c*cos(x[1])*(q1*u[0]-
q2*x[2]);

const double  L32=abs((1/detr*detr)*(detr*(-c*sin(x[1])*b2*x[3]-
2*J2*J2*cos(2*x[1])*x[2]*x[3]+0.5*J2*c*sin(x[1])*sin(2*x[1])*x[2]*x[2]-
J2*c*cos(x[1])*cos(2*x[1])*x[2]*x[2]
+J2*c*sin(x[1])*x[3]*x[3]+g*c*c*cos(2*x[1])/L1)-
f3*(2*J2*J2*cos(x[1])*sin(x[1])+2*c*cos(x[1])*c*sin(x[1]))));
const double  L33=(1/detr)*(-J2*b1-J2*J2*sin(2*x[1])*x[3]-
2*c*cos(x[1])*sin(2*x[1])*x[2]-J2*q2);
const double  L34=abs((1/detr)*(c*cos(x[1])*b2-
J2*J2*sin(2*x[1])*x[2]+2*J2*c*sin(x[1])*x[3]));
const double  L42=abs((1/detr*detr)*(f4*(2*J2*J2*cos(x[1])*sin(x[1])+
2*c*cos(x[1])*c*sin(x[1]))+detr*(-x[2]*c*c*sin(x[1])*b1-
2*r[3]*b2*J2*cos(x[1])*sin(x[1])-x[2]*x[3]*c*J2*sin(x[1])*sin(2*x[1])
+2*x[2]*x[3]*c*J2*cos(x[1])*cos(2*x[1])-
x[2]*x[2]*cos(2*x[1])*(J0*J2+J2*J2*sin(x[1])*sin(x[1]))-
0.5*x[2]*x[2]*sin(2*x[1])*(2*J2*J2*cos(x[1])*sin(x[1]))-
x[3]*x[3]*c*c*cos(2*x[1])+c*sin(x[1])*(q1*u[0]-q2*x[2])
              -g*m2*l2*cos(x[1])*(J0+J2*sin(x[1])*sin(x[1]))-
g*m2*l2*sin(x[1])*2*J2*cos(x[1])*sin(x[1]))));

const double   L43=abs((1/detr)*(c*cos(x[1])*b1+
x[3]*c*J2*cos(x[1])*sin(2*x[1])-
x[2]*sin(2*x[1])*(J0*J2+J2*J2*sin(x[1])*sin(x[1])+c*cos(x[1])*q2)));
const double   L44=(1/detr)*(b2*(J0+J2*sin(x[1])*sin(x[1]))
+x[2]*c*J2*cos(x[1])*sin(2*x[1])-x[3]*c*c*sin(2*x[1]));

      rr[0]=r[2];
      rr[1]=r[3];

      rr[2]=L32*r[2]+L33*r[3]+L34*r[4];
      rr[3]=L42*r[2]+L43*r[3]+L44*r[4];

     };
  scots::runge_kutta_fixed4(rhs,r,u,state_dim,tau,5);
};
```

Figure 5.6 code fragment for the computation of the growth bound and the expressions of the L matrix terms

Chapter 6: estimation of design parameters in SCOTS.

1) Overview:

The theory for symbolic methods is mathematically well established. As a result, many steps in controller synthesis could be fully automated, for example computing the transitions, or solving the auxiliary control problem with Algorithms like minimal and maximal fixed point. However, this does not mean that finding a controller is a mere press-bottom task. In fact, to guarantee the existence of a controller, the execution of Computations within a reasonable time lapse, and the respect of hardware resource limitations as memory, the engineer or the designer has to choose carefully and wisely some parameters, for instance, the sampling time, the state-space domain and space discretization. If this choice might be easy for systems with simple dynamics or low dimensions, it becomes more challenging for system with complex, highly reactive dynamics, or those with a high state-space dimension.

In our case we seek to determine a good and balanced choice of:

- τ : the sampling time
- $\vec{\mathrm{z}} = (\,\mathrm{z}_1, \mathrm{z}_2, \mathrm{z}_3, \mathrm{z}_4\,) \in \mathbb{R}_+^4$, which describes hyper-interval $[\pi - \mathrm{z}_1, \pi + \mathrm{z}_1] \times [-\mathrm{z}_2, \mathrm{z}_2\,] \times [\,-\mathrm{z}_3, \mathrm{z}_3\,] \times [\,-\mathrm{z}_4, \mathrm{z}_4\,]$ expressing a reasonable, physically achievable invariance specification that maintains the pendulum near the upwards origin position $(\pi, 0,0,0)$
- $\vec{x} = (x_1, x_2, x_3, x_4) \in \mathbb{R}_+^4$, which expresses a compact bounded symmetrical state space domain $[\pi - x_1, \pi + x_1] \times [-x_2, x_2\,] \times [\,-x_3, x_3\,] \times [\,-x_4, x_4]$,and u which expresses a symmetrical input interval $[-u, u]$
- $\vec{\eta} = (\,\eta_1, \eta_2, \eta_3, \eta_4\,) \in \mathbb{R}_+^4$, related to space discretization parameters ,and $\mu \in \mathbb{R}_+$ which stands for the input discretization.

The step of finding the right parameters was one of hardest and most time-consuming phase in this project. the difficulty we faced is mainly inherent to:

- The high number of parameters to be estimated, due the four-dimensional nature of the dynamics.
- The non-existence of a general, standard procedure to guide the choice
- The curse of dimensionality, resulting in an exponential time complexity that we have to manage with limited processor and memory performance.

To overcome this hurdle, and guarantee finding a controller, we had to rely on some intuition, a lot of try and error, and an exhaustive numerical simulation of the ODE. We had also to consider whether a reduction of the dynamins dimension is possible.

2) Time complexity analysis in SCOTS:

We want carry on a rough time complexity analysis. By trying various examples available in SCOTS, we figured out that computing the abstraction is the costliest computational part (dominate the synthesis step). Within this step itself, the computation of transitions is the a very time-consuming phase, and depends generally on the number of state alphabets, the number of input state alphabets and the execution time of Rungga-kutta algorithm,

If we denote W the execution time of SCOTS and let it depend on $\vec{x}$, $\vec{\eta}$, τ, μ, u , after a careful analysis of the code used to compute the transitions, it is possible to find a lower bound W in the form of:

$$W(\vec{x}\,,\vec{\eta},u,\mu,\tau) \geq \frac{u\tau}{\mu h} t_{op} \prod_{i=1}^{4} \left(\frac{2x_i}{\eta_i}\right)$$

Where h is the integration time step used by Rungga -kutta , and t_{op} the single cost of one iteration within the numerical integration scheme .

Since $\prod_{i=1}^{4} \left(\frac{2x_i}{\eta_i}\right) = e^{\sum_{i=1}^{4}\left(\frac{2x_i}{\eta_i}\right)}$ we obtain an alternative representation form of the lower-bound:

$$W(\vec{x}\,,\vec{\eta},u,\mu,\tau) \geq \frac{u\tau}{\mu h} t_{op} e^{\sum_{i=1}^{4}\left(\frac{2x_i}{\eta_i}\right)}$$

This form elucidates the exponential time-complexity of scots. This complexity is responsible for what we call the curse of dimensionality, which is a common feature in many gridded-based abstraction methods. A similar argumentation will show that the exponential complexity issue applies also to the memory use.

The curse of dimensionality is one of the cause that makes finding a controller a hard task, especially if we have to choose fine discretization or large state domains. The choice of optimal parameters, which at the same time ensures finding a controller and respects hardware limitation, is a delicate task, for which there is no success guarantee.

3) Solution strategies for the four-dimensional model

3.1) Brute force strategy:

One of the possibility would be to resort to writing a code that test all the possible combinations of the 15 parameters, but since there is uncountable many combinations, and because of the exponential time-complexity of SCOTS, this strategy could take an infinite amount of time to executed, and is therefore unrealistic.

3.2) A try-and-error strategy:

Since a brute-force strategy is not possible, we thought instead to test heuristically a finite number of possible combination, and hope that one of them could lead to finding a controller. We tried tens of combination of parameters. Unfortunately, no one of them could solve the problem. During those trials We experienced two different situations:

- In case we used small domains, or large discretization, the computation in SCOTS could be achieved fast. But always leads to an empty winning domain, which mean that no controller was found. It was also difficult to say which parameters are culpable of not finding a controller, so we do not have a way of adjusting them.
- In case we preferred the used of large domains, or small discretization parameters, the computation will take hours and eventually breaks down since the memory could not store all the transitions.

After a long series of tries, we thought it should be better to make some conjectures or assumptions based on our intuition to the dynamics of the pendulum, because a rigorous analysis or a valid strategy could not be developed:

- We remarked that the pendulum is highly reactive in the upwards positive, which means that it moves and falls very fast. Therefore, it is more probable that the correct choice of the sampling time corresponds to small sampling values (in the range of milliseconds.)
- because of the trade-off between the sampling time and space discretization parameters, the choice of small sampling time will force us to choose also small space discretization parameters. As a consequence, we stipulate that selecting fine discretization parameters is a more plausible choice to enhance the chance of finding a controller, but on the other side will lead to a consequent computational effort and high memory requirements. At this level we could not judge the achievability of our task on the computers we had access. However, we had to believe in the existence of solution and try harder to find some optimal choice of parameters that could reconcile this conflict.

Finally, we had to consider some order while trying different parameters, to create a simple heuristic algorithm. By intuition, we think this order could be a good one:

- first we fix a specification $\vec{\mathrm{z}}$ that we assume physically achievable.
- Then we fix a state domain $\vec{x}$ that is large enough to cover the specification.
- we chose a small sampling time τ.
- We choose an input u within the range of the voltage the real plant, and a good μ to allow a reasonable number of input alphabet(not unnecessary big but rather big enough to give sufficient control possibilities at each state)
- We try to find the best possible discretization parameters $\vec{\eta}$ that could enable a solution but avoid the exploding of state-space alphabet.

▪ If we fail, we have to reiterate by selecting a new specification.

We apply this heuristic algorithm in the next part, with support of an exhaustive numerical simulation at each step, particularly at the level of choosing $\vec{\eta}$.

3.3) Simulation-based strategy:

To ease the choice of parameters, we decided to build a Simulink model for our pendulum, which simulates the behavior its governing ODE.

We designed this model by transforming the terms of ODE in many Simulink blocks. The visual complexity of the model mirrors the complexity and the high-nonlinearities linearity of the ODE. [figure 6.1].

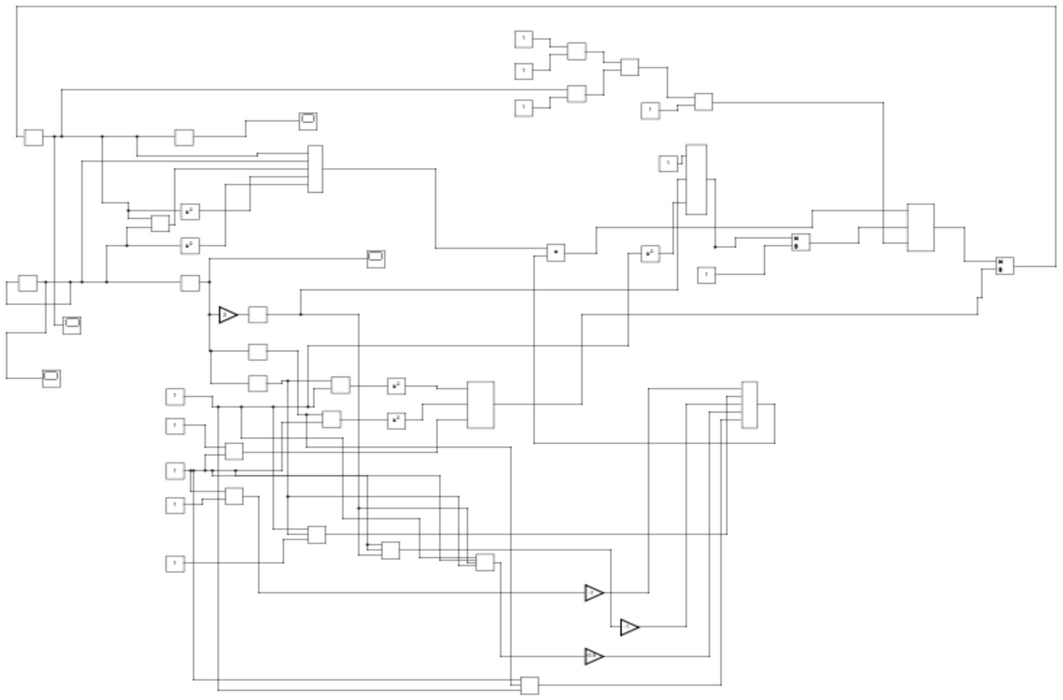


Figure 6.1 Simulink model with blocks and connections depicting the complexity of the ODE

To simplify the Simulink model visually we could merge many blocks with each other [figure 6.2]

We had to work with the Simulink model for a long time, and measure the response of the pendulum, for different initial conditions, different inputs, and different simulation time. After exhaustive simulation we formulate the approximate choice of parameters as:

- Specification: $\left[\pi - \frac{2\pi}{35}, \pi + \frac{2\pi}{35}\right] \times [-\pi, \pi\ ] \times [\ -\pi, \pi\ ] \times [\ -\pi, \pi\ ]$
- State-space: $\left[\pi - \frac{\pi}{5}, \pi + \frac{\pi}{5}\right] \times [-\pi, \pi\ ] \times [\ -\pi, \pi\ ] \times [\ -\pi, \pi\ ]$
- Sampling time: 0.05
- Input-space: $[-4,4]$

- Input-discretization: 0.5
- Space-discretization: $\{\frac{\pi}{100}, \frac{\pi}{50}, \frac{\pi}{50}, \frac{\pi}{50}\}$

We include those parameters in SCOTS, and compiled the files, we had to wait for a long time for the execution, unfortunately the computations could not be finished because at a certain point, the memory of the pc would be fully allocated and no more transitions could be further stored.

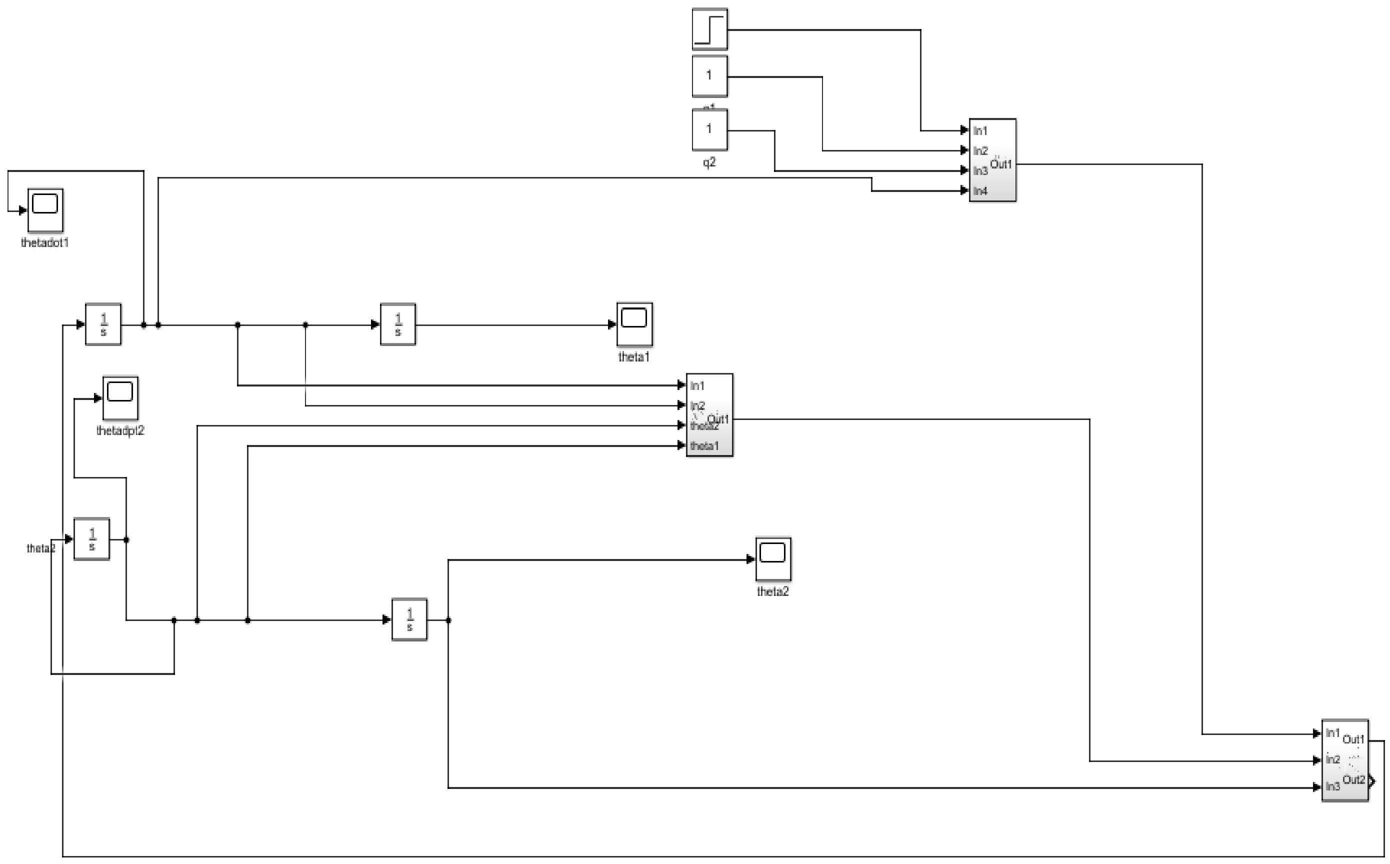


Figure 6.2 Figure 6.2 Simulink model after merging some blocks

Although we have done an effort on simulation, we still could not verify the result of our predictions, unless we can get access to a more performant computers (like super-computers). Since we had not this possibility, the only alternative we can still think of is to investigate whether it is possible to reduce the dimension model, which in the positive case, will result in a better chance of finding a controller because the curse of dimensionality in less pronounced in lower dimensions.

4) Comtroller for the reduced-dimension model:

We could remark, that the ODE of our system has the following form:

$$\dot{\xi}(t) = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} = \begin{bmatrix} f_1(\theta_1) \\ f_2(\dot{\theta}_1, \dot{\theta}_2, \theta_2) \\ f_3(\dot{\theta}_1, \dot{\theta}_2, \theta_2) \\ f_4(\dot{\theta}_1, \dot{\theta}_2, \theta_2) \end{bmatrix}$$

this form suggest that the angle θ_1 is not directly influencing the dynamics of the other variable $\ddot{\theta}_1$, $\ddot{\theta}_2$ and $\dot{\theta}_2$we have also checked the linearized version of the ODE. The matrix A, of the linearized version has as null column in its first column, the same applies to the matrix L of the grouch bound we computed in this chapter. We could make the same observation also on the level of Simulink , in fact any choice of the value θ_1 in the initial condition will not affect the rest of dynamics.

Based on all those arguments, it is possible to use a reduced 3 dimensional state-space version of the ODE, and this by not incorporating θ_1 in the Dynamics.

The new state-space model sums up to

$$\begin{bmatrix} \ddot{\theta}_1 \\ \dot{\theta}_2 \\ \ddot{\theta}_2 \end{bmatrix} = \begin{bmatrix} f_2(\dot{\theta}_1, \dot{\theta}_2, \theta_2) \\ f_3(\dot{\theta}_1, \dot{\theta}_2, \theta_2) \\ f_4(\dot{\theta}_1, \dot{\theta}_2, \theta_2) \end{bmatrix}$$

Fortunately, we had not to restart any previous implementation steps for the new model (Growth bound compilation…), All we had to do is to bring some slight modifications into the previous SCCOTS, mainly by shifting variables indices.

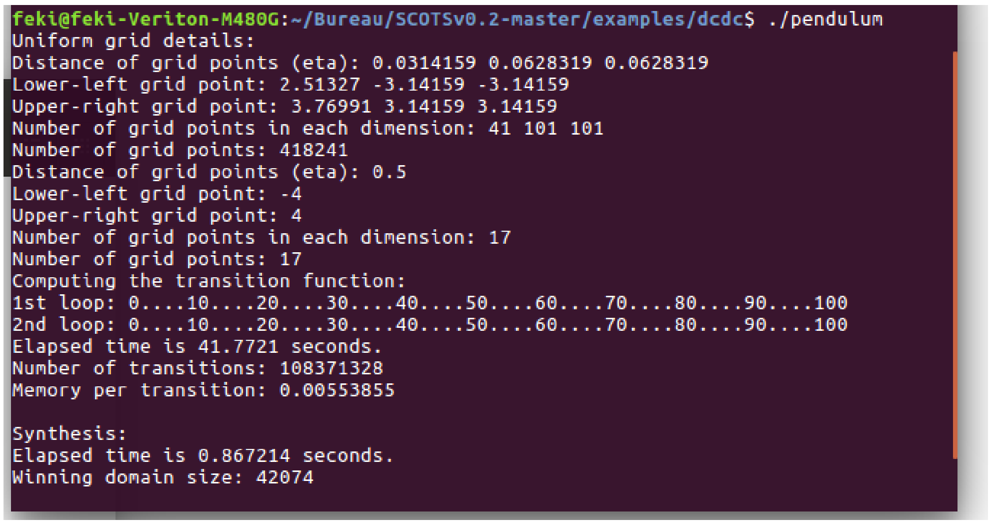
In addition, we had not to start again simulation on Simulink, and used advantage of the knowledge that we have previously gained from simulating the 4-dimensional model.

Our first guess of parameters:

- Specification:$\left[\pi - \frac{2\pi}{35}, \pi + \frac{2\pi}{35}\right] \times [-\pi, \pi\ ] \times [\ -\pi, \pi\ ]$
- State-space: $\left[\pi - \frac{\pi}{5}, \pi + \frac{\pi}{5}\right] \times [-\pi, \pi\ ] \times [\ -\pi, \pi\ ]$
- Sampling time:0.05
- Input-space: $[-4,4]$
- Input-discretization: 0.5
- Space-discretization: $[\ \frac{\pi}{100}, \frac{\pi}{50}, \frac{\pi}{50}]$

Which is obtained simply by projecting the old specification through a projection map.

We insert our choice of parameters in SCOTS, and finally we could end up fast in finding a controller with no empty wining domain. [fig 6.3]. The 3 dimensional Model has proven itself efficient to overcome the curse of dimensionality we have suffered from within the 3 dimensional one.



```
feki@feki-Veriton-M480G:~/Bureau/SCOTSv0.2-master/examples/dcdc$ ./pendulum
Uniform grid details:
Distance of grid points (eta): 0.0314159 0.0628319 0.0628319
Lower-left grid point: 2.51327 -3.14159 -3.14159
Upper-right grid point: 3.76991 3.14159 3.14159
Number of grid points in each dimension: 41 101 101
Number of grid points: 418241
Distance of grid points (eta): 0.5
Lower-left grid point: -4
Upper-right grid point: 4
Number of grid points in each dimension: 17
Number of grid points: 17
Computing the transition function:
1st loop: 0....10....20....30....40....50....60....70....80....90....100
2nd loop: 0....10....20....30....40....50....60....70....80....90....100
Elapsed time is 41.7721 seconds.
Number of transitions: 108371328
Memory per transition: 0.00553855

Synthesis:
Elapsed time is 0.867214 seconds.
Winning domain size: 42074
```

Figure 6.4 controller synthesis with parameters, memory and computational details.

The obtained controller has a winning domain size of 42 074, from a total number of 418 241, which represent a ratio ρ ≈10%, defined as the quotient of winning domain to state-space-domain.

We tried further to adjust all the parameters, in an attempt to find a better ratio ρ . However most of the modifications will result either in finding no controller, or in a very long computation time. After several experiments, the best we could figure out in order obtain a slightly larger winning domain was to opt a finer input discretization step while keeping the rest of parameters unchanged, finally we decided to set $\mu = 0.25$ instead of 0.5.

Upon finishing the compilation and the synthesis of the controller, a file controller.scs is generated containing the abstract controller. [fig 6.5]

```
#SCOTS:i (state) j_0 ... j_n (valid inputs)
#MATRIX:DATA
#BEGIN:418241 33

4201 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
4202 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
   .
   .
   .
4568 29
4569 26 27 28 29 30
4570 23 24 25 26 27 28 29 30
```

Figure 6.6 text fragment from controller.scs containing the abstract controller

each line of controlle.scs contains an identifier corresponding to a cell within the winning domain, followed by a list of identifiers standing for the admissible inputs at each cell. Some states (example state 4202) admit a bench of possible inputs, while others admit only one (example state 4568)

The full c++ SCOTS code, that was used to find the abstract controller could be found in Annex A.

Chapter 7: simulation and Validation of the abstract controller.

1)overview:

So far, we have generated an abstract controller to enforce an invariance specification on the pendulum, maintaining it near the instable upwards position. Although we know from the theory that the controller is correct-by-construction, we are still interested in verifying that is functioning correctly in practice, to ensure for example that we did not commit any mistakes while deriving the model dynamics, or computing the growth bound.to test our controller, we have two possibilities:

- Simulate the controller on a pc, and check whether the specification is respected.
- Refine the controller to a real plant, and verify the good functioning of the plant, which should conform to the specifications.

In the next steps, we are going to both simulate the controller on a pc and test it on a real pendulum.
For the simulation, we have two possibilities offered by SCOTS:

- Use a simulation c++ file, that we should modify according to our example. Then simulate the behavior of the controller given an initial condition, and a fixed simulation time. The behavior of the pendulum will be visualized on the terminal console. The only problem is that we still need to compare values by ourselves for every simulation step to verify that the behavior of the pendulum corresponds to the specifications.
- Use a matlab simulation file, which allows us to visualize the abstract domain, the specification, and the sampled behavior of the system(trajectories)
  Given an initial state and a simulation time horizon. This visualization option is very convenient for tow dimensional state-space models, but is harder for higher dimension, where we have to visualize all 2-D projections of the state-space separately.

However, the best way to simulate and validate the controller is still to test it on a real pendulum. For the purpose we installed a pendulum on a lab and conducted experiments to check the correctness of all our work.

2) Controller simulation on computer:

2.1) Simulation of the closed-loop on the terminal-console:

in order to do that we had to modify an existing c++ simulation files, by adding the dynamics of the pendulum, choosing an initial state, and the number of simulation steps. We have also to verify the initial state is within the winning domain. We can for example set an initial state $x_0 = (\pi \quad 0 \quad 0)$ which corresponds to the upward position without initial angular velocities, and we simulate the closed loop for 100 time steps[figure 7.1]

the full c++ code for simulation is provided in Annex B.

the problem with this simulation approach is that we have to check every line in the terminal-console and compare the values with the specification, to ensure it is respected.

```
  std::cout << "\nSimulation:\n ";
  /* initial state */
  state_type x={{M_PI,0 0}};

    /* iterate */
  for(int i=0; i<100; i++) {
    std::vector<input_type> u =
con.get_control<state_type,input_type>(x);
    std::cout << x[0] <<  " "  << x[1] << " " <<  x[2] << "\n";
    system_post(x,u[0]);
  }
```

Figure 7.1 code fragment from the c++ simulation file to fix simulation parameters.

2.2) simulation of the closed-loop on Matlab:

SCOTS provides the option, through a Matlab interface, to simulate the synthesized controller, visualize the abstract domain, specification domain, and the sampled evolution of the trajectories. For the purpose we have to modify a Matlab file by inserting the ODE, a certain care should be paid to indices because C_{++} and Matlab use different indexing rules.as in the previous case, we have to select an initial state, an a simulation time horizon. In order to visualize results through plots,we have to consider 2-d projections of the state-space, then we could check if the trajectories lies within the safe domain[Figure 7.2].the full Matlab code for simulation is provided in Annex C.

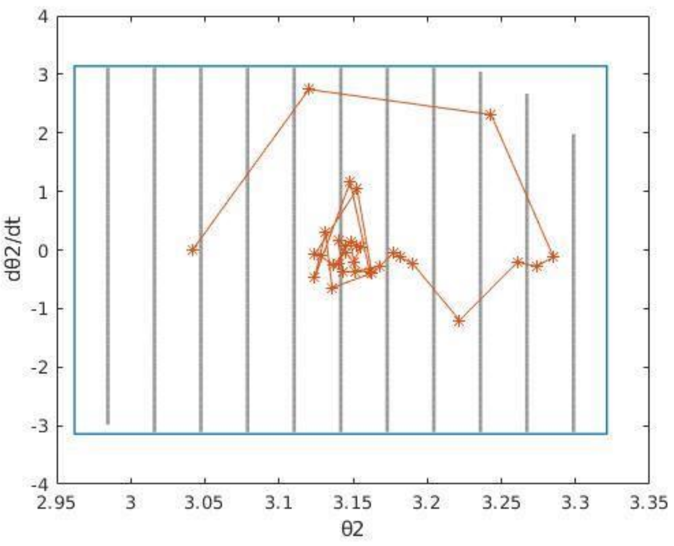


Figure 7.2 an example of a plot obtained by Matlab simulation, the blue lines corresponds to the invariance specifications while the orange lines to a trajectory, we can check that for a trajectory starting within the safe domain, it will remain within it

Remark: although the designed controller is based on a reduced three dimensional model, we still have the option to either simulate the 3-dimensional or 4 dimensional one. In fact ,the angle θ_1 that we did not include in state-space domain in the reduced model, could be reconstituted during the simulation by a simple integrator.

3) Controller simulation on a real pendulum:

3.1) Installation of the experimental setup:

For the real experiments, we used a simple rotatory inverted pendulum manufactured by Quanser®, and we had to carry all the Hardware and Software installations by ourselves.

- First we installed Quarck, a Software provided with Quanser ®to connect the Hardware components with the PC. We had also to install some further tools required for the proper functioning of Quarck.
- We installed a VoltPAQ-X1 Amplifier [figure 7.4], a Q8-USB Data Acquisition Device [figure 7.5]and a SRV02 Rotary Servo Base Unit [figure 7.6],.
- we mounted the simple rotatory pendulum module [figure 7.8] on the Servo base

Figure 7. 4 VoltPAQ-X1 Amplifier

Figure 7.5 Q8-USB Data Acquisition Device

Figure 7.6 SRV02 Rotary Servo Base Unit

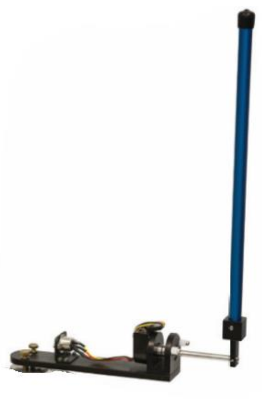


Figure 7.7 simple rotatory pendulum module

we wanted to test the correctness of the installation by running a state-feedback controller provided by Quanser ® to stabilize the pendulum. We encountered however many difficulties while running this controller. We had to test all the components separately to identify the problem. After many tests, we could figure out that the problem was due to some malfunctioning cables. We had to replace them and test the controller again, finally we could check that everything was working properly.

3.2) loading the abstract controller in Matlab:

SCOTS generates a controller.scs file that contains the abstract controller. We tried to load this file to Matlab directly, but we failed because of some incompatibilities.

As an alternative, it was suggested to load the controller through a text file, that we could generate by modifying the c++ simulation file [figure 7.8] in the code, we had to resort to try and catch command, because if the try to read an input value from a non-winning domain cell in the, we will get a compilation error.

```
   /* iterate */
   std::ofstream myfile;
   std::vector<input_type> u ;
   myfile.open ("lookup.txt");
 for(int i=0; i<41; i++) {
     for(int j=0; j<101; j++){
 for(int k=0; k<101; k++){
  x={{M_PI-M_PI/5+i*M_PI/100,-M_PI+j*M_PI/50,-M_PI+k*M_PI/50}};
 try {
      u = con.get_control<state_type,input_type>(x);

             myfile << i << ","<<j << ","<<k<<","<< u[0][0] << "\n";


  } catch (...) {
      myfile << i << ","<<j << ","<<k<<","<<10<< "\n";
     continue;

  }
```

Figure 7.8 code fragment to generate an abstract controller as a text file

After compiling the modified c++ simulation file, a text.cc file will be generated. This text files is composed of 418 241 lines, every line corresponds to a state alphabet. Within each line there 4 entries, the first three entries are integers that describes the cell position in the space-state domain, the fourth value corresponds to a possible admissible input value if the cell is within the winning domain, and the value 10 otherwise (the choice of 10 is arbitrary, just to indicate that there is no possible input value for this state). A fragment of the text file is depicted in figure 7.9.

```
18,95,74,10
18,95,75,0.75
18,95,76,0.5
18,95,77,0.25
18,95,78,0.25
18,95,79,0
18,95,80,-1.25
18,95,81,-1.5
18,95,82,-1.75
18,95,83,-2
18,95,84,-2.5
18,95,85,-3
18,95,86,-3.25
18,95,87,-3.75
18,95,88,-3.75
18,95,89,-3.75
18,95,90,-3.75
18,95,91,-4
18,95,92,-4
```

Figure 7.9 fragment of the text file containing the abstract controller

then we import this text file to Matlab, and convert it to a four dimensional table the obtained table table could be interpreted as look-up table, which for each cell, described by the triplet (i, j, k),associates one corresponding admissible input value if the cell lies within the winning domain, and the value 10 otherwise.

Because manipulating such a table is not easy in Matlab and Simulink, we thought it be better to convert this table into a vector. This is possible, since we can describe each cell with a unique identifier instead of a triplet (i, j, k), This is done by constructing a linear, injective map.

$$\varrho : \mathbb{R}^3 \to \mathbb{R}$$
$$(i, j, k) \to 101.101.(i-1) + 101 * (j-1) + k$$

we implemented a Matlab function that convert the lookup-table to a vector K.[figure 7.10]

```
d=1;
R=zeros(1,41*101*101);
for i = 1:41
    for j = 1:101
        for k = 1:101
          R(1,d)=lookup.u(d);
           d=d+1;
        end
    end
end

d=101*101*(i-1)+101*(j-1)+k;
```

Figure 7.10 Matlab code to generate a vector K from the look-up table

3.3) Controller refinement to the real plant:

One of the advantage of that feedback refinement relation offers is the smoothness the refinement process, which could easily be obtained through a serial composition of a static quantizer with the abstract controller C o Q .[figure 7.11]

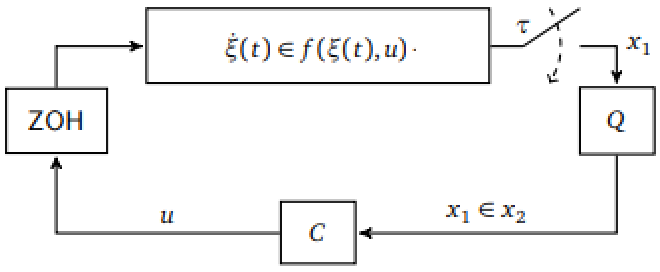
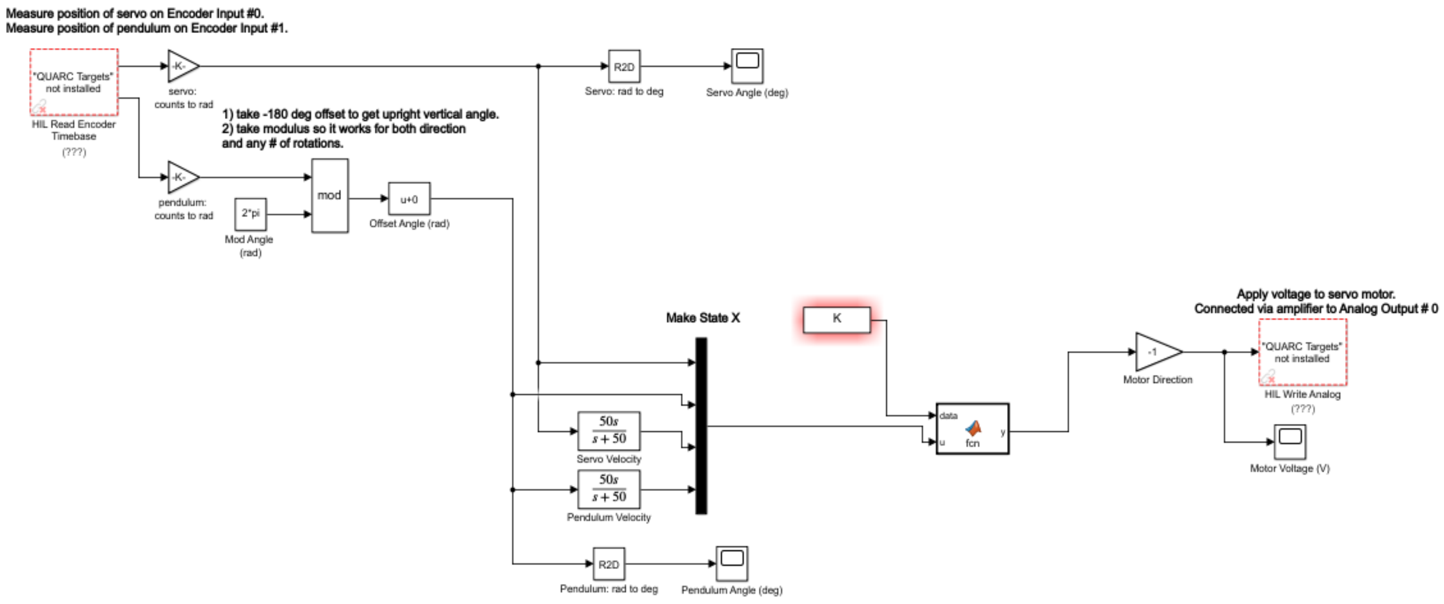


Figure 7.11 refinement process of the abstract controller

To obtain a refined controller we do the following steps:

1)First we build a Simulink model, obtained by modifying an already-existing state-feedback based controller provided by Quanser. [figure 7.12]

2)We load the vector Which contains the abstract controler from Matlab variables-workspace.



Measure position of servo on Encoder Input #0.
Measure position of pendulum on Encoder Input #1.
"QUARC Targets" not installed
HIL Read Encoder Timebase (???)
-K-
servo: counts to rad
-K-
pendulum: counts to rad
1) take -180 deg offset to get upright vertical angle.
2) take modulus so it works for both direction
and any # of rotations.
2*pi
Mod Angle (rad)
mod
u+0
Offset Angle (rad)
R2D
Servo: rad to deg
Servo Angle (deg)
Make State X
K
50s / s+50
Servo Velocity
50s / s+50
Pendulum Velocity
R2D
Pendulum: rad to deg
Pendulum Angle (deg)
data
u
fcn
y
-1
Motor Direction
Apply voltage to servo motor.
Connected via amplifier to Analog Output # 0
"QUARC Targets" not installed
HIL Write Analog (???)
Motor Voltage (V)

Figure 7.12 Simulink model used to control the real pendulum

3)We construct a Matlab function [figure 7.13] which contains a static quantize. for every measurement values obtained from the sensor, this functions associate the corresponding symbol or cell through a quantizer, check whether this cell lies within the winning domain, and finally looks for the correct input voltage from the data (vector K). The input value will be returned to the amplifier, which will apply a voltage on the servo motor.

```
function y = fcn(data, u)
y=0;
i=round((u(2)-(pi-pi/5))*(100/pi))+1;
j=round((u(3)-(-pi))*(50/pi))+1;
k=round((u(4)-(-pi))*(50/pi))+1;
if((i<42)&&(i>0))
   if((j<102)&&(j>0))
       if((k<102)&&(k>0))
       d=101*101*(i-1)+101*(j-1)+k;
       if(data(d,1)<5)
             y=data(d,1);
              else
           y=0;
       end
       end
   end
end
```

figure 7.13 Matlab main function in the Simulink Model

3.4) results of the experiments:

After we finished building the Simulink model, we compile it, connect it to the target, then run it. We bring the pendulum near to the upwards position, once the pendulum lies within the winning domain, the controller starts to work and the pendulum is robustly stabilized in the upward positive. The only minor problem we faced is that the rotatory arm started to spin around so fast in one direction. The controller still enforces the right specification, but the result is visually not nice.[figure 7.14]

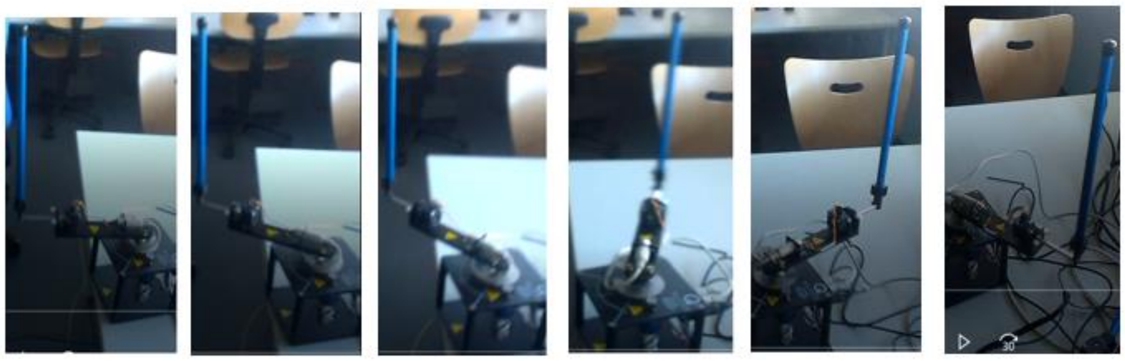


Figure 7.14 result of the first experiment, the pendulum is stabilized but the rotatory arms ins spinning around

We had to think why this was happening, later we figured that problem is due to the manner we chose the inputs while generating the test file contain thee abstract controller. In fact ,we have picked for each cells always the first admissible input .by this ways, we have favored input signals or voltages that tend to rotate the arm in a biased direction.

To cure this inconvenience, we tried to modify again c++ simulation file either by:

- Choose for every symbol in the winning domain a random input from the set of the admissible one [figure 7.15].
- choose the first input for one cell and alternate for the next cell by choosing the last input from the list.

```
int w=0;
int p=0;

for(int i=0; i<41; i++) {
for(int j=0; j<101; j++){
for(int k=0; k<101; k++){

x={{M_PI-M_PI/5+i*M_PI/100,-M_PI+j*M_PI/50,-M_PI+k*M_PI/50}};

try {

u = con.get_control<state_type,input_type>(x);

p=u.size();
w=rand()%p;



myfile << i << ","<<j << ","<<u[0][w]<<","<< v << "\n";

}
```

Figure 7.15 code fragment for generating a file text of an abstract controller with random choice from the admissible input

We tried the two possibilities, which both leads to a better visual results the problem. We could now obtain the balance of the pendulum in the upwards positive, while keeping the rotatory arm centered around an oscillating position.

Considering the results, we obtain from the experiments, we could verify the correctness of the controller and the steps we have carried so far.

Conclusions and outlooks:

The aim of this thesis was to automatically generate a correct-by-construction controller for a rotatory inverted pendulum, which enforces an invariance specification. For the implementation, we used SCOTS, a tool for the synthesis of symbolic controller based on the feedback refinement relation. We simulated the closed-loop behavior on Matlab, then refined the abstract controller to the real plant. We conducted some experiments, and we checked the correctness of our work.

the greatest challenge we faced was to find the optimal choice of parameters in SCOTS, that would result in finding a controller, while respecting some computational and memory performance limits. We had to rely on exhaustive simulation, and some intuition in attempt to estimate those parameters, however we failed to a find a controller with the four-dimensional model, due to the explosive exponential memory and time complexity, inherent to the curse of dimensionality of gridded-based abstractions. To overcome this curse, we had to consider a possible reduction of the model dimension, which turns out to be feasible in our case, we helped us to obtain the desired controller.

It would be interesting if the future research on symbolic methods focus on finding techniques to cure the curse of dimensionality, both on a theoretical and algorithmic level.

On a theatrical level, we could for example investigate the possibility of constructing a state-space discretization-free abstraction, like the one developed for δ-ISS systems (Zamani & Jagtap, QUEST: A tool for state-space quantization-free synthesis of symbolic controllers., 2017). On the algorithmic level, we could try to optimize the time complexity of the execution, for example by parallelizing the computation of transitions

Another approach would be to resort to Super-computers while synthesizing abstract controllers for sophisticated real-word applications.

Bibliographies

Boubaker, O. (2014). *The inverted Pendulum: A fundamental Benchmark in Control Theory and Robotics.*

Majid Zamani, M. R. (n.d.). *SCOTS: A Tool for the Synthesis of Symbolic Controllers.*

Metni, N. (2009). *Neuro-control of an inverted pendulum using Genetic Algorithm.*

Mladenov, V., Tsenov, G., Ekonomou, L., & Harkiolakis, N. (2009). *Neural network control of an inverted pendulum on a cart.*

Prime, B. S. (2011). *On the Dynamics of the Furuta Pendulum.*

Quanser. (n.d.). *User Manual Inverted Pendulum Experiment.*

Reissig, G., Rungger, M., & Alexander, W. (2015). *Feedback Refinement Relations for the Synthesis of Symbolic Controller.*

Roose, A. I., SamerYahya, & Al-Rizzo, H. (2017). *Fuzzy-logic control of an inverted pendulum on a cart.*

Rungger, M. (2017). *SCOTS (0.2) – USER MANUAL.*

Sarnovsky, S., & Jadlovska, J. (2013). *Modelling of Classical and Rotatory Inverted Pendulum Systems – A Generalized Approach.*

Tabuada, P. (2009). *Verification and Control of Hybrid Systems.*

Zamani, M., & Jagtap, P. (2017). *QUEST: A tool for state-space quantization-free synthesis of symbolic controllers.*

Zamani, M., Pola, G., Mazo JR, M., & Tabuada, P. (2011). *Symbolic Models for Nonlinear Control Systems Without Stability Assumptions.*

Annex A: SCOTS c++ code for generating a controller

```
/*
 * pendulum.cc
 *
 *  created: november 2018
 */

#include <iostream>
#include <array>
#include <cmath>

/* SCOTS header */
#include "scots.hh"
/* ode solver */
#include "RungeKutta4.hh"


/* time profiling */
#include "TicToc.hh"
/* memory profiling */
#include <sys/time.h>
#include <sys/resource.h>
struct rusage usage;


/* state space dim */
const int state_dim=3;
/* input space dim */
const int input_dim=1;
/* sampling time */
const double tau = 0.05;

/*
 * data types for the elements of the state space
 * and input space used by the ODE solver
 */
using state_type = std::array<double,state_dim>;
using input_type = std::array<double,input_dim>;

/* abbrev of the type for abstract states and inputs */
using abs_type = scots::abs_type;

/* parameters for system dynamics */
const double g=9.81;
const double jj1=9.982910141666664*0.0001;
const double j2=0.001198730801458;
const double m1=0.2570;
const double m2=0.127;
const double L1=0.2159000;
const double L2=0.336550;
const double l1=0.061912500000000;
const double l2=0.155575000000000;
const double b1=0.002400000000000;
const double b2=0.002400000000000;
const double Rm=2.600000000000000;
const double Ng=0.900000000000000;
const double Nm=0.690000000000000;
const double Kg=70;
const double Kt=0.007682969729280;
const double Km=0.007677634454753;
const double q1=Ng*Kg*Nm*Kt/Rm;
const double q2=Ng*Kg*Kg*Km*Nm*Kt/Rm;
const double J1=jj1+m1*l1*l1;
const double J2=j2+m2*l2*l2;
const double J0=J1+m2*L1*L1;
const double c=m2*L1*l2;
/* parameters for radius calculation */
const double mu=std::sqrt(2);
/* we integrate the pendulum ode by 0.05 sec (the result is stored in x)
*/
auto system_post = [](state_type &x, const input_type &u) noexcept {
  /* the ode describing the dcdc converter */
  auto rhs =[](state_type& xx,  const state_type &x, const input_type &u)
noexcept {

      const double det=J0*J2+J2*J2*sin(x[0])*sin(x[0])-
c*cos(x[0])*c*cos(x[0]);

      xx[0]=x[2];
      xx[1]=(1/det)*(-J2*b1*x[1]+c*cos(x[0])*b2*x[2]-
J2*J2*sin(2*x[0])*x[1]*x[2]-
0.5*J2*c*cos(x[0])*sin(2*x[0])*x[1]*x[1]+J2*c*sin(x[0])*x[2]*x[2]
              +g*0.5*c*c*sin(2*x[0])/L1+J2*(q1*u[0]-q2*x[1]));
      xx[2]=(1/det)*(x[1]*c*cos(x[0])*b1-
x[2]*b2*(J0+J2*sin(x[0])*sin(x[0]))+x[1]*x[2]*c*J2*cos(x[0])*sin(2*x[0])-
0.5*x[1]*x[1]*sin(2*x[0])*(J0*J2+J2*J2*sin(x[0])*sin(x[0]))-
              0.5*x[2]*x[2]*c*c*sin(2*x[0])-
g*m2*l2*sin(x[0])*(J0+J2*sin(x[0])*sin(x[0]))-c*cos(x[0])*(q1*u[0]-
q2*x[1]));

      };
  scots::runge_kutta_fixed4(rhs,x,u,state_dim,tau,5);
};
/* we integrate the growth bound by 0.05 sec (the result is stored in r)
*/
auto radius_post = [](state_type &r, const state_type&x, const input_type
&u) noexcept {
  /* the ode for the growth bound */
  auto rhs =[x](state_type& rr,  const state_type &r, const input_type &u)
noexcept {

      const double det=J0*J2+J2*J2*sin(x[0])*sin(x[0])-
c*cos(x[0])*c*cos(x[0]);
      const double
detd=2*J2*J2*sin(x[0])*cos(x[0])+2*c*sin(x[0])*c*cos(x[0]);
```

```
      const double f3=-J2*b1*x[1]+c*cos(x[0])*b2*x[2]-
J2*J2*sin(2*x[0])*x[1]*x[2]-
0.5*J2*c*cos(x[0])*sin(2*x[0])*x[1]*x[1]+J2*c*sin(x[0])*x[2]*x[2]
              +g*0.5*c*c*sin(2*x[0])/L1+J2*(q1*u[0]-q2*x[1]);
       const double f4=x[1]*c*cos(x[0])*b1-
x[2]*b2*(J0+J2*sin(x[0])*sin(x[0]))+x[1]*x[2]*c*J2*cos(x[0])*sin(2*x[0])-
0.5*x[1]*x[1]*sin(2*x[0])*(J0*J2+J2*J2*sin(x[0])*sin(x[0]))-
               0.5*x[2]*x[2]*c*c*sin(2*x[0])-
g*m2*l2*sin(x[0])*(J0+J2*sin(x[0])*sin(x[0]))-c*cos(x[0])*(q1*u[0]-
q2*x[1]);
      const double  L21=fabs((1/(det*det))*(-f3*detd+det*(-
c*sin(x[0])*b2*x[2]-
2*J2*J2*cos(2*x[0])*x[1]*x[2]+0.5*J2*c*sin(x[0])*sin(2*x[0])*x[1]*x[1]-
J2*c*cos(x[0])*cos(2*x[0])*x[1]*x[1]+g*c*c*cos(2*x[0])/L1)));
      const double  L22=abs((1/det)*(-J2*b1-J2*J2*sin(2*x[0])*x[2]-
J2*c*cos(x[0])*sin(2*x[0])*x[1]-J2*q2));
      const double  L23=fabs((1/det)*( c*cos(x[0])*b2-
J2*J2*sin(2*x[0])*x[1]+2*J2*c*sin(x[0])*x[2]));
      const double  L31=fabs((1/(det*det))*(-f4*detd+det*(-
x[1]*c*cos(x[0])*b1-x[2]*b2*2*J2*sin(x[0])*cos(x[0])-
x[1]*x[2]*c*J2*sin(x[0])*sin(2*x[0])+2*x[1]*x[2]*c*J2*cos(x[0])*cos(2*x[0])
-x[1]*x[1]*cos(2*x[0])*(J0*J2+J2*J2*sin(x[0])*sin(x[0]))-
x[1]*x[1]*sin(2*x[0])*J2*J2*cos(x[0])*sin(x[0])-x[2]*x[2]*c*c*cos(2*x[0])
              +c*cos(x[0])*(q1*u[0]-q2*x[1])-
g*m2*l2*cos(x[0])*(J0+J2*sin(x[0])*sin(x[0]))-
g*m2*l2*sin(x[0])*(2*J2*cos(x[0])*sin(x[0])))));
      const double
L32=fabs((1/det)*(c*cos(x[0])*b1+x[2]*c*J2*cos(x[0])*sin(2*x[0])-
x[1]*sin(2*x[0])*(J0*J2+J2*J2*sin(x[0])*sin(x[0]))+c*cos(x[0])*q2));
      const double  L33=abs((1/det)*(-
b2*(J0+J2*sin(x[0])*sin(x[0]))+x[1]*c*J2*cos(x[0])*sin(2*x[0])-
x[2]*c*c*sin(2*x[0])));

      rr[0]=r[2];
      rr[1]=L21*r[0]*L22*r[1]+L23*r[2];
      rr[2]=L31*r[0]*L32*r[1]+L33*r[2];

      };
  scots::runge_kutta_fixed4(rhs,r,u,state_dim,tau,5);
};

int main() {
  /* to measure time */
  TicToc tt;

  /* setup the workspace of the synthesis problem and the uniform grid */
   /* grid node distance diameter */
  state_type eta={{M_PI/100,M_PI/50,M_PI/50}};
 /* lower bounds of the hyper-rectangle */
  state_type lb={{M_PI-M_PI/5,-M_PI,-M_PI}};
  /* upper bounds of the hyper-rectangle */
  state_type ub={{M_PI+M_PI/5,M_PI,M_PI}};

  scots::UniformGrid ss(state_dim,lb,ub,eta);
  std::cout << "Uniform grid details:\n";
  ss.print_info();

  /* construct grid for the input alphabet */
  /* hyper-rectangle [1,2] with grid node distance 1 */

  /* construct grid for the input space */
  /* lower bounds of the hyper rectangle */
  input_type i_lb={{-4}};
  /* upper bounds of the hyper rectangle */
  input_type i_ub={{4}};
  /* grid node distance diameter */
  input_type i_eta={{0.25}};
  scots::UniformGrid is(input_dim,i_lb,i_ub,i_eta);
  is.print_info();

  /* compute transition function of symbolic model */
  std::cout << "Computing the transition function:\n";
  /* transition function of symbolic model */
  scots::TransitionFunction tf;
  scots::Abstraction<state_type,input_type> abs(ss,is);


  tt.tic();
  abs.compute_gb(tf,system_post, radius_post);
  tt.toc();
  std::cout << "Number of transitions: " << tf.get_no_transitions() <<"\n";

  if(!getrusage(RUSAGE_SELF, &usage))
    std::cout << "Memory per transition: " <<
usage.ru_maxrss/(double)tf.get_no_transitions() << "\n";

  /* continue with synthesis */
  /* define function to check if the cell is in the safe set  */
  auto safeset = [&lb, &ub, &ss, &eta](const scots::abs_type& idx) noexcept
{
    state_type x;
    ss.itox(idx,x);
    /* function returns 1 if cell associated with x is in target set  */
    if (lb[0]+M_PI/7<= (x[0]-eta[0]/2.0) && (x[0]+eta[0]/2.0)<= ub[0]-
M_PI/7 &&
        lb[1] <= (x[1]-eta[1]/2.0) && (x[1]+eta[1]/2.0) <= ub[1] &&
        lb[2] <= (x[2]-eta[2]/2.0) && (x[2]+eta[2]/2.0)<= ub[2])
      return true;
    return false;
  };
  /* compute winning domain (contains also valid inputs) */
  std::cout << "\nSynthesis: \n";
  tt.tic();
  scots::WinningDomain win = scots::solve_invariance_game(tf,safeset);
  tt.toc();
  std::cout << "Winning domain size: " << win.get_size() << "\n";

  std::cout << "\nWrite controller to controller.scs \n";

if(write_to_file(scots::StaticController(ss,is,std::move(win)),"controller"
))
    std::cout << "Done. \n";

  return 1;
}
```

Annex B: SCOTS c++ code for simulating the controller .

```
/*
 * simulate.cc
 *
 *  created: November 2018
*/
#include <iostream>
#include <array>
#include <fstream>

/* SCOTS header */
#include "scots.hh"
/* ode solver */
#include "RungeKutta4.hh"

/* state space dim */
const int state_dim=3;
/* input space dim */
const int input_dim=1;
/* sampling time */
const double tau = 0.05;

/*
 * data types for the elements of the state space
 * and input space used by the ODE solver
 */
using state_type = std::array<double,state_dim>;
using input_type = std::array<double,input_dim>;

/* parameters for system dynamics */
const double g=9.81;
const double jj1=9.982910141666664*0.0001;
const double j2=0.001198730801458;
const double m1=0.2570;
const double m2=0.127;
const double L1=0.2159000;
const double L2=0.336550;
const double l1=0.061912500000000;
const double l2=0.155575000000000;
const double b1=0.002400000000000;
const double b2=0.002400000000000;
const double Rm=2.600000000000000;
const double Ng=0.900000000000000;
const double Nm=0.690000000000000;
const double Kg=70;
const double Kt=0.007682969729280;
const double Km=0.007677634454753;
const double q1=Ng*Kg*Nm*Kt/Rm;
const double q2=Ng*Kg*Kg*Km*Nm*Kt/Rm;
const double J1=jj1+m1*l1*l1;
const double J2=j2+m2*l2*l2;
const double J0=J1+m2*L1*L1;
const double c=m2*L1*l2;
/* parameters for radius calculation */
const double mu=std::sqrt(2);

/* we integrate the penmdulum ode by 0.05 sec (the result is stored
in x)  */
auto system_post = [](state_type &x, const input_type &u) noexcept {
  /* the ode describing the dcdc converter */
  auto rhs =[](state_type& xx,  const state_type &x, const
input_type &u) noexcept {
      const double det=J0*J2+J2*J2*sin(x[0])*sin(x[0])-
c*cos(x[0])*c*cos(x[0]);

      xx[0]=x[2];
      xx[1]=(1/det)*(-J2*b1*x[1]+c*cos(x[0])*b2*x[2]-
J2*J2*sin(2*x[0])*x[1]*x[2]-
0.5*J2*c*cos(x[0])*sin(2*x[0])*x[1]*x[1]+J2*c*sin(x[0])*x[2]*x[2]
             +g*0.5*c*c*sin(2*x[0])/L1+J2*(q1*u[0]-q2*x[1]));
      xx[2]=(1/det)*(x[1]*c*cos(x[0])*b1-
x[2]*b2*(J0+J2*sin(x[0])*sin(x[0]))+x[1]*x[2]*c*J2*cos(x[0])*sin(2*x
[0])-0.5*x[1]*x[1]*sin(2*x[0])*(J0*J2+J2*J2*sin(x[0])*sin(x[0]))-
             0.5*x[2]*x[2]*c*c*sin(2*x[0])-
g*m2*l2*sin(x[0])*(J0+J2*sin(x[0])*sin(x[0]))-c*cos(x[0])*(q1*u[0]-
q2*x[1]));

        };
  scots::runge_kutta_fixed4(rhs,x,u,state_dim,tau);
};


int main() {
  /* read controller from file */
  scots::StaticController con;
  if(!read_from_file(con,"controller")) {
    std::cout << "Could not read controller from controller.scs\n";
    return 0;
  }

  std::cout << "\nSimulation:\n ";
  /* initial state */
  state_type x={{M_PI,0 0}};

    /* iterate */
  for(int i=0; i<100; i++) {
    std::vector<input_type> u =
con.get_control<state_type,input_type>(x);
    std::cout << x[0] <<  " "  << x[1] << " " <<  x[2] << "\n";
    system_post(x,u[0]);
  }

  return 1;
```

Annex C: Matlab code for simulating the controller .

```
%
% pendulum.m
%
% created: novemebr 2017
%
% you need to 1. have the mexfiles compiled
%             2. run the ./pendulum binary first
%
% so that the file: controller.scs is created
%

function pendulum
clear set
close all

%% simulation

% initial state
x0=[pi 0 0 0];
tau_s=0.05;

% load controller from file
controller=StaticController('controller');

% simulate closed loop system
y=x0;
v=[];
loop=100;
while(loop>0)
     loop=loop-1;

  u=controller.control(y(end,:));

  %-------------here choose your controller input-------------%
  %in=u(end,:);
  in=u(end,:);

  %-----------------------------------------------------------%

  v=[v; in];
  [t x]=ode45(@unicycle_ode,[0 tau_s], y(end,:), odeset('abstol',1e-
10,'reltol',1e-10),in');

  y=[y; x(end,:)];
```

```
end
%% plot the state domain
% colors
colors=get(groot,'DefaultAxesColorOrder');

% plot controller domain
dom=controller.domain;


% plot initial state  and trajectory
plot(y(:,1),y(:,4),'x')
hold on


%set(gcf,'paperunits','centimeters','paperposition',[0 0 2
2],'papersize',[3 3])

% plot safe set

v=[pi-pi/5+pi/7 -pi;...
   pi+pi/5-pi/7  -pi;...
   pi-pi/5+pi/7  pi;...
   pi+pi/5-pi/7  pi ];
patch('vertices',v,'faces',[1 2 4
3],'facecolor','none','edgec',colors(2,:),'linew',1)
hold on

box on




end

function dxdt = unicycle_ode(t,x,u)
    % parameter initialization
 g=9.81;
jj1=9.982910141666664*0.0001;
j2=0.001198730801458;
 m1=0.2570;
 m2=0.127;
 L1=0.2159000;
 L2=0.336550;
 l1=0.061912500000000;
l2=0.155575000000000;
 b1=0.002400000000000;
 b2=0.002400000000000;
Rm=2.600000000000000;
 Ng=0.900000000000000;
 Nm=0.690000000000000;
 Kg=70;
 Kt=0.007682969729280;
 Km=0.007677634454753;
q1=Ng*Kg*Nm*Kt/Rm;
 q2=Ng*Kg*Kg*Km*Nm*Kt/Rm;
 J1=jj1+m1*l1*l1;
 J2=j2+m2*l2*l2;
 J0=J1+m2*L1*L1;
 c=m2*L1*l2;
 det=J0*J2+J2*J2*sin(x(1))*sin(x(1))-c*cos(x(1))*c*cos(x(1));
    dxdt = zeros(3,1);

            dxdt(1)=x(3);
            dxdt(2)=(1/det)*(-J2*b1*x(2)+c*cos(x(1))*b2*x(3)-
J2*J2*sin(2*x(1))*x(2)*x(3)-
0.5*J2*c*cos(x(1))*sin(2*x(1))*x(2)*x(2)+J2*c*sin(x(1))*x(3)*x(3)+g*
0.5*c*c*sin(2*x(1))/L1+J2*(q1*u-q2*x(2)));
            dxdt(3)=(1/det)*(x(2)*c*cos(x(1))*b1-
x(3)*b2*(J0+J2*sin(x(1))*sin(x(1)))+x(2)*x(3)*c*J2*cos(x(1))*sin(2*x
(1))-0.5*x(2)*x(2)*sin(2*x(1))*(J0*J2+J2*J2*sin(x(1))*sin(x(1)))-
0.5*x(3)*x(3)*c*c*sin(2*x(1))-
g*m2*l2*sin(x(1))*(J0+J2*sin(x(1))*sin(x(1)))-c*cos(x(1))*(q1*u-
q2*x(2)));
            dxdt(4)=x(2);
end
```